Are you a good borrower? Mining interpretable pattern structures in credit scoring

Sergei O. Kuznetsov
National Research University Higher School of Economics, Moscow, Russian Federation

Alexey Masyutin
Sberbank, Moscow, Russia and National Research University Higher School of Economics, Moscow, Russia, and

Aleksandr Ageev
Sberbank, Moscow, Russia

Received 9 September 2020
Revised 11 September 2020
Accepted 11 September 2020

Abstract

Purpose – The purpose of this study is to show that closure-based classification and regression models provide both high accuracy and interpretability.

Design/methodology/approach – Pattern structures allow one to approach the knowledge extraction problem in case of partially ordered descriptions. They provide a way to apply techniques based on closed descriptions to non-binary data. To provide scalability of the approach, the author introduced a lazy (query-based) classification algorithm.

Findings – The experiments support the hypothesis that closure-based classification and regression allow one to both achieve higher accuracy in scoring models as compared to results obtained with classical banking models and retain interpretability of model results, whereas black-box methods grant better accuracy for the cost of losing interpretability.

Originality/value – This is an original research showing the advantage of closure-based classification and regression models in the banking sphere.

Keywords Credit scoring, Closed descriptions, Lazy classification, Pattern structures

Paper type Research paper

1. Introduction

Banks and credit institutions face classification problem each time they consider a loan application. A bank aims to have a tool to discriminate between solvent and potentially delinquent borrowers, i.e. the tool to predict whether the applicant is going to meet his or her obligations or not. Before 1950s, such decision-making process was expert-driven and

JEL classification – C10, C13, C14, C15
MSC2020 classification – 68T05, 68T09, 68T37

involved no explicit statistical modeling (Thomas *et al.*, 2002). The decision whether to grant a loan or not was made upon an interview and information about spouse and close relatives. From the 1960s, banks have started to adopt statistical scoring systems that were trained on data sets of applicants, consisting of their socio-demographic factors and loan application features (Thomas *et al.*, 2002; Siddiqi, 2005).

Classification algorithms can either produce so-called "black box" models with limited interpretability of model result, or, on the contrary, provide interpretable results and transparent model structure (Baesens *et al.*, 2003). As a rule, black-box models have superior accuracy and less sophisticated models may provide less accurate predictions.

This is also shown in previous work as soon as credit scoring problem has been approached with various statistical and machine learning techniques (Yap *et al.*, 2011; Lee *et al.*, 2006; Nanni and Lumini, 2009).

However, the key feature of banking risk management practice is that, regardless of the model accuracy, it should not be a black box. Regulators require that banks are able to provide reject reasons for borrowers and also when central banks examine the bank models they have to understand economic intuition behind them to prove the models are going to show expected and stable performance (bis.org, 2020; law.cornell, 2020).

That is why methods such as neural networks and support vector machines (SVM) classifiers did not earn much trust within banking community. The dividing hyperplane in an artificial high-dimensional space (dependent on the chosen kernel) cannot be easily interpreted to claim the reject reason for the client (Ghodselahi, 2011). As far as neural networks are concerned, they also do not provide the user with a set of reasons why a particular loan application has been approved or rejected. In other words, these algorithms do not provide a decision-maker with knowledge. The predicted class is produced, but no intuition is retrieved from data.

This paper introduces data analysis algorithms that have accuracy superior to simple algorithms widely adopted within the banks (such as logisitic regression, decision trees and scorecards) and still maintain the property of interpretability in sense that they provide a decision-maker with a set of rules applicable to assess the borrower.

2. Lattices of closed descriptions in classification problem

Methods such as generating association rules, emerging patterns and decision trees provide the user with easily interpretable rules which can be applied to the loan application. Algorithms based on formal concept analysis (FCA) also belong to this group of methods, as they use clearly defined concepts to classify objects (Ganter and Wille, 1999; Kuznetsov, 1999; Meddouri *et al.*, 2014). The intent of a concept can be seen as a set of rules supported by the extent of the concept. However, for non-binary data the computation of the concepts and their relations can be very time-consuming. In case of credit scoring we deal with numerical data, as soon as categorical variables can be transformed into a set of dummy variables. Lazy classification (query-based) (Aha, 1997) seems to be appropriate in this case, as it provides a decision-maker with a set of explicit rules for the loan application and can be easily parallelized.

2.1 Main definitions

Let G be a set (of objects), let D be a set of all possible object descriptions equipped with a "more general than" or subsumption partial order $\sqsubseteq$, which is a very natural requirement. For many description types this partial order induces an intersection operation, which is idempotent, commutative and associative (i.e. induces a semilattice on descriptions): for binary attributes this is just a set-theoretic intersection, for multisets this is component-wise min or max, etc. For sequences and graphs this is intersection on sets of graphs and sequences based maximal common subsequences and subgraphs (Kuznetsov, 1999; Kaytoue *et al.*, 2015). So, in what follows we will assume that the set of descriptions D is equipped with such an intersection $\sqcap$, such that given $c, d \in D$ one has $c \sqsubseteq d \leftrightarrow c \sqcap d = c$.

So, let (D, $\sqcap$) be a meet-semi-lattice of all possible object descriptions called *patterns* and let δ: G $\rightarrow$ D be a mapping taking each object to its description. Then (G, $\underline{D}$, δ), where $\underline{D} = (D, \sqcap)$, is called a *pattern structure* (Ganter and Kuznetsov, 2001).

Operation $\sqcap$ is also called a *similarity operation*. A pattern structure $(G, \underline{D}, \delta)$ gives rise to the following *derivation operators* $(\cdot)^{\diamond}$:

$$A^{\diamond} = \sqcap_{g \in A} \delta(g) \qquad \text{for } A \in G,$$

$$d^{\diamond} = \{g \in G \mid d \sqsubseteq \delta(g)\} \text{ for } d \in (D, \sqcap).$$

Here, $A^{\diamond}$ means similarity (as intersection) of all objects from set A, and $d^{\diamond}$ means the set of all objects with descriptions subsuming (i.e. more specific or equal to) d. The pairs (A, d) satisfying $A \subseteq G$, $d \in D$, $A^{\diamond} = d$, and $A = d^{\diamond}$ are called *pattern concepts* of $(G, \underline{D}, \delta)$ with *pattern extent* A and *pattern intent* d. Operator $(\cdot)^{\diamond\diamond}$ is a closure operator on patterns, as it is idempotent, extensive and monotone (Ganter and Kuznetsov, 2001).

Now consider a standard machine learning setting. Suppose we have a set of positive examples G_+ and a set of negative examples G_- w.r.t. a target attribute, $G_+ \cap G_- = \varnothing$. The objects from the set $G_\tau = G \setminus (G_+ \cup G_-)$ are called undetermined examples. A pattern $c \in D$ is called *α-weak positive classifier* iff the size of the set of negative examples it covers is no more than α-fraction of the size of the covered set of positive examples:

$$\frac{|c^{\diamond} \cap G_-|}{|c^{\diamond}|} \leq \alpha \text{ and } \exists A \subseteq G_+ : c \sqsubseteq A^{\diamond} \tag{1}$$

A pattern $h \in D$ is called an *α-weak positive hypothesis* iff:

$$\frac{|h^{\diamond} \cap G_-|}{|h^{\diamond}|} \leq \alpha \text{ and } \exists A \subseteq G_+ : h = A^{\diamond} \tag{2}$$

In case of credit scoring we work with pattern structures where descriptions are tuples of intervals of many-valued attributes. Instead of binarizing (scaling) data, one can directly work with many-valued attributes by applying interval pattern structure. For two intervals $[a_1, b_1]$ and $[a_2, b_2]$, with $a_1, b_1, a_2, b_2 \in \mathbb{R}$ the *meet operation* (or *similarity operator*) is defined as follows (Kaytoue *et al.*, 2011):

$$[a_1, b_1] \sqcap [a_2, b_2] = \left[min(a_1, a_2), max(b_1, b_2)\right]. \tag{3}$$

This definition may seem counterintuitive at the first glance, as intersection of two intervals gives a larger interval. The explanation is that the intersection gives actually less information, as the attribute values are allowed to vary in a larger interval.

To make these main definitions clear, let us provide an example and apply the meet operator (3) to a "toy" data set provided in Table 1.

	A	b
g_1	1	1.5
g_2	−1	0
g_3	0.5	1

Table 1. Toy data set for example 1

Example 1:

$$
\begin{aligned}
&G = \{g_1, g_2, g_3\},\\
&D = \{([a_1; a_2], [b_1; b_2]) | a_1, a_2, b_1, b_2 \in R\},\\
&\delta(g_1) = ([1; 1], [1.5; 1.5]),\\
&\delta(g_2) = ([-1; -1], [0; 0]),\\
&\delta(g_3) = ([0.5; 0.5], [1; 1]),\\
&d = \delta(g_1) \sqcap \delta(g_2) = ([-1; 1], [0; 1.5]),
\end{aligned}
$$

$d \sqsubseteq \delta(g_3)$, thus $d^\diamond = \{g_1, g_2, g_3\} = G$.

2.2 Lazy classification with pattern structures

To efficiently classify test objects one can employ the lazy learning approach (Veloso *et al.*, 2006; Veloso and Wagner, 2011), where one does not need to generate all possible good classifiers in advance. Having a similarity operation on object descriptions, one can compute similarity of the test object with the objects from the training set. Consider a pattern structure $(G(D,\sqcap),\delta)$ and suppose that we have a training set given by disjoint sets G_+, $G_- \subseteq G$ of positive and negative examples w.r.t. a target attribute, and a set of unclassified test objects G_τ. Then the value of the target attribute of a test object $g_n \in G_\tau$ appears in the closure of the intersection of description of g_n with descriptions of every object $g \in G$. If for some object g the closure contains the target attribute, then g_n is classified positively, otherwise the test object is classified negatively. More formally, this can be described as the following simple two-stage procedure:

(1) For every $g \in G$ compute $(\delta(g_n) \sqcap \delta(g))^\diamond$, i.e. select all objects from G whose descriptions contain $(\delta(g_n) \sqcap \delta(g))^\diamond$. This takes $O(|G|\ (p(\sqcap) + |G|\ p(\sqsubseteq)))$ time; and

(2) If for some $g \in G$ all objects from $(\delta(g_n) \sqcap \delta(g))^\diamond$ have the target attribute, classify g_n positively, otherwise negatively. This takes $O(|G|^2)$ time for looking for the target attribute in object descriptions in at most $|G|$ families of object subsets, each subset consisting of at most $|G|$ objects.

Lazy classification is thus reduced to computing $(\delta(g_n) \sqcap \delta(g))^\diamond$ and testing the target attribute in all objects of this set. This computation is easily parallelizable: one partitions the data set G in $G = G_1 \cup \ldots \cup G_k$, where k is the number of processors, computes the set of objects $(\delta(g_n) \sqcap \delta(g))^\diamond$ in each G_i, tests the target attribute for all objects in the union of these sets over i.

2.3 Query-based classification algorithm

In credit scoring, however, the original lazy classification setting may become inappropriate (Masyutin and Kuznetsov, 2016,Masyutin *et al.*, 2015). The reason is that the data is typically numerical, features can have arbitrary distributions and take wide range of values. At the same time categorical variables and dummies can occur. With relatively large number of attributes (over 100) it produces high-dimensional space of continuous variables. So, the meet operator (3) gives a very specific result, i.e. for almost every $g \in G$ only g and g_n have the description $\delta(g_n) \sqcap \delta(g)$. This happens owing to the fact that numerical variables, ratios especially, can have unique values for every object. This results in that for test object g_n the number of positive and negative classifiers is close to the number of examples in G_+, G_-, respectively.

Thus, it seems reasonable to seek the concepts with larger extents and with not too specific intent. At the same time, we would like to preserve the advantages of lazy classification, as we do not need to compute full concept lattice and one can take advantage of parallelization.

The query-based classification algorithm is our modification. Idea behind proposed algorithm is to check whether it is positive or negative class that test object is more similar to.

The first step is a *mining step*, when we extract α-weak hypotheses from the data. The procedure is carried out for G_+ and G_- separately. Random subsample of examples is extracted and their descriptions are intersected. The resulting description $d = \delta(g_1) \sqcap \ldots \sqcap \delta(g_s)$ is checked whether it is α-weak or not. If hypothesis is α-weak then it is added to a set of hypotheses to be used for classification later.

The size of subsample s is an algorithm's hyperparameter, and it is tuned via grid search. The number of times (i.e. number of iterations) we randomly select a subsample is the second hyperparameter, which is also tuned through grid search. As we mentioned, the greater the subsample size, the more it is likely that $(\delta(g_1) \sqcap \ldots \sqcap \delta(g_s))^\diamond$ contains an example of the opposite class. It is α threshold hyperparameter that controls this issue.

The second step is an *updating step*, when the test object description $\delta(g_n)$ is intersected with each α-weak hypothesis. If resulting description is also α-weak then it is considered to be α-weak classifier, i.e. the rule relevant for this particular test object g_n.

The third and final step is a *voting step*, when all α-weak classifiers vote to produce prediction for a test object g_n.

2.4 Voting schemes

The final classification for a test object is based on the voting of α-weak classifiers. In the most general case voting scheme F is a mapping:

$$F\left(g_n, c_1^+, \ldots, c_p^+, c_1^-, \ldots, c_n^-\right) \to [-1, 1, \varnothing]$$

where g_n is the test object with unknown class, c_i^+ is a positive α-weak classifier $\forall i = \overline{1, p}$ and c_j^- is a negative α-weak classifier $\forall j = \overline{1, n}$, -1 is the label for the negative class, and 1 is the label for the positive class (i.e. defaulters). In other words, F is an aggregating rule that takes classifiers to classification labels (empty label is allowed).

If the label is empty it is said that the algorithm abstains from classification. It can happen when there is no α-weak classifier found for the test object, which can be owing to poor algorithm tuning (e.g. inappropriately low α threshold or small number of iterations).

There may be different approaches to build up aggregating rules. The voting scheme is built upon weighting function $\omega(\cdot)$, aggregation operator $A(\cdot)$ and comparing operator $\otimes$.

$$\begin{aligned} &F(\omega(\cdot), A(\cdot), \otimes) = \\ &= \left(A_{i=1}^{p}\left[\omega\left(c_i^+\right)\right]\right) \otimes \left(A_{j=1}^{n}\left[\omega\left(c_j^-\right)\right]\right) \end{aligned}$$

To configure a new weighting scheme it is sufficient to define the operators and the weighting function. In this paper, the best weighting scheme (in terms of accuracy) is based on the relative number of positive versus negative α-weak classifiers, weighted by their confidence:

$$\begin{aligned} &A(\cdot) = \sum(\cdot), \\ &\omega(c) = 1 - \frac{|c^\diamond \cap G_{-/+}|}{|c^\diamond|}, \\ &a \otimes b = \frac{a}{b}. \end{aligned}$$

However, there are many different ways to build voting schemes, and a number of them can be found in Github code repository [1].

One can think of margin $a \otimes b$ as a measure for discrimination between two classes and consider, e.g. the decision boundary based on receiver operating characteristic analysis. As soon as decision boundary is defined (i.e. when $a \otimes b > x$ *then* 1 *else* -1), the voting scheme produces the predicted label.

3. Experiments with open data

In this paper we use open data sets for credit scoring. To compare our algorithm against benchmarks we keep some portion of data as validation sample, which is not used when training the model, and then we calculate performance metrics (Gini coefficient) for that sample. Where applicable we use grid search for hyperparameters to tune benchmark performance. The results are summarized in Table 2.

3.1 Lending club loan data

Lending Club is a large US peer-to-peer online credit platform. It has accumulated hundreds of thousands of payment profiles on loans being issued since 2007 till nowadays [2]. The data has been used widely as a benchmark when testing machine learning models (kaggle.com, 2020; triamus.github.io, 2020). We extracted 25,000 observations with nine features. They represent client credit history, loan term, rate, borrower's ownership information, income, employment length, etc. The target attribute is loan status which indicates whether the payments were made on time or not.

3.2 Credit Scoring Catalonia data

This data set is designed for studying purposes but, nevertheless, it is still applicable for benchmark analysis. The data consists of 4,456 observations and 13 numeric and categorical features with single target attribute [3]. Categorical variables were one-hot-encoded before applying algorithms. The features are similar to the ones in previous data sets.

3.3 Give me some credit data

"Give Me Some Credit" data set is taken from Kaggle contest held in 2012 [4]. The data has a binary target variable (class label) whether the borrower defaulted or not. We develop a scorecard and examine its accuracy via out-of-sample validation with provided target variable. The data set we used consists of 25,000 observations with ten numeric features. They describe client's status and previous credit experience and contain information on total balance on credit cards, monthly income, debt-to-income ratio, number of dependents (children, spouse), current revolving utilization limit, etc.

Table 2.
Experiments results (Gini coefficients)

	Lending club loan data	Credit scoring Catalonia	Give me some credit
Scorecard	0.5292	0.6487	0.7034
Random forest	0.5523	0.6979	0.7249
Decision tree	0.4374	0.5781	0.6071
kNN	0.3913	0.3962	0.2267
XGBoost	0.5654	0.7155	0.7304
QBCA	0.5576	0.7043	0.7281

3.4 Benchmarking: scorecards and black-box methods

We compare our algorithm to both classical algorithms adopted in banks and ML algorithms. As far as, classical algorithms are concerned, we use scorecards (Siddiqi, 2005) which are, in effect, logistic regressions run on transformed features. The features are transformed according to weight-of-evidence transformation (WOE) (stats.stackexchange.com, 2020; documentation.statsoft.com, 2020). The WOE-transformation was controlled for maximum number of observations in the final nodes of one-factor trees to escape overfitting at the starting point. The example of variable binning is provided in Figure 1.

As soon as we have transformed the factors, the individual Gini coefficients were calculated to assess predictive power of features. We excluded variables that have shown dramatic pairwise correlation and Gini drop on validation sample, so the rest were fed to logistic regression and the final model was fitted. The pipeline can be found in Bitbucket code repository [5].

Finally, we applied the Xgboost, Random Forest, Decision Tree and kNN algorithms to the same data to estimate the classification quality achievable with the "black-box" models as well.

As we can see, Xgboost performs better in terms of Gini. However, its results are not interpretable, and the best explanation for classification that we one can extract from the trained Xgboost model is the estimated feature importance, based on the number of times splits in trees that were done with each feature.

3.5 Random sampling alteration

In subsection 3.5, we study an alternative approach to generate α-weak hypotheses, which described in the previous section. Modification impacts the way we extract random subsamples from G_+ or G_-. In previous setting subsample of fixed size s is extracted. Thus, such classifiers have some fixed predictive power, and therefore, their effectiveness can be lower for some test objects and higher for others. To solve this problem, it is proposed to vary subsample size while the algorithm is running, and extract subsamples of different sizes in each iteration.

In this paper, two ways of specifying the size of subsample in each iteration were considered:

(1) Random choice of a number from the uniform distribution from 1 to N, where N is the size of the subsample.

(2) The choice of the number m from 1 to N with probability proportional to the number of combinations $\binom{N}{m}$.

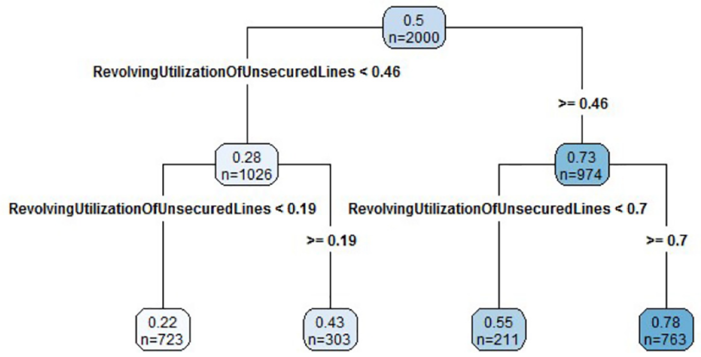

0.5
n=2000
RevolvingUtilizationOfUnsecuredLines < 0.46
>= 0.46
0.28
n=1026
0.73
n=974
RevolvingUtilizationOfUnsecuredLines < 0.19
>= 0.19
RevolvingUtilizationOfUnsecuredLines < 0.7
>= 0.7
0.22
n=723
0.43
n=303
0.55
n=211
0.78
n=763

Figure 1. One-factor trees for WOE-transformation of revolving utilization of unsecured lines

It is worth noting that the second method is equivalent to uniform sampling from the power set of the context. Thus, the second method provides convergence to classic lazy classification, as it considers descriptions of all possible combinations of objects (after a large number of iterations). Nevertheless, even with a small training sample size N, convergence will never be achieved, as the number of all subsets 2^N is too large. In practice, given large values of the subsample size m, the description of $\delta(g_n) \sqcap \delta(g_m)$ turns out to be too "general" and the proportion of objects of the opposite class in the set $(\delta(g_1) \sqcap \ldots \sqcap \delta(g_k) \sqcap \delta(g_m))^{\diamond}$ exceeds the given threshold α in each iteration. That is, for any value of the threshold α, it makes sense to extract such subsamples that its size does not exceed a certain maximum value of M. Thus, the two methods described above for setting the subsample size in each iteration can be slightly modified as follows:

(1) Random choice of a number from the uniform distribution from 1 to M, where M is the number calculated experimentally, depending on the data array used.

(2) Select a number m from 1 to M with a probability proportional to the number of combinations $\binom{N}{m}$.

Note that in the second method, probabilities of sampling a subsample of size k and $k+1$ are related as follows:

$$\frac{\binom{N}{k+1}}{\binom{N}{k}} = \frac{N!k!(N-k)!}{(k+1)!(N-k-1)!N!} = \frac{N-k}{k+1}.$$

It means that the sample of size $k+1$ will be extracted approximately $\frac{N-k}{k+1}$ times more frequently than the sample size k. It follows that given a large amount of data and a fixed number of iterations, small sample sizes is not used in the classification. The most probable size of the subsamples equals the value $\min\left(M, \left\lfloor \frac{N}{2} \right\rfloor\right)$. As a result of experiments with Kaggle data set, it was found that the optimal maximal subsample size is $M = 20$.

3.6 Visualization and interpretability of hypotheses

There is no unified definition for model interpretability in machine learning. However, there are some general requirements which are common among researchers. Some authors focus on the rule induction criterion (Feraud and Clerot, 2002), i.e. model is interpretable if it provides user with a set of simple rules to make decision.

Also, the causality is emphasized in (Miller, 2017) by stating that interpretability is the degree to which a human can understand the cause of a decision.

Further, one distinguishes between global and local interpretability in (Kim *et al.*, 2016). Global interpretability shows which features have major impact on prediction and also impact direction. Local interpretability answers the question why this particular test object received that particular prediction.

In this paper, we try to combine previously mentioned criteria and, to be more precise, we outline three properties of interpretability:

(1) Prediction is performed based on rules derived from initial factors preserving initial feature space.
(2) The algorithm processes initially defined target attribute.
(3) Prediction can be explained individually on test object level.

For example, decision trees and random forests are considered to be interpretable algorithms as soon as they process initial feature space and target attribute and produce rules which then are applied to test objects.

On the contrary, SVM with kernels does not satisfy first condition as soon as classification is performed in artificially constructed feature space. Also, XGBoost lacks the second property as soon as each next tree fits the errors of previous one, which is not the initially defined target attribute. Neural networks do not provide rules for a decision-maker. So, all these examples of algorithms cannot be interpreted.

The situation is different with query-based classification algorithm as soon as it works with hypotheses and, therefore, is an interpretable algorithm.

Hypotheses are mined in initial feature space and, in effect, they are just tupples of intervals. So, hypotheses define an area in initial feature space and serve as rules for a decision-maker. Therefore, premise can be visualized as a hypercube in a space of dimension *d*, where *d* is the number of intervals (and features). To visualize the premise, one can make the projection of this hypercube on the plane. As far as prediction explanation is concerned, each test object receives a number of rules (i.e. α-weak classifiers), which can be treated as portraits of good and bad borrowers. So, if the loan applicant is rejected it happens owing to the fact he or she is more similar to delinquent clients, i.e. more positive α-weak classifiers were found for the applicant.

Figure 2 shows two positive and two negative hypotheses on two features plane. Positive hypotheses are given in red, and the negative ones are given in blue. To construct each positive hypothesis, two objects from the positive context were randomly extracted. Then the meet-operator was applied and a set of intervals was obtained. After that only the intervals for two features were left. The same algorithm was run for negative hypotheses and negative context.

In Figure 3, there were ten positive and ten negative hypotheses built according to the same algorithm, whereas in Figure 4 their number reaches fifty. One can see that they are localized in different areas. As long as we extract more random hypotheses the boundary between good and bad regions becomes more and more obvious.

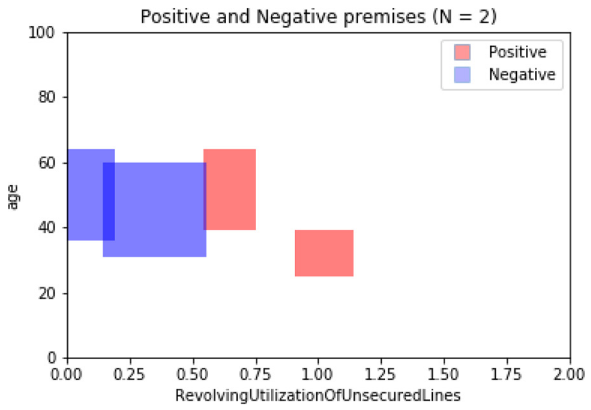


Figure 2.
Random positive and negative hypotheses

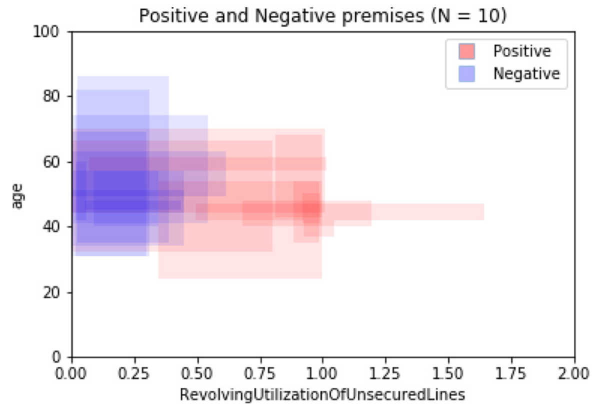


Figure 3.
Random positive and negative hypotheses

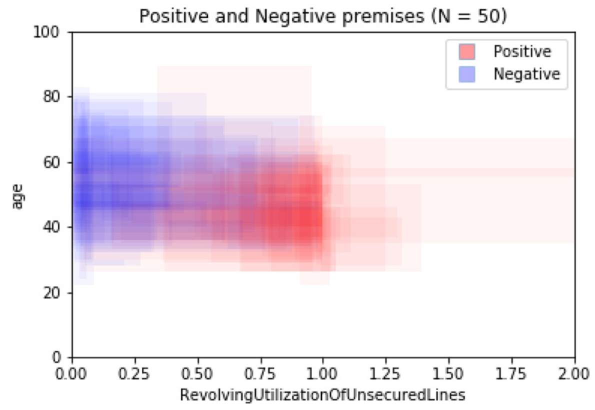


Figure 4.
Random positive and negative hypotheses

For 1,000 random hypotheses, the boundary is almost clear (Figure 5). As long as the number of extracted negative and positive hypotheses increases the number of multiple intersections between them grows. One can observe an expansion of the area with sparse positive hypotheses, while negative ones are fixed at interval from 0 to 1, positive intervals are in the range from 0.4 to 2.

It is interesting to realize that certain patterns can be extracted from the query-based classification algorithm (QBCA) model. We can observe rules such as if a loan applicant's age is greater than 50, and there was no delinquency in the past and the overall revolving utilization of unsecured lines was less than 11%, then the probability of default is almost four times lower than average. On the other side, applicants younger than 30 and having revolving utilization of unsecured lines greater than 72% will default 1.5 times more frequently than on average. This is where we enjoy the advantage of interval pattern structures: they represent the rules that can be easily interpreted, and at the same time they make prediction for each new object in validation data set individually, which allows to improve classification accuracy over the default scorecard model.

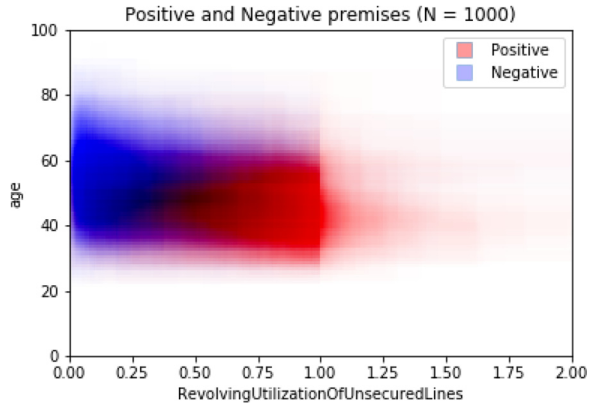


Figure 5.
Random positive and negative hypotheses

In addition, it is possible to see disputable areas (depicted in purple), that are areas of features values which are shared by both positive and negative hypotheses. In addition to this, one can see that for some hypotheses, the right border of the *RevolvingUtilizationOfUnsecuredLines* feature's interval is 1. But some hypotheses have a right-hand boundary of more than 1. Based on this one can make a conclusion about data errors or heterogeneity of the values of a given feature (the hypotheses were constructed on data without preprocessing). Thus, such visualization has an additional practical value.

4. Lattices of closed descriptions in regression problem

Classification is not the only problem that arises in credit scoring process. One has to predict other client parameters which can be continuously distributed. For example, when loans are granted online borrowers fill in their income amount in loan application form. It is necessary to verify those amounts by taking into account all other borrowers features such as education, employment experience, ownership, etc. Therefore, income prediction model is built to compare the predicted value and the one filled in the application. If the latter dramatically exceeds the value predicted by model then the warning alert can be sent indicating the borrower may have provided fake information.

In Section 4, we will adopt previously developed QBCA model to the case of continuous target variable (i.e. regression problem).

4.1 Augmented interval pattern structures

For the case where the target attribute is not a class label, but a continuous variable we adjust the definition of the interval pattern structure by equipping it with additional component h.

Let us define an *augmented interval pattern structure* as a quadruple (G, $\underline{D}$, δ, h), where the description d consists of two elements d_x and d_y (d_y is an interval for target attribute $y \in \mathrm{R}$ and d_x is a vector of intervals for explanatory attributes x which are supposed to predict the target attribute y), $\delta : G \rightarrow D$ and $h \in H$, where H is a family of density distribution functions for the target attribute y, i.e. $\int_{-\infty}^{+\infty} h(s)ds = 1$. We will also use notation δ_x and δ_y to distinguish between descriptions containing explanatory attributes and target attribute, respectively. The definition of the meet operation is left unchanged.

Suppose, we have an arbitrary set of objects $A_0 \subseteq G$, i.e. $A_0 = \{g_1, g_2, \ldots, g_J\}$, $\delta(g_j) = \{\delta_x, \delta_y\} = \{[x_{1j}; x_{1j}], \ldots, [x_{Mj}; x_{Mj}], [y_j; y_j]\}$, for $j = 1, \ldots, J$, where M is the number of explanatory attributes. Then we define the derivation operator in the following way

$$A_0^{\diamond} = (d_0, h_0)$$

where $d_0 = \{d_{x0}, d_{y0}\}$, and $d_{x0} = \delta_x(g_1) \sqcap \ldots \sqcap \delta_x(g_J)$ and target attribute description $d_{y0} = \delta_y(g_1) \sqcap \ldots \sqcap \delta_y(g_J)$, which is in fact a single interval $[y_{min}, y_{max}]$ and $h_0\colon d_{y0} \to [0;1]$. The h_0 is in effect a target attribute density distribution function based on observations of A_0, which we describe below. Let $\tau_0, \ldots, \tau_K$ be a partition of d_{y0} and $\tau_0 = y_{min}$, $\tau_K = y_{max}$ and $\Delta\tau_i = \frac{y_{max} - y_{min}}{K} = \tau_i - \tau_{i-1}, i = 1, \ldots, K$. Then:

$$h([\tau_{i-1}, \tau_i)) = \frac{|\{g \in G | [\tau_{i-1}, \tau_i) \sqsubseteq \delta_y(g)|}{|A|}, \forall i = 1, \ldots, K$$

Thus, h is a function of target attribute y values of objects in A. We will use the second derivation operator in a similar way it was used with interval pattern structures, however it will return the image for the description d_{x0} whatever target description d_{y0} and density function h are:

$$A_0^{\diamond\diamond} = (d_0, h_0)^{\diamond} = d_{x0}^{\diamond} = A_1$$

where $A_1 = \{g \subseteq G | d_{x0} \sqsubseteq \delta_x(g)\}$. Generally speaking, $A_0 \subseteq A_1$. Finally, $A_1^{\diamond} = (d_1, h_1)$. Note, that $d_1 = (d_{x0}, d_{y1})$, i.e. only target attribute description d_y is updated, so does h density function, while the explanatory variables description d_{x0} remains the same.

To approach target attribute prediction problem it will be useful to define *α-weak descriptions* by analogy with binary target case. An h-augmented interval pattern $d \in D$ is called an *α-weak hypothesis*:

$$1 - \frac{|\{g \in d_x^{\diamond} | d_y^{min} \leq \delta_y(g) \leq d_y^{max}\}|}{|d_x^{\diamond}|} \leq \alpha \ and \ \exists A \subseteq G : d_x = A^{\diamond}$$

An h-augmented interval pattern $d \in D$ is called an *α-weak regressor*:

$$1 - \frac{|\{g \in d_x^{\diamond} | d_y^{min} \leq \delta_y(g) \leq d_y^{max}\}|}{|d_x^{\diamond}|} \leq \alpha \ and \ \exists A \subseteq G : d_x \sqsubseteq A^{\diamond}$$

where $d = (d_x, d_y)$, d_y is a single interval $\left[d_y^{min}; d_y^{max}\right]$ for target attribute y, and h is a density function which reflects the distribution of target attribute within the interval d_y based on objects from A. This definition involves the parameter α that controls the frequency of hypothesis falsifications, i.e. how dramatically it is falsified.

To emphasize the connection between α-weak descriptions in classification and regression cases it's convenient to apply a small transformation to the definition above:

$$1 - \frac{|\{g \in G | d_y^{min} \leq \delta_y(g) \leq d_y^{max}\}|}{|A|} \leq \alpha \Longleftrightarrow$$

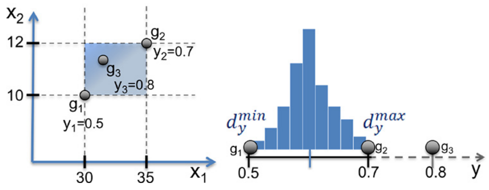
$$\Longleftrightarrow 1 - \frac{|\{g \in G | d_y \sqsubseteq \delta_y(g)\}|}{|A|} \leq \alpha \Longleftrightarrow \frac{|\{g \in G | d_y \not\sqsubseteq \delta_y(g)\}|}{|A|} \leq \alpha$$

The example below demonstrates the crucial differences between α-weak descriptions in cases of binary and continuous target variable.

Example 4.1 Consider a dataset provided in Table 3. Suppose $A_0 = \{g_1, g_2\}$. First, let's calculate $A_0^{\diamond}$.

$d_0 = (d_{0x}, d_{0y}) = ([30; 35], [10; 12], [0.5; 0.7])$, $h_0 = \{0.5, 0.7\}$,
thus $A_0^{\diamond} = (d_0, h_0)$.

Next let us find $A_0^{\diamond\diamond}$.
$A_0^{\diamond\diamond} = (d_0, h_0)^{\diamond} = d_{0x}^{\diamond} = A_1 = \{g_1, g_2, g_3\}$,
and hence $d_1 = ([30; 35], [10; 12], [0.5; 0.8])$, $h_1 = \{0.5, 0.7, 0.8\}$.
Finally, $A_0^{\diamond\diamond\diamond} = A_1^{\diamond} = (d_1, h_1)$.



Let d = ([30; 35], [10; 12], [0.5; 0.7]). As soon as g_3 has target attribute out of range [0.5;0.7], d is 2/3-weak regressor.

4.2 Query-based regression algorithm

Assume that we have a set of objects G and numerical data with a section of explanatory attributes $x_1, \ldots, x_M$ and continuous target attribute y. Now, suppose we receive a test object g_t with observable attributes x, but with unknown value of target attribute y. Next, we describe an approach to predict y using interval pattern structures.

The steps of query-based regression algorithm (QBRA) are similar to the case of classification. First, we mine α-weak hypotheses. Second, we calculate α-weak regressors by intersecting description $\delta(g_t)$ with the hypotheses. Third, we predict target attribute for test object g_t using mined α-weak regressors.

Let us start by choosing *subsample size* parameter which is the number of objects being randomly extracted from the set of objects G. Upon random extraction of objects $A_0 = \{g_1, \ldots g_K\}$ we calculate the following pattern $d_0 = \delta(g_1) \sqcap \ldots \sqcap \delta(g_K)$ and density distribution function h_0 for target attribute values. If d_0 is an α-weak hypothesis, then it is added to the collection of hypotheses that will be used for prediction later. Together with the pattern it is necessary to store the density function h_0.

On the second stage of out algorithm we derive α-weak regressors by calculating intersections $d_x \sqcap \delta(g_t)$ and obtain the new density functions h_1. Having finished with regressors mining, we move on to the next stage which is building up a prediction for the target attribute. In our case, the resulting prediction was defined by the mixture of distributions from all regressors. In practice, all target attribute values stored within regressors were put together

	x_1	x_2	y
g_1	30	10	0.5
g_2	35	12	0.7
g_3	31.5	11.5	0.8

Table 3. Toy data set for example 2

to form a final distribution. Finally, we tried both the average and the median of this distribution as the prediction for target attribute. Such approach takes into account *supports* of the regressors (the regressors supported by greater number of objects will contribute more). But which of h_0, h_1 or other we have to use?

Here we introduce another hyperparameter of the algorithm which is called "*capped*." Capped is a Boolean value, and if true, then the range for target attribute d_{y1} in $d_0^{\diamond\diamond}$ is truncated to d_{y0} and corresponding density function is h_1 calculated on the truncated set of target values. If capped hyperparameter is false, then we add d_{y1} and calculate the density function based on all target values that fell into d_{y1} based on objects from $d_0^{\diamond}$. The whole procedure is repeated so many times as controlled by the *number of iterations* parameter.

However, one can argue that regressors are different in sense of *anti-support* and *deviation* in target attribute values. Indeed, we would put more weight to the prediction based on regressors with narrow range of target attribute values. Therefore, we added target values to the final distribution with different weights, also both weighted average and weighted median were used as prediction.

We introduced two Boolean hyperparameters which control the scheme of assigning weights. The first one is *account for anti-support* and the second is *penalty for high deviation*. When *account for anti-support* is true, then the target values $\delta_y(g)$ of objects $g \in A$ with the regressor d are given weight according to the anti-support of that regressor:

$$w_a = 1 - \frac{|\{g \in d_x^{\diamond} | d_y^{min} \leq \delta_y(g) \leq d_y^{max}\}|}{|d_x^{\diamond}|}$$

When *penalty for high deviation* is true, then the weight is decreased with the higher deviation in the target attribute values:

$$w_p = \frac{1}{\sigma\left(\delta_y(g)\right)}$$

If the parameters values are false then the weights are equal to one. The final weight for the target attribute value of the object g, which will be contributed to aggregate distribution used for prediction, is defined as the product of the two weights:

$$w(g) = w_a \cdot w_p$$

Finally, suppose that P is a set of mined α-weak regressors. The prediction for target attribute y of a test object g_t can be based on weighted average:

$$\widehat{\delta_y(gt)} = \frac{\sum_{p \in P} \sum_{g \in A_p} \delta_y(g) \cdot w(g)}{\sum_{p \in P} \sum_{g \in A_p} w(g)}$$

or on the weighted median:

$$\widehat{\delta_y(gt)} = \underset{g \in \cup_p A_p}{\text{median}} \left(\bigcup_{p \in P} \bigcup_{g \in A_p} \left(\delta_y(g), w(g)\right)\right)$$

In case where P is an empty set, the prediction is average or median of all target attribute values in the knowledge base, i.e. the prediction is based on "naive" model.

4.3 Data and experiments

The data we used for the calculations represent verified borrower income information. We predict the income level using all other borrower features. Having an income prediction model one can apply it to the borrowers when there is no opportunity to verify income (e.g. in online lending). In case when the predicted value is dramatically lower than the value provided in a loan application form one can expect the borrower is trying to embellish his or her financial standing. Other use cases for income modeling include client base segmentation when different products are offered to customers based on their expected income level.

All three data sets Lending Club, Credit Scoring Catalonia and Give Me Some Credit contain income column which is a new target variable for QBRA.

The data was randomly divided into two parts with 70% of observations in one part and 30% in the other. The bigger part was used for training benchmarks and QBRA and 30% was used as a test set to evaluate predictions and their accuracy summarized in Table 4. The accuracy of predictions were evaluated in terms of mean absolute percentage deviation (MAPE) [6]:

$$MAPE = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

where y_i is a target attribute (recovery rate) for i-th client in the test set and $\hat{y}_i$ is prediction.

The accuracy of the algorithm was compared to benchmarks represented by random forests, as soon as their predictions are based on combination of simple rules, too. The proposed query-based regression algorithm showed comparable quality in the greater number of runs and in certain parameters area it outperformed random forests.

5. Conclusion

In this paper, we propose query-based classification approach to credit scoring problem. Model interpretability versus model accuracy is discussed and it is emphasized that model transparency is an important requirement in banking risk management. We define properties of interpretability and demonstrate that proposed algorithms satisfy them. We also address the continuous target variable prediction with query-based regression algorithm.

We argue that proposed algorithms allow user to extract interpretable rules for prediction and at the same time they outperform wide-spread methods in banking (e.g. scorecards, decision trees), however, black box models may have superior accuracy at the cost of interpretability.

Introduced algorithms are applied to open data on retail loan applicants and benchmark analysis is performed. Gini coefficient is used as a model quality metric. We performed grid search by running algorithms with different hyperparameter values.

Then, we describe a query-based regression algorithm and apply it to income prediction problem. The proposed algorithm shows comparable quality with benchmarks and in certain hyperparameters area it outperforms random forests. Mean absolute percentage error is used as a model quality metrics.

	Lending club loan data	Credit scoring Catalonia	Give me some credit
Random Forest	44.36	41.65	35.66
Decision Tree	45.36	43.33	38.17
kNN	49.22	44.89	42.81
XGBoost	42.84	40.65	35.88
QBRA	38.53	38.94	35.64

Table 4. Experiments results comparison (MAPE)

As an area for further research, one can consider and compare accuracy when other voting schemes are used. It is expected that taking into account α-weak classifiers specificity one can improve overall accuracy of the classification algorithm or, alternatively, one will reach the same accuracy given less number of iterations, which can improve time required for calculations. As for regression algorithm, one can consider keeping the density function h not only for target attribute in regressors, but also for explanatory attributes as well. It can be expected, that if α-weak regressors are mined based on some additional properties of features distribution, then they will be more relevant for test objects and will produce more accurate predictions for target attribute.

Acknowledgment
The work of Sergei O. Kuznetsov presented in Sections 1-2 was carried out at St Petersburg Department of Steklov Mathematical Institute of Russian Academy of Science and supported by the Russian Science Foundation grant no. 17-11-01276.

Notes

1. Available at: https://github.com/veegaaa/Interval-Pattern-Structures
2. Available at: www.lendingclub.com/info/download-data.action
3. Available at: https://github.com/gastonstat/CreditScoring
4. Available at: www.kaggle.com/c/GiveMeSomeCredit
5. Available at: https://bitbucket.org/Mosyamac/scorecards-in-r
6. Give Me Some Credit data set contains a significant number of observations with income close to zero, therefore, for this case we used MAE divided by sample average income value instead of classical MAPE.
7. Available at: https://github.com/veegaaa/Interval-Pattern-Structures

References

Aha, D.W. (Ed.) (1997),*Lazy Learning*, Kluwer Academic Publishers, Norwell, MA.

Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M. and Suykens, J. (2003), "Benchmarking state-of-the-art classification algorithms for credit scoring", *Journal of the Operational Research Society*, Vol. 54 No. 6, pp. 627-635.

bis.org (2020), available at: www.bis.org/publ/bcbsca.htm

documentation.statsoft.com (2020), available at: http://documentation.statsoft.com/STATISTICAHelp.aspx?path=WeightofEvidence/WeightofEvidenceWoEIntroductoryOverview

Feraud, R. and Clerot, F. (2002), "A methodology to explain neural network classification", *Neural Networks: the Official Journal of the International Neural Network Society*, Vol. 15, pp. 237-246, doi: 10.1016/S0893-6080(01)00127-7.

Ganter, B. and Kuznetsov, S. (2001), "Pattern structures and their projections", in Delugach, H. and Stumme, G. (Eds), "Conceptual structures: broadening the base", vol. 2120 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, pp. 129-142.

Ganter, B. and Wille, R. (1999), *Formal Concept Analysis: Mathematical Foundations*, Springer, Berlin.

Ghodselahi, A. (2011), "A hybrid support vector machine ensemble model for credit scoring", *International Journal of Computer Applications (0975-8887)*, Vol. 17 No. 5.

kaggle.com (2020), available at: www.kaggle.com/janiobachmann/lending-club-risk-analysis-and-metrics

Kaytoue, M., Codocedo, V., Buzmakov, A., Baixeries, J. and Kuznetsov, S.O. (2015), “Amedeo Napoli: pattern structures and concept lattices for data mining and knowledge processing”, ECML/PKDD (3), pp. 227-231.

tKaytoue, M., Kuznetsov, S.O., Napoli, A. and Duplessis, S. (2011), “Mining gene expression data with pattern structures in formal concept analysis”, *Information Sciences*, Vol. 181 No. 10, pp. 1989-2001.

Kim, B., Khanna, R. and Koyejo, O.O. (2016), “Examples are not enough, learn to criticize! Criticism for interpretability”, Advances in Neural Information Processing Systems.

Kuznetsov, S.O. (1999), “Learning of simple conceptual graphs from positive and negative examples”, LNCS (LNAI). 1704, pp. 384-391.

law.cornell (2020), available at: www.law.cornell.edu/uscode/text/15/1691

Lee, T.S., Chiu, C.-C., Chou, Y.-C. and Lu, C.-J. (2006), “Mining the customer credit using classification and regression tree and multivariate adaptive regression splines”, *Computational Statistics and Data Analysis*, Vol. 50 No. 4, pp. 1113-1130.

Masyutin, A. and Kuznetsov, S.O. (2016), *Continuous Target Variable Prediction with Augmented Interval Pattern Structures: A Lazy Algorithm*, CLA, pp. 273-284.

Masyutin, A., Kashnitsky, Y. and Kuznetsov, S.O. (2015), *Lazy Classication with Interval Pattern Structures: Application to Credit Scoring*, FCA4AI@IJCAI, pp. 43-54.

Meddouri, N., Khoufi, H. and Maddouri, M. (2014), “Parallel learning and classification for rules based on formal concepts”, *Procedia Computer Science*, Vol. 35, pp. 358-367.

Miller, T. (2017), “Explanation in artificial intelligence: insights from the social sciences”, arXiv Preprint, arXiv:1706.07269.

Nanni, L. and Lumini, A. (2009), “An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring”, *Expert Systems with Applications*, Vol. 36 No. 2, pp. 3028-3033.

SAS Institute Inc (2012), *Developing Credit Scorecards Using Credit Scoring for SAS Enterprise Miner 12.1*, SAS Institute Inc, Cary, NC.

Siddiqi, N. (2005), *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*, Wiley, ISBN: 978-0-471-75451-0.

stats.stackexchange.com (2020), available at: https://stats.stackexchange.com/questions/189568/replacing-variables-by-woe-weight-of-evidence-in-logistic-regression

Thomas, L., Edelman, D. and Crook, J. (2002), “Credit scoring and its applications”, *Monographs on Mathematical Modeling and Computation*, SIAM, Pliladelphia, pp. 107-117.

triamus.github.io (2020), available at: https://triamus.github.io/project/lending-club-loan-data-in-r/#defining-default

Veloso, A. and Wagner, M. Jr. (2011), *Demand-Driven Associative Classification*, Springer.

Veloso, A., Wagner, M. Jr and Zaki, M.J. (2006), “Lazy associative classification”, ICDM, IEEE Computer Society, pp. 645-654.

Yap, B.W., Ong, S.H. and Husain, N.H.M. (2011), “Using data mining to improve assessment of credit worthiness via credit scoring models”, *Expert Systems with Applications*, Vol. 38 No. 10, pp. 13274-13283.

Zhou, L. and Wang, H. (2012), “Loan default prediction on large imbalanced data using random forests”.

Appendix

Query-based algorithms described above have been implemented in Python and tested on three credit scoring datasets. The source code is available in the public Github repository [7]. It allows one to reproduce results derived in this paper as well as perform experiments with any custom dataset.

In the repository we provide separate Jupyter-notebooks for every dataset and algorithm considered in this paper. Those notebooks are fully executable so any user is able to run all the cells sequentially. We also provide this section with hyperparameters grid search for classification and regression versions of algorithms. In addition, we present the set of hyperparameters that delivers the best model quality metric (Gini for classification and MAPE for regression).

We also provide the pseudo-code of query-based classification and regression algorithms so the reader grasps the idea behind them.

Classification algorithm

```
Algorithm 1 Mining step (case of positive $\alpha$ - weak hypotheses)
    Input: $\{G_+, G_-\}$ - positive and negative numerical contexts.
    Output: $H^+$ and $H^-$ are lists for positive and negative $\alpha$-weak
            hypotheses.
1:  for $num.iter$ times do
2:      $\{g_1, \ldots, g_s\}$ is a random sample from $G_+$ of size $sub.smpl$
3:      $d = \delta(g_1) \sqcap \ldots \sqcap \delta(g_s)$
4:      $d_-^\diamond = \{g \in d^\diamond | g \in G_-\}$
5:      if $|d_-^\diamond| \leq \alpha \cdot |d^\diamond|$ then
6:          Add $d$ to $H^+$
```

```
Algorithm 2 Classification and voting step
    Input: $\{G\ ,\ G\ \}$ - positive and negative numerical contexts.
    $g$ - test$^{+}$obj$^{-}$ect. $H_+$ and $H_-$ are lists for positive and negative
    $\alpha^{t}$-weak hypotheses.
    A(·), $w$(·), $\otimes$ - voting scheme.
    Output: $margin(g)$ - measure that is produced by the voting rule.
1: $P = A_i^p\left(w\left(h_i^+ \sqcap \delta(g_t)\right)\right)^t$
2: $N = A_j^n\left(w\left(h_j^- \sqcap \delta(g_t)\right)\right)$
3: $margin(g_t) = P \otimes N$
```

Regression algorithm

```
Algorithm 3 Mining hypotheses)
Input: - numerical context.
Output: $H$ is a list of $\alpha$-weak hypotheses.
1:      for $num.iter$ times do
2:      $\{g_1, \ldots, g_s\}$ is a random sample from $G$ of size $sub.smpl$
3:      $d_x = \delta_x(g_1) \sqcap \ldots \sqcap \delta_x(g_s)$
4:      $A = d_x^\diamond$
5:      if $1 - \frac{|\{g \in A | d_y^{min} \leq \delta_y(g) \leq d_y^{max}\}|}{|A|} \leq \alpha$ then
6:          $d = (d_x, d_y)$
7:          Add $d$ to $H$
```

```
Algorithm 4 Regression step
    Input: $G$ - numerical context.
    $g_t$ - test object. $H$ is a list of $\alpha$-weak hypotheses.
    Output: $\widehat{\delta_y(gt)}$ - predicted target value.
1: for hypothesis $d$ in $H$ do
2:      $d_{x,t} = d_x \sqcap \delta_x(g_t)$
3:      $w_a = 1 - \frac{|\{g \in d_x^\diamond | d_y^{min} \leq \delta_y(g) \leq d_y^{max}\}|}{|d_{x,t}^\diamond|}$
4:      $w_p = \frac{1}{\sigma(h_y)}$
```

```
5:      $w_h = w_a \cdot w_p$
6:      add $w^h$ to W
7:      add $\sum_{g \in d_x^\diamond} \delta_y(g) \cdot w_h$ to S
8:      $\widehat{\delta_y(gt)} = \frac{\sum_{s \in S} s}{\sum_{w \in W} w}$
```

All hyperparameters were tuned through the grid search. subsample size took values from 1 to 20, number of iterations: 10,000, 50,000, 100,000, 200,000, 300,000, 400,000. Alpha threshold – from 0 to 1 with 0.01 step. The best combinations of hyperparameters are shown in Tables A1 to A3.

Table A1. Best hyperparameters sets in classification problem (positive hypotheses)

	Num.iter	*Subs.size*	α
Give me some credit	400,000	3	0.4
Lending club loan data	300,000	2	0.9
Credit scoring Catalonia	200,000	2	0.12

Table A2. Best hyperparameters sets in classification problem (negative hypotheses)

	Num.iter	*Subs.size*	α
Give me some credit	400,000	5	0.9
Lending club loan data	300,000	5	0.95
Credit scoring Catalonia	200,000	3	0.57

Table A3. The best hyperparameters sets in regression problem

	Num.iter	*Subs.size*	α
Give me some credit	200,000	2	0.33
Lending club loan data	200,000	2	0.76
Credit scoring Catalonia	100,000	2	0.41

Corresponding author
Sergei O. Kuznetsov can be contacted at: skuznetsov@yandex.ru