Burger, Csaba; Berndt, Mihály

Working Paper

Error spotting with gradient boosting: A machine learning-based application for central bank data quality

MNB Occasional Papers, No. 148

Provided in Cooperation with:
Magyar Nemzeti Bank, The Central Bank of Hungary, Budapest

Suggested Citation: Burger, Csaba; Berndt, Mihály (2023) : Error spotting with gradient boosting: A machine learning-based application for central bank data quality, MNB Occasional Papers, No. 148, Magyar Nemzeti Bank, Budapest

This Version is available at:
https://hdl.handle.net/10419/299276

CSABA BURGER

MIHÁLY BERNDT

ERROR SPOTTING WITH GRADIENT BOOSTING: A MACHINE LEARNING-BASED APPLICATION FOR CENTRAL BANK DATA QUALITY

MNB OCCASIONAL PAPERS | 148

2023

MAY

ERROR SPOTTING WITH GRADIENT BOOSTING: A MACHINE LEARNING-BASED APPLICATION FOR CENTRAL BANK DATA QUALITY

MNB OCCASIONAL PAPERS | 148

2023

MAY

MNB Occasional Papers 148

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

(Anomália-detekció gradient boosting eljárással: központi banki adatminőség-ellenőrzés gépi tanulással)

Written by Csaba Burger, PhD, CFA, data science advisor at Magyar Nemzeti Bank, Mihály Berndt, data scientist at Clarity Consulting

Budapest, May 2023

Published by the Magyar Nemzeti Bank

Publisher in charge: Eszter Hergár

H-1013 Budapest, Krisztina körút 55.

www.mnb.hu

ISSN 1585-5678 (online)

Contents

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 such relationships, outliers, are flagged as potential data errors.

This paper implements an outlier-based error-spotting algorithm using gradient boosting, and presents a blueprint for the modelling pipeline. More specifically, it underpins three main modelling hypotheses with empirical evidence, which are related to (1) missing value imputation, (2) the loss-function choice and (3) the location of the error. By doing so, 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 to test its hypotheses.

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, which is piecewise quadratic up until the Huber-slope parameter and linear above it, is better suited to cope with outlier values; it is therefore 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, and the modelling pipeline described underscores significant modelling decisions.

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

JEL codes: C5, C81, E58

Összefoglaló

Növekvő népszerűségnek örvendenek az olyan felügyelt gépi tanulási módszerek az adathibák azonosításában, amelyekben nem szerepelnek hibacímkék. Az ilyen algoritmusok arra támaszkodnak, hogy az adatok egyfajta alapigazságot tükröznek, azaz azt feltételezik, hogy a megfigyelések többsége helyes. Az így felállított függvényeknek nem megfelelő pontokat – kiugró értékeket – potenciális adathibaként jelölünk meg.

Ebben a cikkben egy anomália-detekcióra épülő hibafeltáró algoritmust mutatunk be *extreme gradient boosting* módszer (xgboost) felhasználásával, és megvizsgáljuk a kapcsolódó modellezési folyamatot. Ennek során három fő modellezési hipotézist fogalmazunk meg, amelyet empirikus úton igazolunk; ezek (1) a hiányzó érték kezelésére, (2) a veszteségfüggvény kiválasztására és (3) a hiba helyéhez meghatározására vonatkoznak. A hipotézisek igazolásához MNB Hitelregiszter adatbázisának egy keresztmetszeti nézetét választottuk, amelyben a hitelbírálatkori hitelfedezeti arányt (LTV) és kapcsolódó oszlopait elemeztük, valamint szintetikusan előállított hibák megtalálási arányát vizsgáltuk.

A tanulmány azt mutatja, hogy a *gradient boost*ing eljárás hatékonyságát nem befolyásolja érdemben a hiányzó adatok pótlásának módszere, ezért a számításilag leghatékonyabb, konstanssal való helyettesítés javasoljuk. Másodszor, a Huber-veszteségfüggvény, amely négyzetes a *Huber-slope* paraméterig, és lineáris felette, jobban kezeli az extrém kiugró értékeket a tanulás során, így az adathibákat is hatékonyabban találja meg. Végül, a módszer a célváltozóban található adathibákat tárja fel legnagyobb arányban, amiket a prediktorokban szinte egyáltalán nem azonosít. Az empirikus eredményeink az adatok sajátosságaitól függően általánosíthatóak, és a cikkben leírt lépések segítséget nyújthatnak modellezési döntések megalapozott meghozatalában.

1 Introduction

Central bank data collection processes have been gradually transforming from the collection of aggregate values to the compilation of granular, contract or security-level information. At the same time, the explosion of data volumes has been offering more avenues for data quality investigations to uncover possible data errors.

Granular central bank data has unique characteristics. First, it is usually delivered by a manageable number of data providers, which prepare their contribution by reading legal text and extracting data from their data warehouses accordingly. This may introduce data specificities or even errors on provider level. Second, such granular 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, for other loan types, missing values are rampant. 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 economic or financial relationship.

Once rule-based data errors are taken care of (format and constraint violations, pattern violations (values that violate syntactic and semantic constraints), duplicates, see 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. And although Benatti (2018) used extreme gradient boosting (xgboost) model to do so, which is a well-suited algorithm to deal with non-linearity between variables, many of his modelling decisions, and their implications for error detection, were scantly discussed.

This paper is an empirical study of a gradient boosting error detection mechanism: it proposes a modelling pipeline to uncover data-related anomalies and potential data errors in a central bank collected granular data set. By doing so, it looks for errors in the data, and calculates how two target metrics change when certain modelling decisions are made. The following target metrics are computed on the test data set: (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 considered are the imputation of missing values, the choice of the xgboost loss function and the location of the error (whereby an error in the target variable seems to be more easily discovered). 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. We believe that our pipeline can serve as a blueprint for modelling decisions with other datasets too.

First, 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. Second, an xgboost with a more robust, Huber loss function seems capture more data errors than using the squared error loss function. Finally, we show that 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. While our empirical results may or may not be specific to the dataset used, the paper discusses potential implications of the decisions during the modeling pipeline, and it thereby 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, including the description of our hypotheses. The fourth section discusses the data and methodology in detail, while the fifth presents the outcome of the error simulations. The final section concludes.

2 Supervised learning for data error spotting

The concept of data quality and of 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 between 0 and 100), duplicates (which are best captured with record-linkage algorithms), or to outliers (Abedjan et al., 2016). In contrast to that, outlier-based errors represent a subset of that, 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 are capable of doing 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, outlier detection assumes the existence of a "ground truth", meaning a relationship between variables which are expected to represent the baseline mechanism. For data quality purposes the 'ground truth' hypothesis assumes that the *majority* 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 is also expected to be present in the train data. In addition, this means that any algorithm only flags outliers if they break the underlying learned function, even if some real errors (e.g. incorrect data records) are part of it.

Using supervised learning for outlier detection may not be the first choice. Data errors are most often untagged, resulting in the fact that 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 well 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 start with the assumption that there is a true relationship in the data between the variables, and erroneous observations do not reflect this relationship. They propose a method called 'attribute-wise learning for scoring outliers' (ALSO), in which a target variable is explained with the help of 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 score[1] for each observation. The approach of Paulheim and Meusel (2015) can be viewed as a transformation of the data points into a new residual feature space. Within it, outliers can be considered as observations that are far away from the origin. 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.

Benatti (2018) follows the method of 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. One may assume that central banks as data collectors face similar error types. Second, his model was specified with a gradient boosting algorithm (xgboost), which provides sufficient flexibility to account for categorical

[1] If is the predicted and 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: . One may weigh the difference between and to reflect the strength of each relationship using their relative squared error values. With the weights assigned, the weighted residuals in the k-th model are: , and the weights are calculated by , where .

and data provider-level specificities. It is capable of learning highly non-linear relationships and variable interactions, works well with high dimensional data. In addition, it can deal with missing data within the modelling phase (Chen and Guestrin, 2016). In short, xgboost is an excellent tool to learn the complexity of true relationships within the data, hence, it is discussed in detail in the next section.

2.1 THE SPARSITY-AWARE LOGIC OF XGBOOST

In a nutshell, the xgboost algorithm entails a sequential development of several decision trees[2]. The first tree attempts to predict the target variable with a simple aggregation[3], the second tree predicts the prediction 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 set of trees, and its prediction is added to the last one. This process is repeated several times. The number of repetitions, meaning the number of trees can be determined by cross-validation or Bayesian Optimization, discussed later in this paper.

This section presents a formal overview of the xgboost, based on Chen and Guestrin (2016). The authors provide a more detailed overview of the algorithm their paper. While the reason for incorporating the exact algorithm in this paper is to specify the xgboost implementation used, as required by the editor, the paper does not contribute to the formulae themselves.

A tree ensemble model uses K additive functions to predict the output.

$$\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \sum_{\boldsymbol{k}=\boldsymbol{1}}^{\boldsymbol{K}} \boldsymbol{f}_{\boldsymbol{k}}(\boldsymbol{x}_{\boldsymbol{i}}), \;\; \boldsymbol{f}_{\boldsymbol{k}} \in \boldsymbol{\mathcal{F}}, \tag{1}$$

where 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:

$$\boldsymbol{\mathcal{L}}^{(t)} = \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{n}} \boldsymbol{l}(\boldsymbol{y}_{\boldsymbol{i}}, \hat{\boldsymbol{y}}_{\boldsymbol{i}}^{(\boldsymbol{t}-\boldsymbol{1})} + \boldsymbol{f}_{\boldsymbol{t}}(\boldsymbol{x}_{\boldsymbol{i}})) + \boldsymbol{\Omega}(\boldsymbol{f}_{\boldsymbol{t}}) \,. \tag{2}$$

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:

$$\widetilde{\boldsymbol{\mathcal{L}}}^{(t)} = \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{n}} \left[\boldsymbol{g}_{\boldsymbol{i}} \boldsymbol{f}_{\boldsymbol{t}}(\boldsymbol{x}_{\boldsymbol{i}}) + \frac{\boldsymbol{1}}{\boldsymbol{2}} \boldsymbol{h}_{\boldsymbol{i}} \boldsymbol{f}_{\boldsymbol{t}}^{\boldsymbol{2}}(\boldsymbol{x}_{\boldsymbol{i}}) \right] + \boldsymbol{\Omega}(\boldsymbol{f}_{\boldsymbol{t}}) \,, \tag{3}$$

whereby $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ 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:

$$\widetilde{\boldsymbol{\mathcal{L}}}^{(\boldsymbol{t})}(\boldsymbol{q}) = \frac{\boldsymbol{1}}{\boldsymbol{2}} \sum_{\boldsymbol{j}=\boldsymbol{1}}^{\boldsymbol{T}} \frac{\left(\sum_{\boldsymbol{i} \in \boldsymbol{I}_{\boldsymbol{i}}} \boldsymbol{g}_{\boldsymbol{i}}\right)^{\boldsymbol{2}}}{\sum_{\boldsymbol{i} \in \boldsymbol{I}_{\boldsymbol{i}}} \boldsymbol{h}_{\boldsymbol{i}} + \boldsymbol{\lambda}} + \boldsymbol{\gamma} \boldsymbol{T}. \tag{4}$$

An important insight here is that in an 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

[2] 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.

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

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:

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

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.

A major 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).

There are several steps in the modelling pipeline which were left unanswered by the papers discussed above. Therefore, the next section discusses some of the decision points we have identified and may help future analysts using the method for error spotting.

3 Machine learning pipeline and its preprocessing decisions

We have identified three points we found to be crucial during the preprocessing phase. Specifically, we discuss (1) missing value treatment options (besides letting xgboost deal with missing values, there are imputation options), the (2) use of various loss functions in xgboost (in a continuous setting, squared loss and the Huber-loss function), and (3) the identification of the erroneous variable, which helps in looking for data errors, once the modelling is finished and the results are handed back to the data provider. In addition, the remainder of the paper describes further crucial modelling aspects, including rare categorical value transformations, encoding, and the handling of quasi constant variables.

The first step in the process leading up to the modelling concerns the way to deal with missing values among predictors. One possibility is to make use of the sparsity-aware method of xgboost, which puts missing instances in the direction with the greatest loss reduction. Using this may reduce the model bias on the train dataset, but this may result in high prediction errors during the prediction phase, since the underlying 'ground truth' is not learnt by the model.

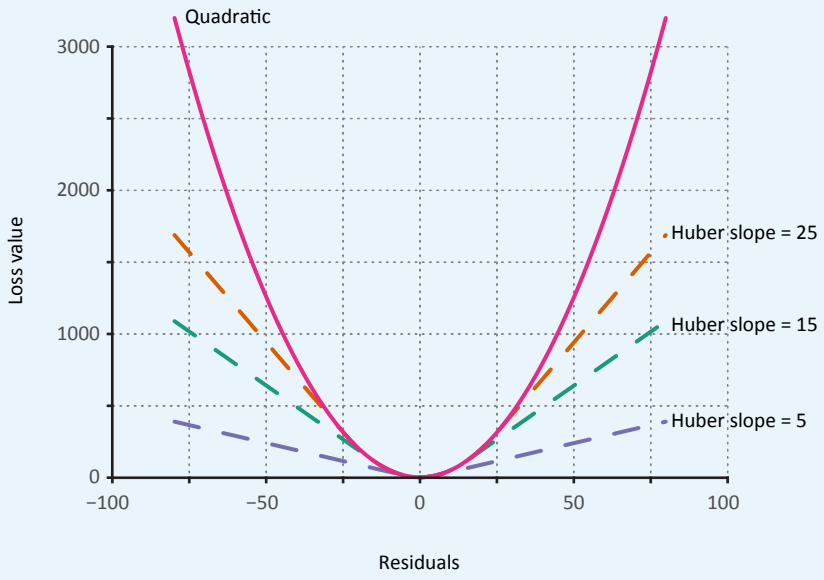
Another option is to revert to the broad literature on missing value treatment options, and to use explicit missing value replacement *before* the modelling phase. This may be the case if the user wishes to use imputed values for further analysis. For instance, Emmanuel et al. (2021) describe several types of estimator functions. These can be as simple as a replacement with a constant, or, a learner-function based prediction using cases where all values are filled. Most scholars agree that that 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). At the same time, Twala et al. (2008) do not find convincing difference in the performance of a classification tree when comparing a simple 'missing' flag with an imputation approach. We also argue that a sufficiently flexible tree-based embedded outlier detection algorithm will not require a sophisticated missing value imputation function, where noise and potential errors can occur in the data. **We argue that replacement with a constant value and the addition of an imputed value flag variable (1/0) is not inferior to an imputation algorithm. In addition, we recommend to be 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. Since our estimator function wishes to capture the true relationship between the variables, it should not be materially impacted by the data errors we are about to identify. While this may seem to be a chicken or egg problem, a robust estimator function may just serve our purpose. Begashaw and Yohannes (2020) provide a neat overview of such approaches in a linear regression setting; Sadouk et al. (2020) propose a similar method for deep learning applications. In our case, with gradient boosting, we propose changing the default objective function, which is minimized during the model fitting. More specifically, we argue for replacing the squared loss function with the Huber loss function. This latter makes sure that the 'distance', which the model minimizes, denoted as *r* (for residual), 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 , and its value is determined during the hyperparameter-optimization process, as discussed later. The simple[4] Huber-loss function is reflected by **Equation 6** and illustrated by **Figure 1**.

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

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

Figure 1
Comparing the loss values of a quadratic and of several Huber-loss functions with different slope values



The Huber-function ensures that small deviations are squared, permitting fine-tuning for such data points. Large deviations between estimates and actuals count in a linear manner only. As a result, the parameter tuning will not chase 'detected' outliers during optimization and can learn the underlying ("true") function better. **We argue that an xgboost developed with the Huber-loss function is better suited to uncover data errors than the squared loss objective function (Hypothesis 2).**

Finally, an xgboost learner function is developed, estimations are calculated, and residuals are computed. All outliers, meaning that observations with residuals above a certain threshold, are investigated for being a potential data error. In contrast to a full ALSO-algorithm, we stop after developing an xgboost for one target column only. The reason for that is to support interpretability: **we argue that potential errors, concerning one column only, 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'. Furthermore, B may contribute to the approximation of the underlying "true" function in a less precise way than 'A' could. In sum, if an outlier is detected, it may be a good practice to look for the data error within the target variable and not in the explanatory variables. (A process which may start by looking at record level feature contributions.)

4 Data and Methodology

4.1 THE DATA

The Credit Register (Hitelregiszter or Hitreg) of the Central Bank of Hungary (MNB) is a monthly updated collection of all loan contracts issued by Hungarian banks, containing 22 reporting tables, 482 attributes and 31 identifiers. These together usually yield between 30-60 million new lines each month. The Hitreg 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 a subset of Hitreg, which is relevant for the loan-to-value (LTV) reporting of the MNB. This limits the analysis to mortgages and home equity loans of the residential sector, and issued after 1st October 2021, when the last change to the legal circumstances of the LTV-requirement was introduced. The final analytical table is a combination of information from several Hitreg reporting tables, comprising data on the debtor(s), on the collateral(s) and on the loan contract itself. 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 chosen to be the LTV-value, calculated for the time of credit issuance, which is subject to strict legal limitations, hence its accuracy is of utmost interest. Despite that LTV at issuance is only relevant before granting the loan, it is reported at each reporting date. As a logical consequence, we kept the columns which were relevant at the time of issuance (to illustrate: current outstanding loan amount was removed, since it has nothing to do with the LTV at issuance, despite probable correlations).

In theory, calculating LTV is a straightforward matter, obtained as the loan amount divided by the allocated collateral value. However, in practice, it is rather complex, given the fact that many decisions, including the choice of the collateral evaluation method and the allocation process of a collateral value to the specific loan contract, are at the data providers' discretion. Having analyzed the data set thoroughly, we are now confident that the columns available within the data are insufficient to calculate LTV directly. The relationship between LTV and the other columns is complex, which makes xgboost an appropriate choice for the analysis.

In short, the following statements justify the choice of LTV as the response variable in the model. 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.

4.2 MODELLING PIPELINE

The modelling pipeline starts with the arbitrary removal of variables exhibiting high multicollinearity (over 0.90). Although xgboost predictions are not affected by multicollinearity, feature importance may well be distorted. Since we expect certain degree of interpretability (for results' analysis, not discussed in this paper), we kept one out of the pairwise collinear variables only. We also eliminated quasi constant values, where the frequency ratio of the most frequent value and the second most frequent value is extremely high (hence the variance of the feature close to zero), since they may lead to overfitting or misleading conclusions.

In a similar manner, we recoded categorical and discrete variables with rare values to reduce the chance of overfitting caused by high variance and noise in the estimation. Any category below a threshold of 2.5 percent was recoded into an 'other' category, and the 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 non at random*' (MNAR) variables, and in such cases, simple flagging is preferred to imputation.

The next step was to create new meaningful features that could help the algorithm to learn. An illustrative example is date conversion: instead of using the contract start date in a raw format, we calculated the number of days elapsed between the start date and the reporting date.

The following stage of the pipeline was encoding (dealing with discrete non-numeric variables). During our experiments, we found that *label encoding*[5] improves model performance most even for variables that enumerations cannot be sorted (also known as nominal variables). Our intuition is that it can be explained by avoiding the "*curse of dimensionality*", which means that as the number of predictors increases, the difference between the maximum and minimum distances[6] 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**). To start with, 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. Based on that, and confirmed by **Figure 2,** there seems to be some degree of systematic rule for missingness, since certain feature values are missing for the same observations. This may be quite typical: some variables are not interpreted for one or another kind of loan contract. Hence, the decision between the imputation methods is just an artificial -technical question to feed the xgboost.

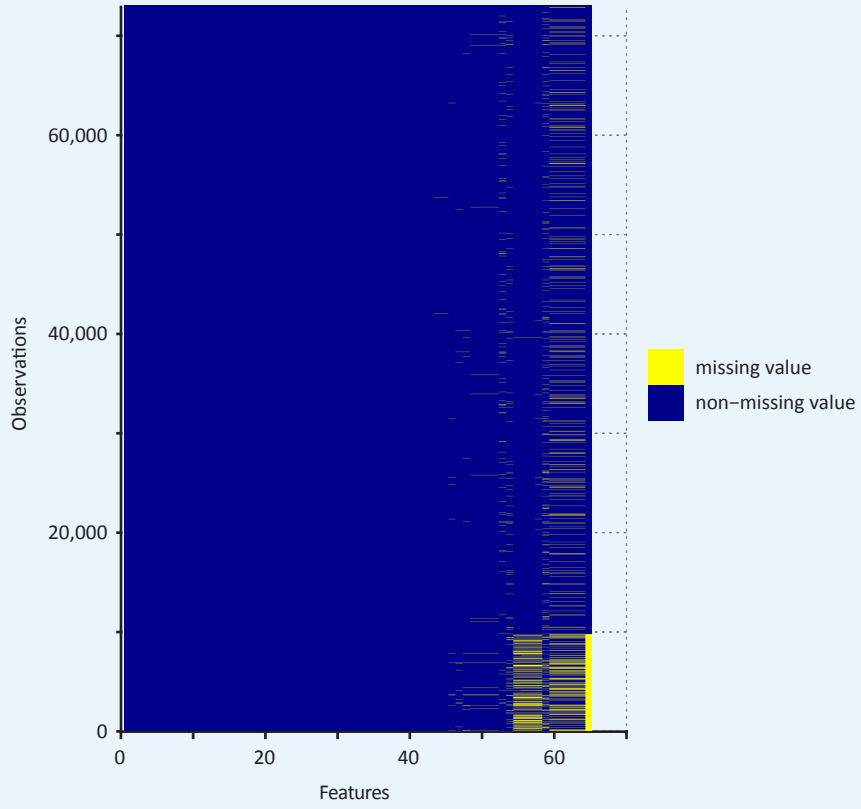
Three approaches to deal with missing values were assessed using the remaining columns. In the first approach, missing values were recoded into an arbitrary but relatively outlier value (e.g. collateral market value of zero HUF). We assume that a tree-based ensemble method such as the xgboost would learn that this is a special value not necessarily linked to an economic meaning.

Table 1
Missing value treatment options for Hypothesis 1

Missing value treatment	Description	Assumption
Replacement with a constant	The constant value used, zero, is an infrequent or nonexistent value in the original data set	Zero may be found where the data provider does not have values, but a non-null value is enforced. We assume that the xgboost is capable of learning that it is not an economic relationship but a special value. Additionally, if the ratio of missing values exceeds 5 percent, a new indicator column is added to indicate which rows are manually filled (1) and which are not[7] (0).
Imputation with MissRanger	All complete predictors are used (except for the target variable to prevent data leakage)	This approach assumes existing relationships between predictors, which is certainly present in the data set.
Using the 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 default class (i.e. 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'.

[5] 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.
[6] Maximum and minimum distances are distances calculated between the furthest and nearest observations respectively.
[7] The presence or absence of the additional flag column is not material: it changes little on model performance or variable interpretation.

Figure 2
Missing values among the continuous variables. Please refer to the Appendix for an overview of the share missing values and further descriptive statistics



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 (i.e. with zero missing values) 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 in an earlier section.

4.3 BAYESIAN OPTIMIZATION FOR HYPERPARAMETER-TUNING

The last step was the development of the xgboost function itself, which requires several decisions to be made during process. One of them is the choice of the loss function, subject to our second hypothesis and discussed above. The Huber slope parameter, which causes the loss function to be quadratic only up to a certain value, and linear above it, is determined with the help of a Bayesian optimization procedure (Snoek et al., 2012), along with all other hyperparameters. This is an automatic and efficient method that fits a Gaussian process to the results, where the goal is to find the set of hyperparameters that obtains the best result on the test data based on the selected evaluation metric (RMSE or the mean pseudo-Huber error).

In short, the Bayesian Optimization does the following. The objective function is the evaluation metric based on the hyperparameter setups, which function is unknown. The algorithm samples hyperparameter sets, observe the evaluation metric and updates the prior form of the function. There is a predefined similarity measure (Kernel) between the 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 formal representation of the Bayesian Optimization framework is cited from Shahriari et al. (2016) and Garrido-Merchán et al. (2023). To retrieve the optimum hyperparameters of the unknown objective function, we solve the following mathematical problem:

$$\boldsymbol{x}^* = \boldsymbol{arg}\ \boldsymbol{\min_{x \in X}}\ \boldsymbol{f}(\boldsymbol{x}). \tag{7}$$

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

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

where $f(x)$ is the optimized function, $\mathcal{GP}$ is a Gaussian Process, $\alpha(\cdot)$ is an acquisition function, and $p(f(x)|D)$ 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:

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

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 $\boldsymbol{X}$ and $\boldsymbol{X}^*$, $\boldsymbol{f}$ and $\boldsymbol{f}^*$ are the unobserved true function outputs for all the inputs, $\mathbf{K}_{\boldsymbol{X},\boldsymbol{X}}$ is N-by-N matrix containing the similarities between the observations calculated by the Radial Basis Function $\kappa(\boldsymbol{x_i}, \boldsymbol{x_i'}|\boldsymbol{\tau}) = \sigma^2 \exp\left(-\frac{1}{2}\left(\frac{x-x'}{\ell}\right)^2\right)$, where the hyperparameters of the Kernel denoted by $\boldsymbol{\tau}$, and $\widehat{\mathbf{K}}_{\boldsymbol{X},\boldsymbol{X}} = \mathbf{K}_{\boldsymbol{X},\boldsymbol{X}} + \sigma_\varepsilon^2 \mathbf{I}$. 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:

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

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

The predictive distributions are the estimates of unknown functions areas, and the next samples are determined by the acquisition function, which 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, which is the following:

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

The approach vastly reduced computation time compared to a brute force-based grid search. The optimization was parallelized to improve calculation times further. The permitted bounds for the hyperparameter-optimization, which were 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.

4.4 SYNTHETIC ERRORS AND MODEL EVALUATION

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

– LTV values were divided by 100 (since an LTV of 50 percent is expected to be submitted as '50', it is often submitted as 0,5)

– LTV values were set to 80 (as if a constant dummy value was submitted, which also happens in reality). This error type was applied to observations with a true LTV of 60 or less, in order to have a sufficiently sizeable error.

– LTV values were multiplied by a sample drawn from uniformly distributed ranges of U(0.3, 0.5) and U(1.2, 1.4)[8] random variables. This in essence means a multiplication by a value in any of those bounds. The two ranges are not symmetrical in order to limit the number of errors above 100 (percent) of LTV.

In a separate experiment set, similar errors were introduced to one of the predictors. We opted for introducing errors to the second most important predictor in terms of gain, cover, and frequency in the feature importance matrix, as described later, which is the allocated collateral value. The reason for this is the fact that its value is less strictly determined than for instance of the loan amount (the other most frequently occurring important feature, which 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 introduced to the predictor were:

• Allocated collateral values were multiplied by a sample of the uniformly distributed ranges U(0.3, 0.5) or U(1.2, 1.4) random variables. This in essence means a multiplication by a value in any of the mentioned bounds.

• Allocated collateral 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, in order to have a sufficiently sizeable error. The assumption behind this move can be the use of a dummy value, or an internal allocation cap within the reporting institution.

Table 3
Overview of data errors used for error spotting 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 constant 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

[8] We used the usual abbreviation of the uniform distribution (denoted by U).

The different outlier detection approaches were evaluated with the help of two metrics (**Table 4**). First, we look at the share of discovered errors out of all errors. An error is considered as discovered when its absolute residual was above a pre-defined threshold of 20 (given the fact that the target variable, LTV is expected to be between 0 and 80 (a legal limitation), the use of 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 relationship between the share of intentional errors within outliers, compared to the share of intentional errors in the whole test data set (referred to as 'lift'). With a good model, error density should be considerably higher in the outlier set than in the data.

We also tested two outlier definitions. First was based on a pre-defined (unstandardized[9]) 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	

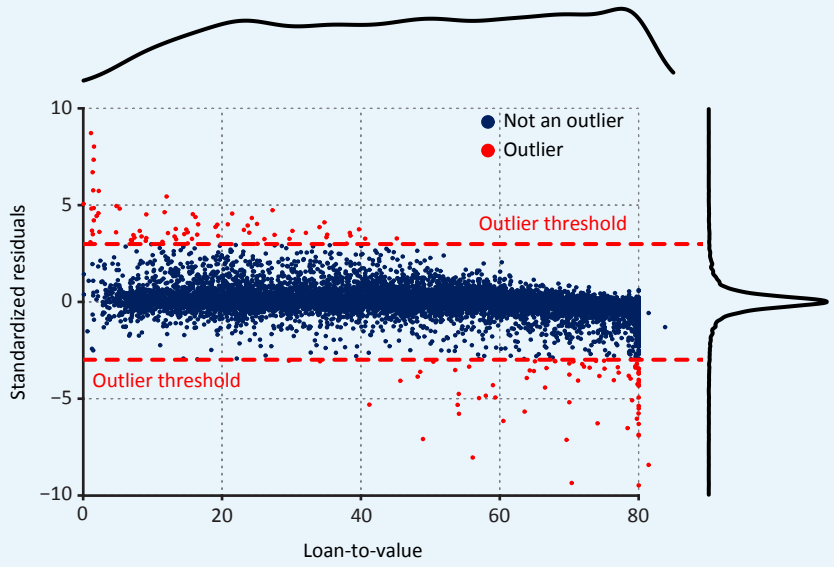
[9] We decided on using the unstandardized value for better interpretability, given the fact that LTV is expected to move between 0 and 100. In more heterogeneous cases or if the ALSO-algorithm is to be carried out in its full extent, one may standardize the residuals.

5 Results

5.1 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 (recall that LTV is used), and the share of outliers is 1.4 percent only, using the widely accepted cutoff value on *standardized* residuals of 3. These metrics show that the algorithm learns the underlying structure. Moreover, a sample of our flagged outliers were identified as data errors by domain experts. The algorithm also found intuitive errors, such as when the data provider sent LTV as a fraction between 0 and 1 instead of a percentage value. These observations are reflected by **Figure 3** as the overestimations near zero.

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**). Given the fact that 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 lies in the fact that banks are permitted to apply certain rules in using these values during LTV-calculation. They may use only a percentage of the total allocated value, cap it or modify it in any other manner. Similarly, it may not the entire loan amount which appears in the nominator.

Table 5
The top 10 most important features (feature importance) in the baseline model

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

Figure 4 and **Figure 5** depict the individual feature contributions for a randomly selected 10 thousand observations for the two most importance features (by the gain metric). The individual feature contribution is a weight the given value of the predictor has on the target variable. Since xgboost develops a complex set of trees (as described in section **4.2**), and predictor interactions are reflected by it, 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 has on the target variable.

These plots reflect an encouraging characteristic of the data: both the 'Loan amount' (which is related to the nominator) and the 'Allocated collateral value' (which is related to the denominator) are related to a division. As an example, small 'Loan amount' values typically correspond to negative contributions; and as we increase the value, the contribution increases as well. Note, however, the strange curve of the 'Allocated collaretal 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. Our intuition is that in these cases 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 the feature 'Loan amount' for a randomly selected 10 thousand observations (we added a LOESS-function line for illustration)

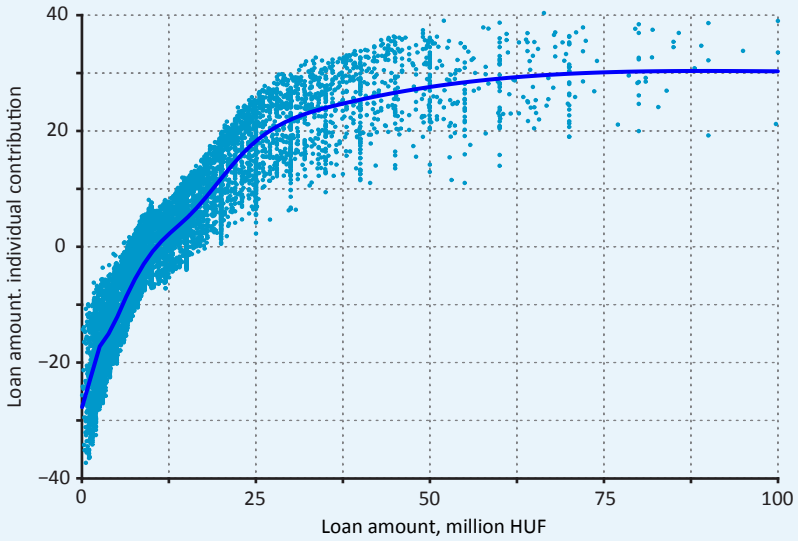
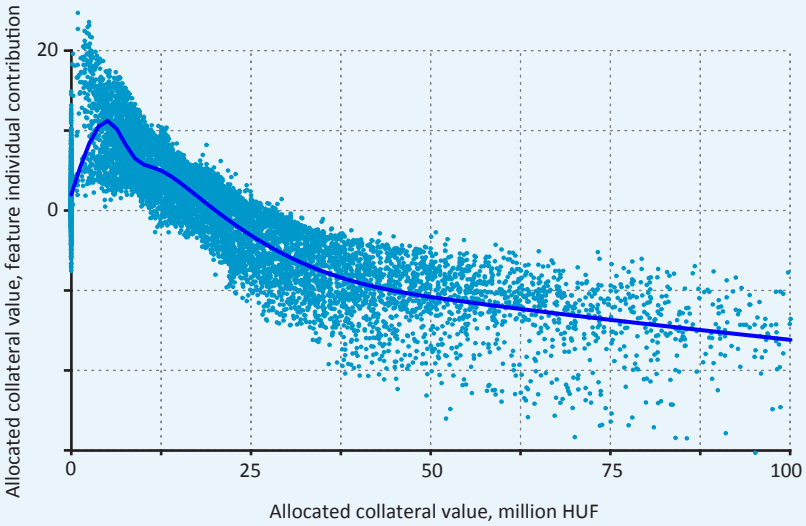


Figure 5
Individual feature contribution of the feature 'Allocated collateral value' for a randomly selected 10 thousand observations, (we added a LOESS-function line for illustration)



5.2 TESTING HYPOTHESES 1 AND 2

To test our hypotheses, we calculated the evaluation metrics with a static outlier threshold of 20. This means, all observations with an absolute residual of 20 or higher were classified as outliers. As **Table 6** shows, the 80-dummy and the div_100 error types are discovered with a great accuracy at a lift value of around 10-12, with the exception the 'none' (no explicit missing value replacement) and Huber-loss combinations. In this latter case, the share of captured 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
Error detection 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

In general, there is not necessarily a tradeoff between discovered error shares and the lift values. In fact, **Table 5** does not even say that the models with imputed missing value treatment would outperform models where missing values were imputed with a constant. It also shows that the Huber objective function performs better than the squared loss functions do in terms of the first metric, the share of discovered intentional errors. In addition, 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 performance 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. 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. **Figure 6** reinforces model stability and the messages of our research described above.

Figure 6
The False-positive-rate and true-positive rate development of each model when moving the absolute outlier threshold from 25 to 5, with synthetic errors in the target (response) variable

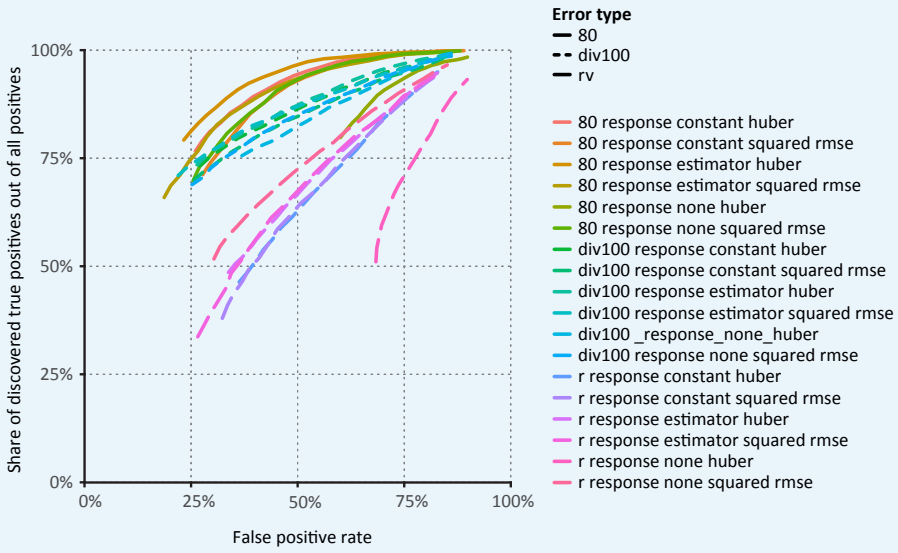


Table 7 contains model metrics when a relative outlier threshold of 5 percent of all observations is used, as opposed to the absolute outlier threshold discussed above. In a similar vein, **Table 7** confirms that the estimation of missing values as imputation in our case delivers better outlier recognition performance than using a constant value. As a reminder, during feature engineering we dropped the columns where the share of missing values exceeded 20 percent, hence the amount of potential added noise by imputation is reduced.

We can now report on our initial Hypotheses. The Huber function often outperforms the squared loss function in both metrics - but only if outliers are determined using a threshold as opposed to a firm percentage of all observations, and only if an explicit missing value imputation is selected. In short, the Huber-loss most often delivers better results, and an imputation is most often better than no imputations. This means that we can find indications which would confirm Hypotheses 1 and 2, but we would rather put our message in a humbler way. Every dataset should be tested for these potential candidates.

Table 7
Error detection metrics when using a relative outlier threshold of 5 percent of all observations; 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	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

5.3 TESTING HYPOTHESIS 3

To investigate Hypothesis 3, we run the same experiments as before but with slightly different error types. First, we used a random value multiplication; and second, we put allocated collateral value to 10 mln HUF in case of randomly selected observations with a true allocated collateral value of 40 mln HUF or more. The outcome of the simulations is summarized by **Table 8**.

Table 8
Error detection metrics when using an absolute outlier threshold of 20; with a synthetic error in the second most important predictor variable

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. Error detection is in fact not better than randomly selecting observations from the data. The metrics send a similar message when using a relative outlier threshold (not shown).

To understand the reason for that, we compare the feature importance matrix for the baseline model and for the corresponding model with the synthetic errors in the second most important predictor, the 'Allocated collateral value'. Admittedly, xgboost is a vastly complex algorithm with up to 280 trees permitted in each model. The feature importance matrix, which is a summary of the results, can only signal changes and will not deliver a mathematical proof. Nevertheless, it indicates change directions and provides a rather convincing argument.

Table 9 shows that the importance of this predictor decreases slightly from all three aspects (gain, cover and frequency). At the same time, the third predictor, the full collateral market value at origination, exhibits higher importance values. In addition, we can see various movements in variable importance values. These changes are admittedly tiny, but 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 predictor variable allocated collateral value (using the same missing value imputations (constant) and the RMSE loss function)

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*

6 Conclusion

In this paper, we analyzed a few technical aspects of using 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. In a nutshell, it relies on the assumption that the majority of the data points is correct, and that there are 'ground truth' relationships between the features in it, which can be non-linear. Data points deviating from such a relationship, outliers, are flagged as a potential data errors, which are to be followed upon by analysts and the data providers. While the results we found may be typical only 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 hypotheses relevant during the model development pipeline, we introduced three types of synthetic errors to the data.

The hypotheses tested relate to technical steps during model development. First, we tested the treatment of missing values in the data (when the data is missing both during model training and prediction). As our empirical investigation demonstrated, the computation-intensive imputation method most often proposed by the literature is not necessarily superior to using a constant value as a placeholder for 'missing' label for continuous variables. 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 the xgboost. Our empirical investigation showed that the Huber loss function is better suited to capture errors using this algorithm, compared to the traditional, squared loss function. We argue that using squared loss-function can prove to 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 supervised learning method to capture data errors most likely 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' erroneous predictor variables 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. This way consistency over time can be better ensured. 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.

7 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, Miss Forest 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.

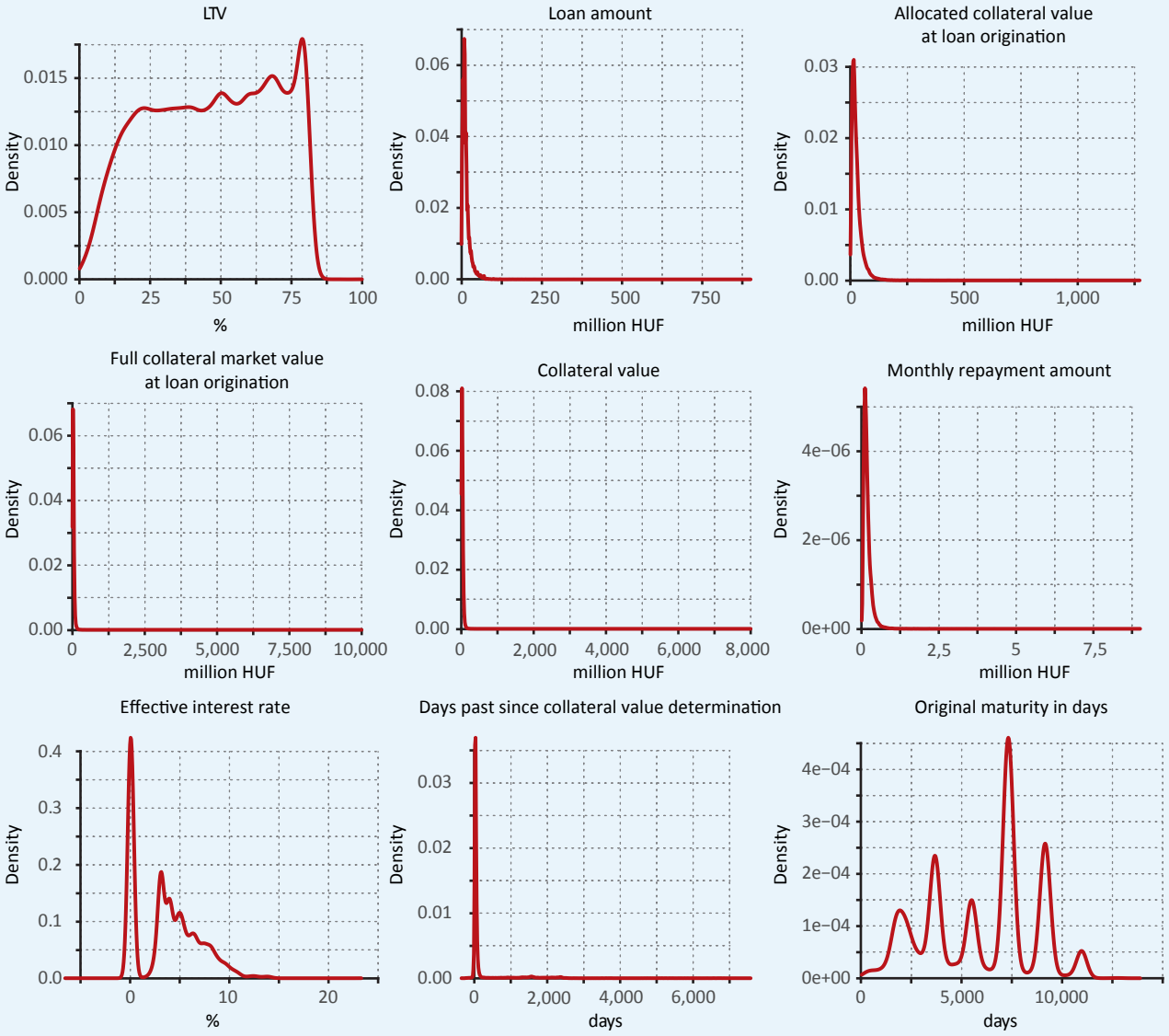
Twala, B. E. T. H.; Jones, M. C. and Hand, D. J. (2008). Good methods for coping with missing data in decision trees. Pattern Recognition Letters, 29(7) pp. 950–956.

8 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

MNB OCCASIONAL PAPERS 148

ERROR SPOTTING WITH GRADIENT BOOSTING:

A MACHINE LEARNING-BASED APPLICATION FOR CENTRAL BANK DATA QUALITY

May 2023

Print: Prospektus Kft.

6 Tartu u., Veszprém H-8200

mnb.hu

H-1013 BUDAPEST, KRISZTINA KÖRÚT 55.