N. Neuhaus, B. Kovalerchuk, Interpretable Machine Learning with Boosting by Boolean Algorithm, Joint 2019 8th Intern. Conf. on Informatics, Electronics & Vision (ICIEV) & 3rd Intern. Conf. on Imaging, Vision & Pattern Recognition (IVPR), Spokane, WA, 2019, 307-311.

Interpretable Machine Learning with Boosting by Boolean Algorithm

Nathan Neuhaus
Dept. of Computer Science
Central Washington University
USA
Nathan.Neuhaus@cwu.edu

Boris Kovalerchuk
Dept. of Computer Science
Central Washington University
USA
Boris.Kovalerchuk@cwu.edu

***Abstract*— Much research has been conducted for developing efficient, automated, interpretable, and highly accurate Machine Learning algorithms. However, the primary challenge remains -- developing algorithms successful in all of these aspects simultaneously. For applications, where human lives are at stake, having to choose between accuracy and interpretability is not acceptable. To address this challenge, previously the Dominance Classifier and Predictor (DCP) algorithm, capable of discovering of the human-understandable models in simple and visualizable terms was proposed. On the benchmark Wisconsin Breast Cancer (WBC) dataset, it achieved greater accuracy than other interpretable algorithms, reducing the gap between the prediction accuracy of interpretable and non-interpretable algorithms on these data. This paper proposes a new interpretable algorithm RPPR, to bridge this gap more by boosting DCP via discovering properties of misclassified cases. Experiments with RPPR on WBC and two other benchmark datasets using 10-fold cross validation, achieved accuracies greater than 99%. Thus, the accuracy of non-interpretable algorithms is reachable without sacrificing interpretability.**

Keywords— machine learning, interpretability, boosting, classifier, dominant intervals, human-understandable models.

I. Introduction

The quest for interpretable models is growing [11-14] and multiple approaches were proposed to generate interpretable machine learning models [6, 7, 13-18]. The approach that we present in this paper is based on boosting the DCP interpretable algorithm [6] with another interpretable algorithm by discovering the relations between attributes, then learning from the said relations to override/reverse the inaccurate predictions of the first.

A new algorithm called **Reverse Prediction Pattern Recognition (RPPR)** boosts DCP by discovering pair relations between attributes, then learning from said relations to override inaccurate DCP predictions. By boosting DCP with RPPR, we show on the three benchmark datasets using 10-fold cross validation that it is possible to achieve accuracy over 99%, reaching the accuracy of non-interpretable algorithms and beyond. *As far as we know* for the pertinent benchmark datasets, for the first time it is possible to have both accuracy and interpretability, rather than having to choose one at the expense of the other.

A. Steps of DCP Algorithm

Below we present the necessary background information -- main steps of the DCP algorithm from [6]. The **first step** of DCP is to produce the dominance classifier structure $S = \langle \{V_i, h_{1i}, h_{2i}, \ldots, h_{ki},\} \rangle$, essentially a table containing intervals $\{V_i\}$ and the number of cases $h_{1i}, h_{2i}, \ldots, h_{ki}$, of each class on the training data within the respective interval on each predictor attribute X_i.

For example, in Table 1, the interval [0.1,0.3] on the predictor attribute X_i contains 10 cases of class 1 and 200 cases of class 2 while the interval [0.4, 0.5] on the same X_i contains 20 cases of class 1 and 10 cases of class 2.

TABLE 1. EXAMPLE: INTERVALS OF DISTRIBUTION OF CASES OF CLASSES IN AN ATTRIBUTE.

Interval in attribute X_i	Number of cases of Class 1, h_{1i}	Number of cases of Class 2, h_{2i}
[0.1,0.3]	10	200
[0.4, 0.5]	20	10

There are three sub-steps in the step 1.

Step 1.1 *Sorting* each predictor attribute X_i by ascending value, then *removing* all duplicate values by grouping values. For instance, values 0.4, 0.4, 0.4, 0.1, 0.1, 0.9, 0.5 on the attribute X_i are ordered 0.1, 0.1, 0.4, 0.4, 0.4, 0.5, 0.9 and then grouped to get 0.1, 0.4, 0.5, 0.9 without duplicates.

Step 1.2: *Computing* the number of times each value x appears by class j for each predictor attribute X_i, e.g., we can get a pair (1,1) -- one case of class 1 and one case of class 2 for value x_i=0.1.

Step 1.3 Using the information from steps 1.1 and 1.2 for each predictor attribute to *calculate the dominant class* for each point, then *to group* all points into *intervals*, which are contiguous, dominated by the same class, and have a length greater than or equal to some *length threshold T*. For instance, we can get three intervals with counts for classes: ([0.1,0.1],(1,1)), ([0.4,0,5],(0,4)), and ([0.9,0,9],(1,0)). Here interval [0.4,0,5] contains 4 cases of class 2 with full dominance

of class 2. For the length threshold T=0.08 this interval is a dominance interval on X_i.

The **step 2** of the DCP algorithm includes two sub-steps.

Step 2.1: Computing the *dominant class* in each attribute for a given case **x**. First, a search is conducted to ascertain whether an interval containing the point x_i exists in the attribute X_i. Otherwise, no prediction is associated with the given attribute. The *dominance level* is the ratio of total number of points for the given class to the total number of points of all the classes.

Step 2.2: Voting to *combine dominances for class prediction*, based on individual attributes, to determine the total prediction. For example, for the input n-D point **x** with x_i=0.25 the use of Table 1 would activate interval [0.1,0.3], which in turn would predict class 2 over class 1 due to dominance on this interval with a much greater number of instances of class 2. This is a simple single-attribute **prediction rule R0**:

$$\text{If } \mathbf{x}=(x_1,x_2,\ldots,x_i,\ldots,x_n) \;\&\; x_i \in V \;\&\; h_j(V) = max(h_1(V),h_2(V),\ldots,h_k(V)) \text{ then } \mathbf{x} \in \text{class } j \quad (1)$$

where $h_1(V)$, $h_2(V)$,…,$h_k(V)$ are the numbers of cases of classes 1,2,…,k, respectively, that have values in interval V on attribute X_i.

To define a rule for n attributes we use *notation* $j(\mathbf{x},i)$ to denote that class j is a dominant class for the n-D case **x** based on the attribute X_i.

B. DCP Voting Methods

The voting methods differ in the number of votes assigned to each dominant class. In a simple base method for n attributes, denoted as **VM,** each dominant interval gets one vote, V_{ij}=1, if j is a dominant class on X_i, else V_{ij}=0. In VM the *total vote for class j* is a sum of votes for this class V_{ij} for all the attributes X_i, it is the number of times when $d(\mathbf{x},i)=j$ by considering all the attributes X_i,

$$V^{(j)} = \sum_{i=1}^{n} V_{ij} = |\{d(\mathbf{x},i)=j: i=1,n\}| \quad (2)$$

The classification decision is based on the comparison of total votes and finding their max value,

$$V^{(t)}(\mathbf{x}) = max\,(V^{(1)}(\mathbf{x}), V^{(2)}(\mathbf{x}), \ldots, V^{(k)}(\mathbf{x})) \quad (3)$$

with the **classification rule, R1:**

$$\text{If } V^{(t)}(\mathbf{x})=max\,(V^{(1)}(\mathbf{x}), V^{(2)}(\mathbf{x}), \ldots, V^{(k)}(\mathbf{x})) \text{ then } \mathbf{x} \text{ belongs to class } t \quad (4)$$

For each class j rule R1 checks that a given dimension's value x_i was classified to that class j (belongs to the interval that is dominated by class j), and if so, adds a single class-vote to class j votes, then finds the max of these sums in formula (4).

II. REVERSE PREDICTION PATTERN RECOGNITION ALGORITHM

As described above, DCP finds patterns in the form of dominant intervals in individual attributes. To enhance DCP, we look for patterns in the pairs of attributes that are specific solely to DCP false-positive or DCP false-negative cases.

Further enhancement of DCP is possible via finding patterns that are more complex such as ones that involve three or more attributes.

A. Basis schema of the RPPR Algorithm

The basis schema of the RPPR algorithm consists of the following steps:

1) Encoding elements of DCP algorithm as Boolean vectors,

2) Finding training cases *misclassified* by the DCP,

3) Discovering all the *unique* pairs for DCP False-Negative (FN) and DCP False-Positive (FP) n-D points on training data,

4) Finding FN and FP n-D points in the *validation/test*ing dataset with these unique pairs, and

5) Reversing prediction for these n-D points.

The first step of presenting elements of DCP algorithm as Boolean vectors is as follows. Consider *n-D training* dataset with attributes $X_1,X_2,\ldots,X_n$.

Let $V_1,V_2,\ldots,V_n$ be the dominance intervals for class 1 found by the DCP algorithm on these training data. Then, the algorithm produces n Boolean attributes $B_1,B_2,\ldots,B_n$ and n-D *Boolean point* **b**=$(b_1,b_2,\ldots,b_n)$ in these attributes from n-D point **x**=$(x_1,x_2,\ldots,x_n)$ as follows:

$$b_i=1 \Leftrightarrow x_i \in V_i,$$

i.e., x_i belongs to the dominance interval of class 1 on attribute U_i. Fig. 1 illustrates it with several 5-D Boolean points.

	B_1	B_2	B_3	B_4	B_5
b_1	0	1	1	1	0
b_2	1	1	0	0	0
b_3	0	0	1	0	0
b_4	1	0	0	1	1
b_5	1	0	0	0	1

Fig. 1. Binary n-D points produced from DCP algorithm.

The step 2 produces the input for the third step -- DCP false-negative and DCP false-positive n-D points in the binary form by comparing the DCP predicted value to the Target value on training data. See Fig. 2 that illustrates this step.

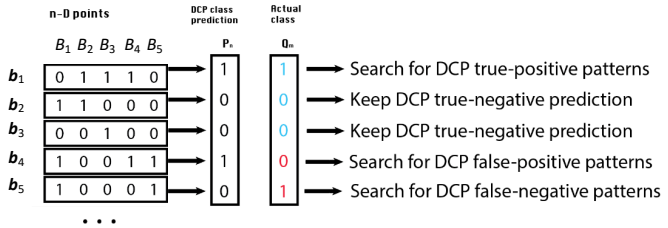

Fig. 2. Analysis of DCP algorithm performance to find misclassified training cases.

On the step 3, the algorithm discovers unique pairs for DCP false-negative and false-positive *n*-D points found in step 2:

3.1) Collect:

- all pair-combinations (b_i,b_j) for false-positive n-D points in the array A, denoted as bin-A;
- all pair-combinations (b_i,b_j) for false-negative n-D points in the array B, denoted as bin-B;
- all pair-combinations (b_i,b_j) for true-positive n-D points in array C, denoted as bin-C; and
- all pair-combinations (b_i,b_j) of true-negative n-D points in the array D, denoted as bin-D;

3.2) Remove all pair-combinations from bin-A which also exist in bin-C;

3.3) Remove all pair-combinations from bin-B, which also exist in bin-D.

B. Example

Fig. 3 illustrates finding the set of all pair-combinations (b_i,b_j) in n-D point $\mathbf{b}_5$ from Fig. 1 and Fig. 2 that have *incorrect* DCP predictions equal to 0. The table on the left shows all pairs from $\mathbf{b}_5$, starting from the pair $(b_1,b_2) = (1,0)$ on the top and ending with the pair $(b_4,b_5) = (0,1)$ on the bottom. We call pairs in this set, *candidate-pairs*. Notice *n*-D point $\mathbf{b}_5$ is a DCP false-negative *n*-D point, therefore the algorithm searches for the pairs (x_i,x_j) that can be used to alter the DCP prediction 0 for $\mathbf{b}_5$.

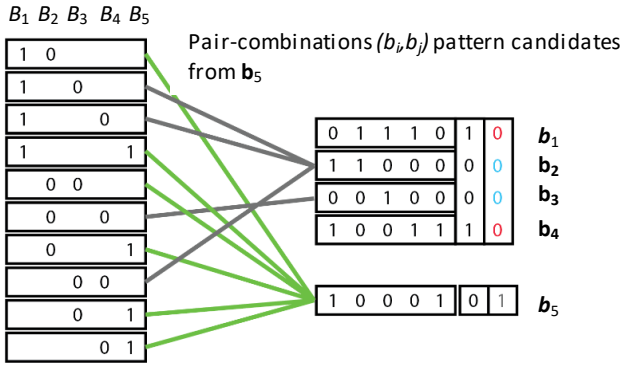

Fig. 3. Examples of unique and non-unique pair patterns.

In Fig. 3, green lines connect (b_i,b_j) pairs on the left, which are solely present in DCP false-negative *n*-D points $\mathbf{b}_1$-$\mathbf{b}_5$, to uniquely false-negative *n*-D points. The remaining pairs that present in both false-positive and true-positive *n*-D points are marked by the grey lines connecting them to respective non false-positive n-D points.

This process of discovering unique pairs is repeated for all $\mathbf{b}_i$. In this way, the RPPR algorithm produces unique pairs for DCP false-negative n-D points, found on training data. Unique pairs for false-positive n-D points are produced in a similar process. The step 4 uses these unique pairs to find the FN and FP n-D points in the validation set, and reverse them on step 5.

It is possible that the training data could be too small-- do not have enough n-D points to reverse DCP false prediction. In such a case, the RPPR result will be erroneous. The remedy for this situation requires simply increasing the amount of training data, if possible.

III. RESULTS OF EXPERIMENTS

The experiments have been conducted on three datasets from the University of California at Irvine machine learning repository [8], normalized to [0,1] interval:

- The Wisconsin Breast Cancer dataset (WBCD-9) that consists of 688 samples, and 9 features. The classes are: benign and malignant, with 444 and 239 samples, respectively.
- The Heart Disease dataset that consists of 303 samples, and 12 features. The classes are: "has heart disease" and "does not have heart disease".
- The Wisconsin Breast Cancer dataset with 32 attributes (WBCD-32) that consists of 569 samples, and 32 features. The classes are benign and malignant.

Table 2 shows 10-fold cross validation accuracies with DCP boosted by RPPR on the Wisconsin Breast Cancer with nine features, the Heart Disease Dataset with 14 dimensions, and the Wisconsin Breast Cancer with 32-features. For these datasets, the average accuracy is above 99% for each of them with 100% reached for the third dataset.

For the WBCD dataset 1, the best accuracy reported for the SVM in [10] is 96.995% with the 10-fold cross-validation tests. Other results are 96.84% [2] and 96.99% [4] for the SVM, and 97.28% [10] by combing SVM, C4.5 decision tree, naïve Bayesian classifier, and the k-Nearest Neighbors algorithms.

TABLE 2.TEN-FOLD CROSS VALIDATION ACCURACIES WITH DCP BOOSTED BY RPPR.

	Breast cancer dataset 1: 9-D	**Heart Disease dataset: 14-D**	**Breast cancer dataset 2: 32-D**
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	0.99	1.00	1.00
4	0.97	1.00	1.00
5	1.00	1.00	1.00
6	0.99	1.00	1.00
7	1.00	1.00	1.00
8	1.00	0.99	1.00
9	1..00	1.00	1.00
10	1.00	1.00	1.00
Average	0.993	0.999	1.00

Our results on this dataset range from 97 to 100% with average of 99.3% in 10-fold cross validation, which exceeds other known published results. In addition, in contrast with SVM, it is *visual*, has clear *interpretation* and is *explainable to domain experts* which is essential in domains with the characteristic of high cost errors, where model explanation is mandatory.

For the Heart Disease (dataset 2), the accuracies reported in [3] are 77.56% for C4.5 decision tree, 83.5% for Naïve Bayes, and 84.12% for SVM with the 10-fold cross-validation tests, while our experiments resulted in 99.9% accuracy.

For the breast cancer dataset 2 with 32 attributes, the accuracies reported in [1] are 97.2% for particle swarm optimization, 96.6% for genetic algorithms, and 97.3 for ANN. These results averaged 30 runs with 80% of data allocated to the training set and the remaining 20% is allocated to the test/validation set. Our 10-fold cross validation experiments resulted in 100% accuracy.

Fig. 4 and Table 3 summarize all of these comparisons.

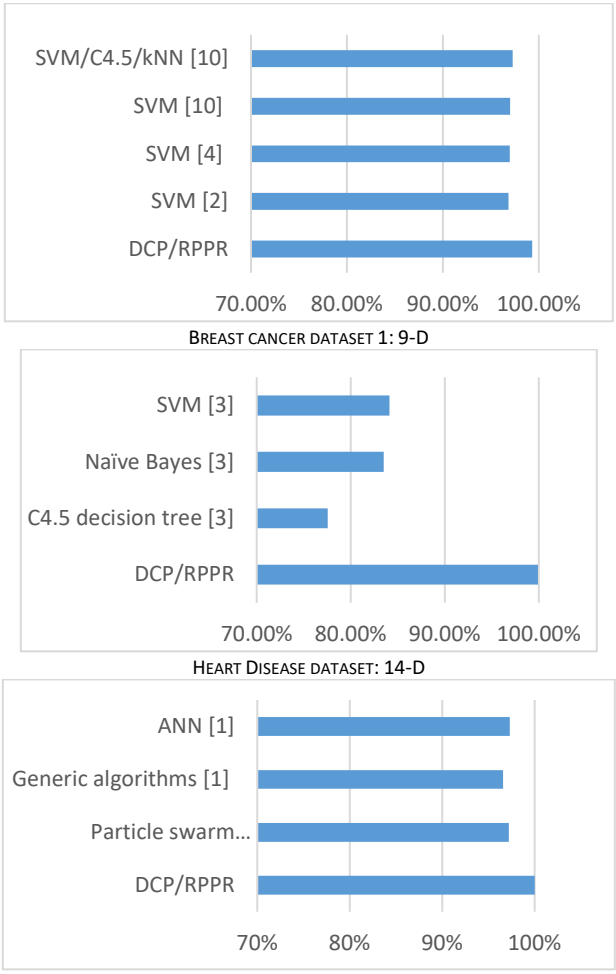

Fig. 4.comparions of ten-fold cross validation accuracies for DCP/RPPR and other algorithms.

TABLE 3. COMPARISONS OF TEN-FOLD CROSS VALIDATION SCCURACIES.

Breast cancer dataset 1: 9-D	
DCP/RPPR	**99.3%**
SVM [2]	96.84%
SVM [4]	96.99%
SVM [10]	96.995%
SVM/C4.5/kNN/Bayesian [10]	**97.28%**
Heart Disease dataset: 14-D	
DCP/RPPR	***99.9%***
C4.5 decision tree [3]	77.56%
Naïve Bayes [3]	83.5%
SVM [3]	**84.12%**
Breast cancer dataset 2: 32-D	
DCP/RPPR	**100%**
Particle swarm optimization [1]	97.2%
Generic algorithms [1]	96.6%
ANN [1]	**97.3%**

IV. COMPARISON WITH THE BOOSTING META-ALGORITHMS

A. *Boosting Meta-Algorithms*

Boosting algorithms demonstrated their efficiency in many tasks [9,19]. In adaptive boosting (AdaBoost) meta-algorithm and related methods, the boosted classifier is a linear combination of the weak classifiers h_i [9] of the form

$$C(x_i) = \alpha_1 h_1(x_i) + \alpha_2 h_2(x_i) + \ldots \alpha_1 h_m(x_i).$$

Thus, boosting is a form of linear regression. The booster learner's job in the two-class classification task [5] is to find *weak hypotheses* h_t: $X \rightarrow \{-1,1\}$, and the boosted classifier $C(x_i)$ with the output final hypothesis:

$$H(x_i) = sign\big(C(x_i)\big).$$

B. *Comparison*

Boosting commonly assumes that weak classifiers are classifiers of the same type such as decision trees. While decision trees are interpretable, the interpretation of the boosted $C(x_i)$ and $H(x_i)$ is more difficult, especially if the number of voting trees is large.

To avoid this difficulty, the RPPR approach does not create a weighed sum of weak classifiers of a given type, but builds a classifier of another type on the top of DCP classifier distinguishing it from the other boosting approaches.

The similarity to the other boosting approaches is the focus on misclassified cases, to improve the accuracy of the classifier. RPPR is applied to the cases that are poorly classified by DCP.

Typically, other boosting approaches first assign equal weight values ($1/n$) to each case for computing the total prediction error. Subsequently, it then attributes lesser weight to cases with inaccurate predictions, which are then carried over for retraining the classifier on the later boosting iterations. This forces the classifier to learn the cases, which it failed in the previous iterations.

In the weight terms, RPPR sets up only two weights: 0 for cases correctly classified by the DCP algorithm and 1 to cases, which the DCP incorrectly classified. RPPR is trained to improve the accuracy on the later cases of false predictions.

This ensures that the accuracy on the former cases will not be deteriorated by RPPR, which is a major potential drawback of boosting.

Next, RPPR operates at a more fine-grained resolution by considering the interactions between the attributes within each case, instead of prioritizing each misclassified case entirely. Computationally RPPR is also more efficient needing only a single pass on the training set, instead of the successive rounds.

Thus, conceptually, RPPR boosts DCP, but in a very different way in comparison with AdaBoost and similar

boosting methods. RPPR has reached the level of accuracy of uninterpretable algorithms in the experiments described above, while being fully interpretable.

V. CONCLUSION

In the previous experiments [6], the DCP algorithm was more accurate than other interpretable algorithms on the benchmark dataset, but less accurate than non-interpretable Neural Networks. This paper has shown that, when DCP algorithm is boosted by the RPPR algorithm, the average accuracy achieved using 10-fold cross validation, is greater than 99.8%, on three benchmark datasets, thus reaching and exceeding the accuracy of the non-interpretable algorithms.

In addition to the greater accuracy, RPPR algorithm simplifies and boosts the DCP algorithm via the discovery of pair relations between attributes, then learns from these relations, to reverse the false DCP predictions, rather than the previous approach of using the complicated voting schemes. In summary, DCP and RPPR are interpretable and have advantages and disadvantages, relative to the algorithms such as Decision Trees and Naïve Bayes, howe ver they can be combined with other methods to mitigate these disadvantages, and can be developed further to incorporate more complex relations, beyond the pair relations. This paper had shown that it is possible to have both high accuracy, and full interpretability, without the need to choose one at the expense of the other.

REFERENCES

[1] Aalaei S, Shahraki H, Rowhanimanesh A, Eslami S. Feature selection using genetic algorithm for breast cancer diagnosis: experiment on three different datasets. Iranian journal of basic medical sciences. 2016;19(5): 476.

[2] Aruna, S., Rajagopalan, D.S., Nandakishore, L.V. Knowledge based analysis of various statistical tools in detecting breast cancer. Comput. Sci. Inf. Technol. 2011, 2, 37–45.

[3] Chaki D, Das A, Zaber MI. A comparison of three discrete methods for classification of heart disease data. Bangladesh Journal of Scientific and Industrial Research. 2015 Dec 11;50(4):293-6.

[4] Christobel, A., Sivaprakasam, Y. An empirical comparison of data mining classification methods. Int. J. Comput. Inf. Syst. 2011, 3, 24–28.

[5] Freund Y, Schapire R, Abe N. A short introduction to boosting. Journal-Japanese Society for Artificial Intelligence. 1999, 1;14(771-780): 1612.

[6] Kovalerchuk B., Neuhaus, N. Toward Efficient Automation of Interpretable Machine Learning. In: 2018 IEEE International Conference on Big Data, pp. 4933-4940, 978-1-5386-5035-6/18, Seattle, Dec. 10-13, 2018 IEEE.

[7] Kovalerchuk B. Visual Knowledge Discovery and Machine Learning, 2018, Springer.

[8] Lichman, M. UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, 2013.

[9] Rojas, R. 2009. AdaBoost and the super bowl of classifiers a tutorial introduction to adaptive boosting. Freie University, Berlin, Tech. Rep. http://www.inf.fu-berlin.de/inst/ag-ki/adaboost4.pdf

[10] Salama, G.I.,Abdelhalim, M., Zeid, M.A. Breast cancer diagnosis on three different datasets using multi-classifiers. International Journal of Computer and Information Technology, 1(1), 2012, 36-43.

[11] DARPA, Explainable Artificial Intelligence (XAI), DARPA-BAA-16-53, 2016

[12] General Data Protection Regulation, Article 22. Parliament and Council of the European Union. 2016.

[13] Lipton Z. The Mythos of Model Interpretability, Communications of the ACM, 2108, 61,36-43.

[14] Lundberg SM, Nair B, Vavilala MS, et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. Nature Biomedical Engineering. 2018 Oct;2(10):749.

[15] Schüller P., Kazmi M. Best-Effort Inductive Logic Programming via Fine-grained Cost-based Hypothesis Generation. arXiv preprint arXiv:1707.02729. 2017 Jul.

[16] Speichert S., Belle V. Learning Probabilistic Logic Programs in Continuous Domains. arXiv preprint arXiv:1807.05527. 2018 Jul 15.

[17] Kovalerchuk B., Vityaev E., Data Mining in Finance: Advances in Relational and Hybrid Methods 2000, Kluwer/Springer.

[18] Lakkaraju H, Kamar E, Caruana R, Leskovec J. Faithful and Customizable Explanations of Black Box Models. 2019, AAAAI/ACM Conference on Artificial Intelligence, Ethics and Society, http://www.aies-conference.com/wp-content/papers/main/AIES-19_paper_143.pdf

[19] Chen T, Guestrin C. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining 2016 Aug 13 (pp. 785-794). ACM.