Knowledge and Information Systems (2023) 65:1657–1683
https://doi.org/10.1007/s10115-022-01786-2

REGULAR PAPER

Hypergraph-based importance assessment for binary classification data

Pawel Misiorek[1] · **Szymon Janowski**[1]

Received: 25 July 2022 / Revised: 28 September 2022 / Accepted: 31 October 2022 /
Published online: 25 December 2022

Abstract
We present a novel hypergraph-based framework enabling an assessment of the importance of binary classification data elements. Specifically, we apply the hypergraph model to rate data samples' and categorical feature values' relevance to classification labels. The proposed Hypergraph-based Importance ratings are theoretically grounded on the hypergraph cut conductance minimization concept. As a result of using hypergraph representation, which is a lossless representation from the perspective of higher-order relationships in data, our approach allows for more precise exploitation of the information on feature and sample coincidences. The solution was tested using two scenarios: undersampling for imbalanced classification data and feature selection. The experimentation results have proven the good quality of the new approach when compared with other state-of-the-art and baseline methods for both scenarios measured using the average precision evaluation metric.

Keywords Hypergraphs · Machine learning · Imbalanced data · Random undersampling · Feature selection

1 Introduction

In this paper, we investigate the area of research on data modeling for the binary classification problem. Specifically, we focus on the application of a hypergraph, which is regarded as a convenient and expressive tool to model and visualize the coincidences between data elements in complex structures [1, 2]. A hypergraph is a generalization of a graph, allowing edges that may consist of any number of vertices. Although they are less prevalent than standard graphs, hypergraphs have already been successfully used in various machine learning tasks, including community detection and clustering [3, 4], biomedical applications such as modeling

Szymon Janowski have contributed equally to this work.

✉ Pawel Misiorek
pawel.misiorek@put.poznan.pl

Szymon Janowski
szy.janowski@gmail.com

[1] Institute of Computing Science, Poznan University of Technology, Piotrowo 3, 60-965 Poznan, Poland

virus spreading networks, relations between genes and diseases [1] or finding genes which are central in the host response to viral infection [2], as well as e-commerce applications, mainly aimed at prediction [5]. In contrast to the most common machine learning applications of hypergraphs based on modeling structure forming networks, this paper investigates the application domain related to processing binary classification datasets. Specifically, we study how to model a binary classification dataset in a way allowing us to assess the importance of each input data element. We investigate two research scenarios in which we apply the hypergraph-based importance rates and evaluate their quality: dealing with imbalanced data [6, 7] and feature selection [8–10]. Both problems are regarded as highly practical and are currently intensively investigated in the area of research on machine learning [8, 11].

Many state-of-the-art classification algorithms, e.g., those based on neural networks [12, 13], are not able to provide a human-interpretable explanation, allowing users to understand how the values of original data elements influence the algorithm decision. This explanation capability is even more critical when studying the fairness of machine learning solutions [14], which is a crucial issue in every area where automated decision-making algorithms should give users the decision explanation to avoid discrimination. Graphs are widely used and convenient tools to model and visualize the structure of various datasets or networks in an interpretable way, and consequently, they are applied in various machine learning applications [15]. However, due to being limited to modeling dyadic interactions in data by means of standard edges, a graph is a suboptimal modeling solution for many datasets, especially for classification data [1, 3]. Moreover, existing graph- and hypergraph-based approaches to machine learning either require the preprocessing step transforming input data to a network, e.g., utilizing some distance measure [16], or embedding of input samples to a vector space [12, 13], which leads to loss of interpretability.

The motivation for building our hypergraph-based modeling framework for binary classification datasets is twofold. Firstly, we apply hypergraphs due to their ability for human-interpretable modeling of dependencies between original feature values and labels. Secondly, in contrast to standard graphs, hypergraphs enable flexible polyadic modeling [17], i.e., modeling interactions occurring within groups of arbitrary size using hyperedges. It has been already proven that this ability of hypergraphs is beneficial in research on properties of social, collaboration, or biological networks [2, 4]. We recognize the features of hypergraphs stated above as crucial for deriving structure-aware importance rates for classification data. Moreover, the proposed importance assessment framework may be used in many practical scenarios. It may be applied in intensively investigated research scenarios, like dealing with imbalanced classification data, feature selection, explanatory data analysis, and algorithm fairness. Both dealing with the class imbalance and feature selection are regarded as important research problems in machine learning, especially when dealing with high dimensional data [9, 11, 18]. Specifically, the undersampling and feature selection methods can be successfully applied to make model training faster, lower model complexity and improve its interpretability, or even improve model efficiency from the perspective of machine learning metrics [8, 19].

The novel contribution of the paper may be summarized as follows:

- We define the hypergraph-based data representation model for the binary classification task. Then, with the aim of walks on the hypergraph, we propose the method to calculate Hypergraph-based Importance (HI) rates for data samples and feature values, which are theoretically grounded on a concept of the hypergraph cut conductance minimization.

- We construct the adaptive undersampling algorithm for imbalanced classification. The algorithm is aimed at reducing the number of elements in the majority class data based on HI rates for data samples.
- We propose the feature selection algorithm using HI rates for values of features.

Furthermore, we confirm the proposed algorithms' effectiveness through experiments involving several datasets of various characteristics.

It must be clearly stated that our approach focuses on datasets with mostly categorical features and it cannot be regarded as a solution for the general classification problem. Such datasets are less explored but not necessarily less common in machine learning applications. We introduce the model enabling us to explore higher-order relations between data elements as hypergraph $H = (V, E)$, with vertices from set V denoting data samples and hyperedges from set E corresponding to values of categorical features used to describe these samples. The information on data labels is introduced as a partition over the hypergraph. Finally, we propose the importance assessment framework based on the hypergraph cut conductance minimization approach, which may be applied to optimize data preparation steps to improve the accuracy and performance of classification algorithms. To the authors' knowledge, such hypergraph application is novel in machine learning.

We evaluate the quality of HI rates using imbalanced data classification and feature selection research scenarios. It is well known that the classification effectiveness for imbalanced datasets is greatly influenced by dataset characteristics [20, 21]. The problems are usually caused by so-called difficulty factors influencing the classification, such as class overlapping, decomposition of the minority class, or the presence of too many minority examples inside the majority class region [11]. In order to ensure satisfactory results for datasets with different characteristics, our undersampling solution introduces a novel mechanism that controls the influence of the hypergraph-derived factor on the final algorithm outcome. On the other hand, the novelty of our feature selection algorithm is related to the ability to calculate the importance rate for every attribute value, taking into account the higher-order relationships in data. For both scenarios, we measure the quality of HI rates through the efficiency evaluation of classification models. Each experiment consists of the phase aimed at data modification— reducing the number of samples for undersampling or reducing the number of features for feature selection—and the phase aimed at quality evaluation based on building the machine learning models on modified data and measuring their efficiency. For efficiency evaluation, we use gradient boosting tree-based classifiers—namely, CatBoost Classifier [22] and LightGBM Classifier [23]—that are known for their proven performance and—at the same time—can handle and effectively exploit categorical features in data. Furthermore, as tree-based algorithms, they allow us to obtain human-interpretable results. Finally, we deliver the publicly available Python library—https://github.com/hypper-team/hypper—that allows our results to be easily reproduced by other scientists.

The rest of the paper is structured as follows. First, we present the state-of-the-art research related to our proposal (Sect. 2). In the central section (Sect. 3), we introduce the paper's scientific contribution consisting of (i) the description of the hypergraph model for binary classification data, (ii) the definition of importance ratings derived from the hypergraph cut conductance minimization approach, and (iii) the description of algorithms for random undersampling optimization and feature selection. Then, we define the evaluation methodology by introducing the details of experimentation scenarios, experimental datasets, the training and testing procedures, baseline methods, hyperparameters, and the evalua-

tion metric (Sect. 4). Next, the results of the experiments are presented and analyzed in Sect. 5. We finish the paper with conclusions and suggestions for further improvements (Sect. 5).

2 Related work

To the authors' knowledge, no research directly addressing this specific problem has already been published in the relevant literature. However, hypergraphs have already been widely used in machine learning, particularly in research on properties of social, collaboration, or biological networks [2, 4, 24]. They have been applied for many application scenarios, including clustering and community detection [3, 4, 25], modeling virus spreading networks [1], exploring biomedical knowledge graphs [24], finding genes which are central in host response to viral infection [2], or prediction for e-commerce [5]. Attention should also be paid to recent developments in hypergraph neural networks [12, 13, 26], and hypergraph signal processing [16, 27]. In contrast to the solution proposed in this paper, the approaches from these areas are not focused on model interpretability. Therefore, they cannot be used to calculate importance rates for data elements. The authors of [16] have proposed the application of hypergraph signal processing to the classification problem. Nevertheless, their solution requires the transformation of input data to the network defined based on distances between input samples described using numerical and binary features before modeling the data as a hypergraph. Similarly, the algorithm proposed by Qu et al. [27] introduces the spectral transformation of hypergraph signal processing. On the other hand, the solutions based on hypergraph neural networks usually assume the application of node embeddings to a vector space and employ datasets delivered in the form of a network [12, 13, 26].

The proposed hypergraph-based data representation and importance rating models are mainly inspired by research presented in [3, 5, 15]. Kaminski et al. [3] have introduced theoretical foundations of hypergraph modeling and practical algorithms for clustering based on hypergraph modularity methods. Chitra and Raphael [15] have investigated the walks on hypergraphs with edge-dependent vertex weights. Finally, Li et al. [5] have applied the concept of local hypergraph cut conductance to optimize product return prediction algorithms.

Numerous papers dealing with the challenge of class imbalance for the classification task have already been published [7, 11, 20]. Many difficulty factors for this problem have been investigated [11, 20], and various solutions have been proposed [6], mainly for the case of datasets given using numerical and binary features. However, none of these solutions applies hypergraph modeling and focuses on optimizing the random undersampling at the same time.

On the other hand, hypergraphs have already been applied for feature selection [10, 28]. Specifically, the authors of [10] have involved a hypergraph data representation to optimize selecting features for the classification algorithm. Their approach differs from the one proposed in this paper since it assumes to model the features as hypergraph vertices and then rate the feature importance based on the hypergraph clustering outcome. In [28], the authors have proposed an unsupervised feature selection method that involves hypergraph-based modeling. In contrast to our approach, their method applies hypergraph embeddings and deals with features extracted from unlabeled data.

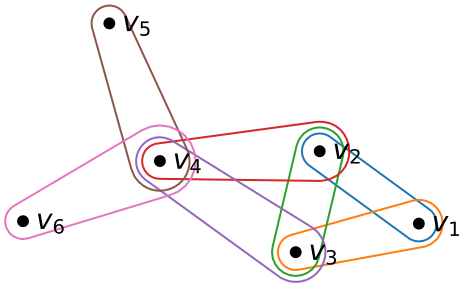


Fig. 1 An example of a simple graph G = (V , E) with V = v1, . . . , v6 and E = v1,v2, v1, v3, v2, v3, v2,v4, v3, v4, v4, v5, v4,v6, (Visualization using the HyperNetX (HNX) library: https://pnnl.github.io/HyperNetX/build/index.html)

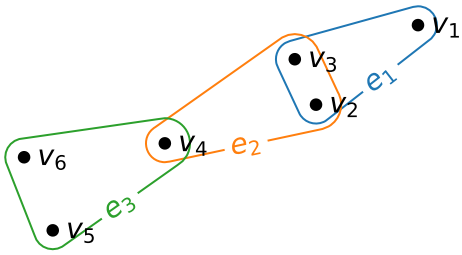


Fig. 2 An example of a hypergraph $H = (V, E)$ with $V = \{v_1, \ldots, v_6\}$ and $E = \{\{v_1, v_2, v_3\}, \{v_2, v_3, v_4\}, \{v_4, v_5, v_6\}\}$

For the experimental research, we follow the evaluation methodology for feature selection algorithms inspired by approaches presented in [8, 18].

3 Proposed method

3.1 Basic definitions

In this section, we follow the notation and definitions provided in [1, 3, 5, 15]. Basic definitions of a simple graph and a hypergraph are given in Definitions 1 and 2, followed by visualizations of examples in Figs. 1 and 2, respectively.

Definition 1 (Simple graph) A graph G is defined as:

$$G = (V, E)$$

where $V = \{v_1, \ldots, v_n\}$ is a set of vertices, and $E \subseteq \{\{v_i, v_j\} : v_i, v_j \in V\}$ is a set of edges.

Definition 2 (Hypergraph) A hypergraph H is defined as:

$$H = (V, E)$$

where vertices are elements of set $V = \{v_1, \ldots, v_n\}$, and hyperedges $e \in E$ are subsets of V of cardinality greater than 1.

For each hyperedge $e \in E$ we define its weight $w(e) \geq 0$. Additionally, we use edge-dependent vertex weights $u(v, e)$ reflecting a contribution of vertex v to hyperedge e. Both weightings allow modeling the impact of hyperedges and vertices based on relations in data converted to the hypergraph. In the case of our framework, we use $u(v, e)$ weights to introduce the information about class imbalance (see details in Subsect. 3.2). Using these weights, we

model random walks on a hypergraph. For a given vertex v, the next random step is determined based on two choices. Firstly, the incident hyperedge e' with probability $p_1 \propto w(e')$ is chosen. Then, the next vertex $v' \in e'$ is determined with probability $p_2 \propto u(v', e')$.

For our algorithms involving the use of a random walk, we introduce the weights normalization. Given a vector $\mathbf{x} = \{x_1, \ldots x_n\}$, we calculate its normalized form $\mathbf{x}' = \{x'_1, \ldots x'_n\}$ based on the L2 normalization for which $x'_i = x_i/z$ where $z = \|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$ and $i = 1, \ldots, n$.

Moreover, we define degree of a vertex v as $d(v) = \sum_{e \in E} u(v, e)w(e)$, and degree of a hyperedge e as $d(e) = \sum_{v \in e} u(v, e)$.

3.2 Modeling binary classification data with a hypergraph

Machine learning classification problem is usually modeled as a task of finding a mapping function $f(\mathbf{x}) = y$ from input variables $\mathbf{x} = (x_1, \ldots, x_m)$ to the discrete output variable y (called the label), where m is a number of variables used to describe input data. For each variable (often called the attribute) $x_i \in \mathcal{D}_i$, where $\mathcal{D}_i$ is a domain of its values. In this paper, we focus on datasets described using discrete (categorical) domains, which may be denoted as $\mathcal{D}_i = \{x_{i,1}, \ldots, x_{i,m_i}\}$, where m_i is a number of different values of attribute x_i. Moreover, for binary classification problem we have $y \in \{0, 1\}$. In our framework, datasets used to train and test machine learning models are represented as sets of samples $(\mathbf{x}, y)$. We model them as hypergraphs $H = (V, E)$ for which:

- V is a set of samples $\{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$,
- E is a set of hyperedges corresponding to the values of attributes $x_{i,j}$ – a given hyperedge consists of vertices corresponding to data samples for which the value of variable x_i is equal to $x_{i,j}$,
- Labels y are modeled as two additional hyperedges which define partition $\mathbf{A} = \{A_1, A_2\}$ of the vertex set V.

The hypergraph structure allows edge weighting $w(e) \geq 0$ and edge-dependent vertex weighting $u(v, e) \geq 0$ [15], enabling to model the local dependencies between attribute values on the data. In the case of our framework, we use edge-dependent vertex weights in order to reflect the information on a class imbalance in the model. For a given partition $\mathbf{A} = \{A_1, A_2\}$, for which A_1 is a minority class we set $u(v, e) = 1$ for $v \in A_1$ and $u(v, e) = |A_1|/|A_2|$ for $v \in A_2$.

Notice that the hypergraph used for binary classification usually has hyperedges containing a relatively large number of vertices. This observation distinguishes our research challenges from the problems investigated in clustering for which hyperedges usually are small [1, 3, 5]. Furthermore, for the clustering case, many hyperedges contain vertices only from one cluster. In contrast, in the case of binary classification, all or almost all hyperedges contain the vertices of both classes.

An example of a hypergraph describing a binary classification dataset is presented in Fig. 3. For the clarity of visualization, hyperedges corresponding to the labels are indicated using the coloring of vertices.

3.3 Hypergraph cut conductance

The idea behind the Hypergraph-based Importance (HI) rating approach proposed in this paper is related to the concept of the hypergraph cut conductance. One can observe that

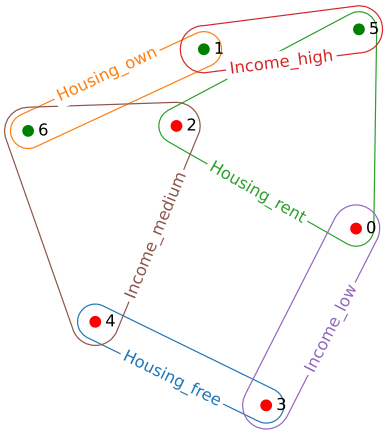


Fig. 3 An example of a hypergraph representing a dataset describing a binary classification problem presented in Table 1 (color figure online)

Table 1 Example of data with a binary label (inspired by German Credit dataset [29]) (color figure online)

Index	Housing	Income	**Credit score (label)**
0	Rent	Low	Bad
1	Own	High	Good
2	Rent	Medium	Bad
3	Free	Low	Bad
4	Free	Medium	Bad
5	Rent	High	Good
6	Own	Medium	Good

binary classification labels correspond to partition $\mathbf{A} = \{A_1, A_2\}$ of vertex set V, which defines a cut on a hypergraph [5]. Formally, the cut on hypergraph $H = (V, E)$ is just a partition that separates the vertex set V into set S and $\bar{S}$, where $\bar{S}$ is the complement of S. This section follows the definitions presented in [5]. Let us define the volume of vertex subset $S \subseteq V$ as $vol(S) = \sum_{v \in S} d(v)$. Then, for a given cut corresponding to the partition $\mathbf{A} = \{A_1, A_2\}$ we introduce the concepts of its boundary and the boundary volume using Definition 3.

Definition 3 (Boundary of the cut (∂A)) The boundary of $\mathbf{A} = \{A_1, A_2\}$ (∂A) is a set of hyperedges that connect A_1 and A_2, i.e., $\partial A = \{e \in E : e \cap A_1 \neq \emptyset, e \cap A_2 \neq \emptyset\}$, for which

$$vol(\partial A) = \sum_{e \in \partial A} w(e) \frac{\left(\sum_{v \in e \cap A_1} u(v, e)\right)\left(\sum_{v \in e \cap A_2} u(v, e)\right)}{d(e)}.$$

The contribution of each hyperedge to the boundary volume reflects the information about the distribution of a given feature value $x_{i,j}$ in minority and majority classes. As a result of vertex weighting introduced in the previous section—for which $u(v, e) = 1$ for v from minority class A_1 and $u(v, e) = |A_1|/|A_2|$ for v from majority class A_2—this contribution is maximal for feature values, whose number of occurrences in each class is proportional to the sizes of these classes. The goal of HI rates is to provide the importance ranking for majority class samples (for the application scenario of undersampling) and for the feature

values (for the application scenario of feature selections), which model the samples' and the feature values' contribution to the hypergraph cut conductance defined in Definition 4. In general, the paper's research goal is to investigate the impact of the hypergraph cut conductance minimization approach applied during data preparation on the efficiency of binary classification models.

Definition 4 (Cut conductance $\Phi(A)$)

$$\Phi(A) = \frac{vol(\partial A)}{\min\{vol(A_1), vol(A_2)\}}$$

It has to be mentioned that the presented approach may be extended to the case of multiclass or multi-label classification, for which the approach of cut conductance minimization should be replaced by hypergraph modularity maximization [3].

3.4 Hypergraph-based importance rates

For our rating framework, we calculate the vertex and the edge contributions to binary-classification hypergraph cut conductance using random walks on hypergraph $H = (V, E)$. We apply a lazy random walk on the hypergraph, for which the choice of the next vertex v' is based on two actions:

- First choose an incident hyperedge e' with probability $p_1 \propto w(e)$,
- Then choose a vertex $v' \in e'$ with probability $p_2 \propto u(v', e')$.

Based on the random walk described above, we define probabilities of class preservation or class change when walking on $H = (V, E)$. Specifically, we define:

- $p_i^H(v, k)$—probability that random walk on H of length $k \geq 1$ starting from vertex $v \in V$ will finish in some vertex from class A_i $(i = 1, 2)$,
- $p_i^H(e, k)$—probability that random walk on H of length $k \geq 1$ using the edge e as the first one will finish in some vertex from class A_i $(i = 1, 2)$,
- $p_i^{H+}(v, k)$—probability that random walk on H of length at most k $(k \geq 2)$ starting from vertex $v \in V$ will finish in some vertex from class A_i $(i = 1, 2)$,
- $p_i^{H+}(e, k)$—probability that random walk on H of length at most k $(k \geq 2)$ using the edge e as the first one will finish in some vertex from class A_i $(i = 1, 2)$.

In the simplest case concerning the edges and $k = 1$, we have:

$$p_i^H(e, 1) = \sum_{v_j \in e \cap A_i} u(v_j, e) \Big/ \sum_{v_j \in e} u(v_j, e),$$

for $i = 1, 2$. Additionally, we investigate approaches applying longer walks for $k \geq 2$. Specifically, for the purposes of experiments presented in this paper, we have checked k from set $\{1, 2, 3\}$. The motivation behind using various values of k is to explore the properties of the hypergraph-based nearest neighborhood of a given vertex or hyperedge.

Based on the probabilities defined above, we introduce three types of HI ratings which we experimentally checked in this paper:

- R_1—the HI rating which promotes vertices and edges 'close' to minority class A_1 defined as follows:
  - $R_1(v, k) = p_1^H(v, k) - p_2^H(v, k)$ for $v \in V$ and $k \geq 1$,

Table 2 Statistics of feature *Purpose* from the German Credit dataset

Feature value	Class-good	Class-bad	Ratio
Car	231	106	0.46
Radio/TV	218	62	0.28
Furniture	123	58	0.47
Business	63	34	0.54
Education	36	23	0.64
Repairs	14	8	0.57
Domestic appliances	8	4	0.50
Vacation/others	7	5	0.71
Whole dataset	700	300	0.43

  - $R_1(e,k) = p_1^H(e,k) - p_2^H(e,k)$ for $e \in E$ and $k \geq 1$,

- R_2—the HI rating which promotes vertices and edges 'close' to majority class A_2 defined as follows:
  - $R_2(v,k) = p_2^H(v,k) - p_1^H(v,k)$ for $v \in V$ and $k \geq 1$,
  - $R_2(e,k) = p_2^H(e,k) - p_1^H(e,k)$ for $e \in E$ and $k \geq 1$,

- R_0—the HI rating which promotes vertices and edges 'close' to one class and 'far' to another, defined as follows:
  - $R_0(v,k) = |p_1^H(v,k) - p_2^H(v,k)|$ for $v \in V$ and $k \geq 1$,
  - $R_0(e,k) = |p_1^H(e,k) - p_2^H(e,k)|$ for $e \in E$ and $k \geq 1$.

Additionally, we investigate ratings R_1^+, R_2^+, and R_0^+ defined analogously with the use of p_i^{H+} instead of p_i^H for $k \geq 2$.

Let us demonstrate the behavior of hypergraph-based importance by means of the experiment conducted using the German Credit dataset (described in Sect. 4). The main objective of this example is to illustrate how the balance of the values for a given feature can influence HI ratings. Additionally, it shows how higher-order relations between feature values can impact the value's importance. The example is focused on the feature describing the purpose of credit application. The feature statistics are summarized in Table 2.

The experiment results are gathered in Table 3. Firstly, we have delivered the baseline results obtained by applying our method to the dataset reduced only to the *Purpose* feature and the label. The positive correlation between the number of feature value occurrences and the output HI rates can be observed. Moreover, the closer the distribution of a given value is to the original class distribution (0.43 for the German Credit dataset, see Table 2), the less its importance is. This explains why value *Radio/TV* is above *Car* in importance ranking. We may observe that features with an equal number of occurrences, which are modeled as hyperedges of the same size, such as *Domestic appliances* and *Vacation/Other*, obtained different importance reflecting their class distributions.

In the second part of the experiment, the proposed hypergraph-based algorithm has been executed using the whole dataset and the ranking variant R_0^+, for $k = 3$. As a result, we have calculated the importance of all the feature-value pairs in the German dataset. The obtained ratings for *Purpose* feature are compared as the final ranking in the right column of Table 3. Experiments show how higher-order relations modeled in hypergraph can affect the

Table 3 Visualization of the hypergraph-based importance rates for feature with unbalanced distribution

	Baseline experiment		Experiment using the whole dataset	
	HI ratings	HI-based ranking	HI-based ranking	HI ratings
1	0.540	Radio/TV	Radio/TV	0.072
2	0.270	Car	Car	0.020
3	0.108	Furniture	Furniture	0.018
4	0.083	Business	Education	0.016
5	0.073	Education	Business	0.015
6	0.017	Repairs	Vacation/other	0.005
7	0.016	Vacation/other	Repairs	0.004
8	0.005	Domestic appliances	Domestic appliances	0.001

Results are presented only for the *Purpose* feature and sorted by HI ratings

feature value importance. We can observe some differences in ranking caused by the impact of hypergraph neighborhood, e.g., for a new ranking *Education* is higher than *Business* and *Vacation/Other* is higher than *Repairs*. Nevertheless, rankings in both cases remain similar. Still, the HI rating for a given feature value may be regarded as some kind of function of the number of occurrences in datasets and the ability to distinguish between classes.

3.5 Algorithms

We design algorithms utilizing the HI rates for random undersampling of imbalanced data and feature selection.

3.5.1 Optimization of random undersampling

In this section, we describe the algorithm for reducing the number of elements in the majority class for imbalanced data. The proposed solution combines the standard random undersampling with the approach based on HI ratings. The motivation behind such an approach is to provide an outcome that preserves global dependencies in the data and introduces the information on local relationships observed in the hypergraph structure. Since the problem depends heavily on data characteristics, the proposed algorithm uses the mechanism that enables control of the influence of the hypergraph-derived factor on the outcome.

Let B_1 and B_2 be subsets of samples defining minority and majority classes, respectively. These subsets correspond to the partition $\mathbf{A} = \{A_1, A_2\}$ of vertices of hypergraph $H = (V, E)$. Let denote $|B_1| = n_1$ and $|B_2| = n_2$. Given a set of majority samples $B_2 = \{x_1, \ldots, x_{n_2}\}$ the undersampling algorithm returns its subset C such as $|C| = t$. In our research, we studied a few settings of t including the standard one for which $t = n_1$.

The *hypper-undersampling()* procedure is presented as a pseudocode in Algorithm 1.

The *random()* procedure used in Algorithm 1 returns the value selected randomly from uniform distribution over the segment [0, 1]. Our method uses α as a hyperparameter tuned during the model training. The method may be seen as the adjustment mechanism for sampling from a uniform distribution. We tested several values of α, including the case of $\alpha = 0$ for which the choice of samples depends on hypergraph-based ratings $\mathbf{r}'$ only. In this setting, the method aims to find the subset A_2' of A_2, which leads to minimum hypergraph cut

Algorithm 1 *hypper-undersampling*$(B_2, \mathbf{r}, t, \alpha)$

Require: $B_2 = (x_1, \ldots, x_{n_2})$ - samples from the majority class, $\mathbf{r} = (r_1, \ldots, r_{n_2})$ - the vector of hypergraph-based ratings for majority samples, t - the size of output set, α - the hyperparameter controlling the impact of hypergraph-based ratings
Ensure: C - the majority class after undersampling
1: Initialize a vector of ratings $\mathbf{r}' = (r'_1, \ldots, r'_{n_2})$ with all entries equal to 0
2: **for** $x_i \in B_2$ **do**
3: $\quad r'_i \Leftarrow r_i + \alpha \cdot random()$
4: **end for**
5: $sort(\mathbf{r}')$
6: set C as a set of all $x_i \in B_2$ for which r'_i is among the top t values in the sorted vector $\mathbf{r}'$

conductance. As a input rates $\mathbf{r}$, we investigated all versions of $R_l(v, k)$ and $R_l^+(v, k)$ (defined in Subsect. 3.4) for $l = 1, 2, 3$ and $k \leq 3$.

Compared to the feature selection algorithm (described in Subsect. 3.5.2), one may observe that the undersampling procedure requires more hyperparameter optimization. The reason is related to difficulty factors that influence the undersampling efficiency, especially those involving class overlapping, decomposition of the minority class, or the presents of many outliers in both classes. It also should be stressed that hypergraphs have a chance to demonstrate their superiority when there are higher-order relationships in the data, and these relationships influence the classification labels. Furthermore, we have observed that the optimal undersampling strategy may depend on the size of the dataset and the imbalance ratio. It is especially noticeable when we attempt to remove many samples from the majority class and leave only a small fraction. In order to address these issues, we introduce the α parameter, which controls the balance between random and hypergraph-derived factors. As confirmed in the experiments, the optimal value of α differs for different dataset characteristics. Additionally, we applied the type of HI ranking as a method hyperparameter involving all variants proposed in Sect. 3.4. This way, we are able to adjust our method for a given dataset characteristic being focused on majority samples most distant from the majority class (R_2 ratings), majority samples from the border between classes (R_1 ratings), or both of them (R_0 ratings).

3.5.2 Feature selection algorithm

This section describes the algorithm for feature selection used in a second application scenario presented in this paper. For each dataset, let denote the features used to describe the samples as $F = \{f_1, \ldots, f_m\}$. As it introduced in Subsect. 3.2, we focus on categorical features with values from discrete domains denoted as $\mathcal{D}_i = \{x_{i,1}, \ldots, x_{i,m_i}\}$, where m_i is a number of different values of feature f_i. The solution is based on ratings for hypergraph edges that correspond to feature values. The procedure returns the subset of most important features of the size controlled by parameter β describing the percentage of features left after selection.

The *hypper-feature-selection()* procedure is presented in Algorithm 2.

In contrast to the undersampling case, the proposed feature selection procedure requires much less extensive hyperparameter optimization. Specifically, we restrict it to checking the ratings from $R_0^+(e, k)$ for $k = 2, 3$. This type of rating treats both classes equally and promotes the features with values that distinguish between them better (see the example presented in Table 3). The only parameter we tune is the extent to which we explore the hyperedge neighborhood.

Algorithm 2 hyper-feature-selection(F, r, β)

Require: $F' = \{f_1, \ldots, f_m\}$ - the set of features used to describe the samples, $\mathbf{r} = (r_{1,1}, \ldots, r_{1,m_1}, r_{2,1}, \ldots, r_{2,m_1}, \ldots, r_{m,1}, \ldots, r_{m,m_m})$ - the vector of hypergraph-based ratings for feature values, β - the parameter controlling the percentage of feature selected
Ensure: $F = \{f_1, \ldots, f_a\}$, where $a = \lceil \beta m \rceil$
1: $a \Leftarrow \lceil \beta m \rceil$
2: Initialize a vector of ratings $\mathbf{r}' = (r'_1, \ldots, r'_m)$ with all entries equal to 0
3: **for** $f_i \in F$ **do**
4: $\quad r'_i \Leftarrow \frac{1}{m_i} \sum_{j=1}^{m_i} r_{i,j}$
5: **end for**
6: $sort(\mathbf{r}')$
7: set F' as a set of all $f_i \in F$ for which r'_i is among the top a values in the sorted vector $\mathbf{r}'$

4 Experimental settings

4.1 Datasets

A broad selection of datasets was used for a better exploratory analysis of the proposed hypergraph-based solutions. Datasets differ in size and characteristics but focus on binary classification tasks. All the datasets are described mainly by categorical features (see Table 4 for details).

The following datasets were used in experiments:

- *Statlog (German Credit) Dataset* [30] (originally from UCI repository [29]) classifies people described by a set of different attributes in terms of good/bad bank credit risk.
- *Criteo Sponsored Search Conversion Log Dataset* [31] classifies product sales based on user actions linked to the product-related advertisement. The original dataset was modified to reduce the number of instances (randomly but preserving the original imbalance ratio) and ignore some features with high entropy. A single sample represents the user action with corresponding product details and user demographics. The classes define if user actions led to conversion (bought product) within the 30 day window.
- *Bank Marketing (Banking) Dataset* [32] classifies term deposit subscription based on the information about client and marketing campaigns. The data were gathered from a Portuguese bank institution's direct marketing campaigns (using phone calls).
- *HR Analytics Dataset* [33] classifies whether a candidate for a job is actually interested in the proposed position. It helps to reduce costs and time required for the training course of candidates looking for new employment. A candidate is defined with various features such as education, experience, and demographics.
- *Phishing Dataset* [34] classifies phishing attacks based on features gathered from 5000 legitimate and 5000 phishing websites. The dataset was created to detect and block phishing attacks where attackers can pose as trusted entities and trick users into sharing sensitive information.
- *Breast Cancer Dataset* [35] classifies if recurrence events, regarding breast cancer, happened for the studied oncology patients. The dataset includes patients' demographics and tumor characteristics.

Table 4 Datasets characteristics summary

Dataset	No. of samples	No. of features	No. of categorical features	No. of feature-value pairs	Minority class samples (%)
German credit	1000	9	6	1033	30
HR analytics	19158	12	10	526	25
Banking	45211	16	10	9541	10
Phishing	10000	30	30	70	50
Breast cancer	286	9	9	45	30
Criteo sponsored search conversion log	100000	16	15	74321	11

4.2 Evaluation methodology

This section presents the evaluation settings for both considered experimentation scenarios: optimization of random undersampling and feature selection.

4.2.1 General settings

Before each experiment, datasets were split into four cross-validation folds. We used 75% of data for each fold as a train set. The remaining 25% of samples were randomly divided into two equal subsets: a validation set and a testing set. We used the validation set to tune the hyperparameters of methods based on HI ratings. For each dataset, we repeated the cross-validation procedure 8 times and reported the average values of final performance provided by means of the evaluation metric. Each experiment consisted of three phases: (i) data preprocessing, (ii) reducing the number of samples (undersampling) or the number of features (feature selection), (iii) final evaluation based on building the machine learning models on modified data and measuring their efficiency using the testing set. When presenting the results of our experiments, we refer to the solution based on HI ratings as the *Hypper* method.

Classification algorithms

For efficiency evaluation, we apply gradient boosting tree-based classifiers, namely CatBoost Classifier [22] and LightGBM Classifier [23]. Firstly, our choice is motivated by the fact that these algorithms are dedicated to handling and effectively exploiting categorical features in data. Secondly, they are known for their proven performance. Specifically, both solutions do not involve the one-hot encoding step, which is crucial when processing bigger and high-dimensional datasets. Finally, as tree-based algorithms, they allow us to obtain human-interpretable results and, simultaneously, can take advantage of discovering the higher-order relations in data. The last feature is of key importance for our research—we believe that proper preprocessing involving HI rates could help these classifiers exploit such relationships when building the tree structures. The details of both selected algorithms are as follows:

- CatBoost classifier [22] implemented using library [36] is a machine learning algorithm developed by Yandex for gradient boosted decision trees. CatBoost handles categorical features without any explicit preprocessing done by the user. Categorical features are transformed into numerical with the use of various statistics on combinations of both

Table 5 Datasets used for benchmarks

Dataset	Undersampling	Feature selection
German credit	√	√
HR analytics	√	
Banking dataset	√	√
Criteo sponsored search conversion log	√	√
Breast cancer		√
Phishing		√

categorical and numerical features. [37]. For our experiments, we applied the CatBoost classifier with default parameters and predefined *random_state*.

- LightGBM classifier [23] implemented using library [38] is a tree-based machine learning algorithm utilizing gradient boosting developed by the Microsoft company. The algorithm can also automatically handle categorical features by finding an optimal split over categories [39]. We applied it with the default settings of parameters and predefined *random_state*.

Evaluation measure

We used Average Precision (AP) [19] as an evaluation metric. This measure is based on precision and recall, the most popular state-of-the-art binary classification metrics. In opposition to them, it enables rating the ranked sequence of samples provided for testing. AP calculates the model performance as a weighted mean of precision achieved at different thresholds used to determine the classification outcome. This way, the results are not dependent on the specific threshold setting, which might heavily influence the performance assessment, especially in the case of imbalanced data. We used the implementation of AP metric from the *scikit-learn* library [40].

4.2.2 Undersampling

We tested the efficiency of the *Hypper* method for the task of reducing the number of samples in the majority class for the classification of imbalanced data. As presented in Table 5, for these experiments, we chose the largest datasets with the smallest imbalance ratio (IR) calculated as n_1/n_2 [41], namely Criteo, Banking, and HR Analytics datasets. In order to test how the method performs for a small dataset, we conducted additional tests using the German Credit dataset.

Hypper method settings

Our main goal was to compare the performance of the undersampling algorithm based on the *Hypper* method with the random undersampling approach. This scenario assumes investigating the case of reducing the number of majority samples to the number of minority samples. To extend the analysis, we tested the *Hypper* method for two additional settings for the number of majority samples after reduction, each time comparing it with the equivalent random undersampling. Finally, we analyzed three settings of parameter t in our tests:

- $t = n_1 + 0.0 \cdot n_2$, where n_1 and n_2 denote the sizes of minority and majority classes, respectively—the method referred to as *Hypper 0.0* (the undersampling which equalizes the numbers of majority and minority samples),
- $t = n_1 + 0.25 \cdot n_2$—the method referred to as *Hypper 0.25*,

- $t = n_1 + 0.5 \cdot n_2$—the method referred to as *Hypper 0.5*.

In order to tune the method, we conducted the grid search approach, including the settings of the following hyperparameters:

- The rating vector $\mathbf{r} = (r_1, \ldots, r_{n_2})$: $R_l(v, k)$ for $l = 1, 2, 3$ and $k = 1, 2, 3$, and $R_l^+(v, k)$ for $l = 1, 2, 3$ and $k = 2, 3$,
- α: 0, 5, 40, 500,
- t: $n_1 + 0.0 \cdot n_2$, $n_1 + 0.25 \cdot n_2$, and $n_1 + 0.5 \cdot n_2$.

We tested several values of α including the case of $\alpha = 0$ for which the choice of samples depends on hypergraph-based ratings $\mathbf{r}$ only, and the case of $\alpha = 500$ for which the impact of HI rates is negligible, so basically, the *Hypper* method follows the random undersampling.

Compared undersampling methods

We compared the effectiveness of the proposed undersampling algorithm with the following state of the art and baseline solutions:

- Tomek Links [42] (we used implementation from [43]). One of the cleaning–based techniques that do not allow to specify the exact number of samples for the majority class after reduction. This method detects Tomek's links and removes samples from a majority class that are also a part of Tomek's link.
- Edited Nearest Neighbors [44] (we used implementation from [45]). The technique uses the k-nearest neighbor algorithm (with Euclidean distance and $K = 3$) to determine if the sample and its neighbors should be remain or be removed. The sample is removed when the majority class from the k-nearest neighbors is different from the sample class.
- Random undersampling [46] (tested using cases *Random 0.0*, *Random 0.25*, and *Random 0.5*), which selects a set of samples based on the uniform distribution (without replacement) from the majority class to match the desired proportion of each class.
- Without undersampling used as a baseline.

4.2.3 Feature selection

In the second application scenario, we evaluated the efficiency of the *Hypper* method for the feature selection task. As seen in Table 5, we have chosen datasets of various characteristics for this scenario, namely German Credit, Banking, Phishing, Breast Cancer, and Criteo datasets. We have excluded only one dataset, namely the HR dataset, due to the presence of missing values in many samples. Specifically, this dataset contains several features with missing values for more than 20% of samples. While missing values do not influence the hypergraph model, this issue is harmful to other state-of-the-art methods used for effectiveness comparison. We have observed that applying the standard approach for compensating missing values using the dedicated default values leads to outcomes for which both state-of-the-art methods achieve worse efficiency than the random baseline approach. Therefore, we have decided not to include these results in order not to bias the conclusions.

Hypper method settings

Using the grid search approach, we investigated the following settings of *Hypper* method's hyperparameters:

- The rating vector $\mathbf{r} = (r_1, \ldots, r_{n_2})$: $R_l^+(v, k)$ for $k = 2, 3$;
- β: 0.15, 0.30, 0.5.

The choice of β settings was motivated by the solutions applied in relevant publications [8, 18].

Compared feature selection methods

We compared the effectiveness of feature selection based on the *Hypper* method with the following state-of-the-art algorithms:

- Feature importance ranking based on Random Forest Classifier [47],
- Feature importance ranking based on Logistic Regression Classifier [48] using absolute values of the Logistic Regression coefficients,
- Random feature importance ranking (used as a baseline).

5 Experimental results

For both experimentation scenarios, we demonstrated the methods' efficiency using tables with detailed results (see Tables 6 and 7) and charts visualizing the efficiency comparison for chosen scenarios (see Figs. 4, 6, and 8). Moreover, we used Friedman's test and Nemenyi's post hoc test [49] to determine whether the differences between the average ranked performances of compared methods are statistically significant.

In the case of Friedman's tests, we followed the approach presented in [50]. For each investigated case, the Friedman rank sum test statistics was greater than the critical value for the assumed level of confidence $\alpha = 0.1$ [49]. Therefore, we concluded that for the level of confidence $\alpha = 0.1$, the null hypothesis that there is no difference between the methods could be rejected, and the post hoc Nemenyi test can be conducted to investigate the difference between each pair of individual approaches. Finally, we provided the significance diagrams (see Figs. 5, 7, 9, and 10) in a form proposed in [49] to illustrate the ranked performance of the compared solutions along with the critical difference (CD).

5.1 Undersampling

Table 6 presents the general performance comparison of undersampling methods measured in terms of average precision for classification models built using the CatBoost and LightGBM algorithms. When comparing CatBoost with LightGBM, one may notice that CatBoost performs better for each dataset and each undersampling method. However, their results are consistent, i.e., they lead to the same conclusions from the perspective of the method used to reduce the number of samples in the majority class. It has to be noticed that for the case of big datasets with a small imbalance ratio, i.e., dataset typical for the undersampling scenario, the best performance is achieved for Tomek Links and the classifier built using unmodified data (the without undersampling baseline). On the other hand, for the small dataset, i.e., German Credit, the *Hypper* method outperforms other compared solutions, whereas Tomek Links and without undersampling obtain the worst results. Such an outcome may be explained by the specific characteristics of this dataset, which contains only 1000 samples with a relatively high imbalance ratio equal to 3/7. Moreover, the AP metric used for evaluation obtains better results for methods that are more precise at the beginning of the ranking used to calculate the metric value. These observations lead to the general conclusion that the method performance depends on the number of removed samples, e.g., the Tomek Links method, which removes about 5% of samples, obtains evaluation results comparable to the baseline approach—without undersampling. Nevertheless, it has to be stressed that the proposed *Hyp-*

Table 6 The evaluation results in terms of the AP metric for the undersampling scenario

		Hypper	Tomek links	Edited nearest neighbors	Random	Without undersampling
Dataset	Evaluation algorithm					
German credit	CatBoost	0.89 ± 0.04	0.80 ± 0.03	0.86 ± 0.04	0.85 ± 0.05	0.79 ± 0.03
	LightGBM	0.88 ± 0.04	0.80 ± 0.03	0.83 ± 0.05	0.84 ± 0.04	0.79 ± 0.03
Banking	CatBoost	0.60 ± 0.01	0.63 ± 0.02	0.55 ± 0.01	0.43 ± 0.01	0.65 ± 0.02
	LightGBM	0.59 ± 0.01	0.62 ± 0.02	0.53 ± 0.01	0.42 ± 0.01	0.65 ± 0.02
HR analytics	CatBoost	0.59 ± 0.02	0.59 ± 0.02	0.54 ± 0.02	0.53 ± 0.02	0.59 ± 0.02
	LightGBM	0.58 ± 0.02	0.59 ± 0.02	0.55 ± 0.02	0.53 ± 0.02	0.59 ± 0.02
Criteo	CatBoost	0.42 ± 0.02	0.46 ± 0.04	0.35 ± 0.03	0.19 ± 0.00	0.47 ± 0.06
	LightGBM	0.43 ± 0.04	0.46 ± 0.09	0.35 ± 0.03	0.19 ± 0.01	0.46 ± 0.13

Table 7 The evaluation results in terms of the AP metric for the feature selection scenario

		Hypper		Random forest		Logistic regression		Random	
Dataset	β[a] (%)	CatBoost	LightGBM	CatBoost	LightGBM	CatBoost	LightGBM	CatBoost	LightGBM
German credit	15	0.75 ± 0.03	0.76 ± 0.04	0.72 ± 0.03	0.73 ± 0.04	0.74 ± 0.04	0.74 ± 0.04	0.70 ± 0.03	0.70 ± 0.03
	30	0.76 ± 0.04	0.77 ± 0.04	0.74 ± 0.03	0.74 ± 0.04	0.76 ± 0.04	0.76 ± 0.04	0.70 ± 0.03	0.71 ± 0.03
	50	0.79 ± 0.03	0.79 ± 0.03	0.78 ± 0.03	0.78 ± 0.03	0.77 ± 0.04	0.78 ± 0.04	0.71 ± 0.03	0.71 ± 0.03
Breast cancer	15	0.50 ± 0.22	0.54 ± 0.23	0.22 ± 0.16	0.26 ± 0.21	0.32 ± 0.19	0.33 ± 0.22	0.01 ± 0.06	0.00 ± 0.00
	30	0.46 ± 0.19	0.45 ± 0.21	0.28 ± 0.17	0.27 ± 0.17	0.38 ± 0.15	0.40 ± 0.23	0.09 ± 0.21	0.12 ± 0.17
	50	0.49 ± 0.19	0.45 ± 0.22	0.37 ± 0.17	0.35 ± 0.15	0.44 ± 0.20	0.42 ± 0.16	0.40 ± 0.24	0.33 ± 0.18
Phishing	15	0.93 ± 0.01	0.93 ± 0.01	0.94 ± 0.01	0.94 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.61 ± 0.01	0.61 ± 0.01
	30	0.94 ± 0.01	0.94 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.90 ± 0.01	0.90 ± 0.01
	50	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.94 ± 0.01	0.94 ± 0.01
Banking	15	0.54 ± 0.03	0.56 ± 0.03	0.56 ± 0.02	0.57 ± 0.03	0.56 ± 0.22	0.56 ± 0.22	0.45 ± 0.09	0.46 ± 0.09
	30	0.60 ± 0.04	0.60 ± 0.03	0.61 ± 0.02	0.61 ± 0.02	0.65 ± 0.03	0.65 ± 0.03	0.45 ± 0.08	0.49 ± 0.10
	50	0.65 ± 0.02	0.64 ± 0.02	0.64 ± 0.02	0.65 ± 0.02	0.65 ± 0.02	0.66 ± 0.03	0.50 ± 0.07	0.48 ± 0.07
Criteo	15	0.49 ± 0.08	0.53 ± 0.11	0.42 ± 0.09	0.43 ± 0.34	0.23 ± 0.42	0.00 ± 0.00	0.14 ± 0.22	0.03 ± 0.11
	30	0.49 ± 0.06	0.53 ± 0.10	0.49 ± 0.09	0.47 ± 0.18	0.15 ± 0.15	0.13 ± 0.29	0.47 ± 0.06	0.44 ± 0.18
	50	0.51 ± 0.07	0.47 ± 0.09	0.49 ± 0.08	0.44 ± 0.16	0.35 ± 0.11	0.08 ± 0.15	0.44 ± 0.07	0.47 ± 0.18

[a]Percentage of features left after feature selection step

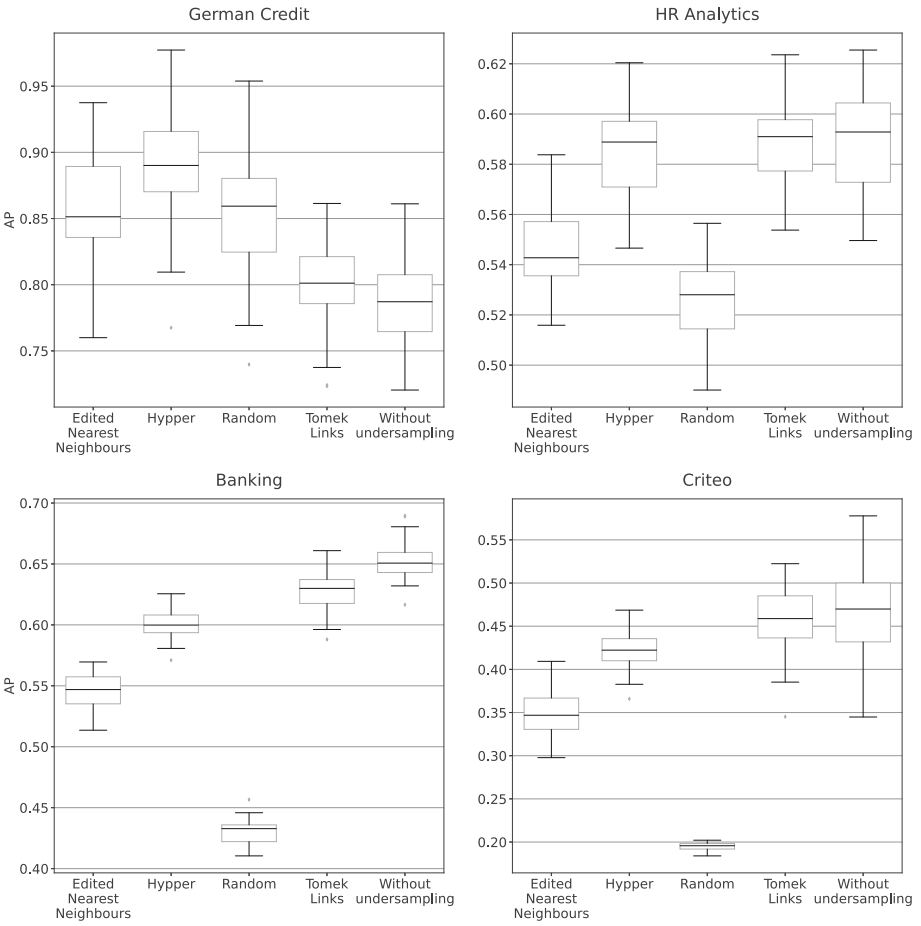
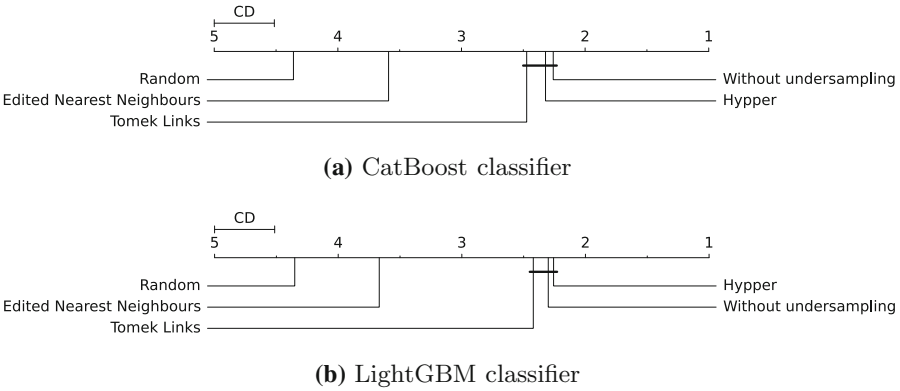


Fig. 4 The undersampling scenario: the performance comparison in terms of the AP metric for the case of the CatBoost classifier

per method outperforms the random undersampling and Edited Nearest Neighbors approach for each benchmark dataset.

The visualizations of experiment results for the CatBoost classifier (see Fig. 4) and the Nemenyi's tests (see Fig. 5) for both CatBoost and LightGBM confirm the conclusions stated above. In order to illustrate the value of the *Hypper* method, we provide its additional comparison with the random undersampling approach for the scenario in which both methods reduce the majority class to precisely the same size. The results are illustrated using Fig. 6 (the case of CatBoost-based evaluation) and Fig. 7 (the visualization of Nemenyi's statistical test).

The results have shown that the *Hypper* undersampling method is more efficient for smaller datasets such as German Credit. In this case, the hypergraph-based undersampling procedure removes a small number of samples from the majority class, so it is able to do it more precisely. We have observed during the hyperparameter tuning phase that for this dataset, the optimal efficiency has usually been achieved for small α and the R_2 rating type. Small α means that the hypergraph-based factor plays a more important role than the random one.



(a) CatBoost classifier

(b) LightGBM classifier

Fig. 5 The undersampling scenario: visualization of Nemenyi's test results for the AP measure

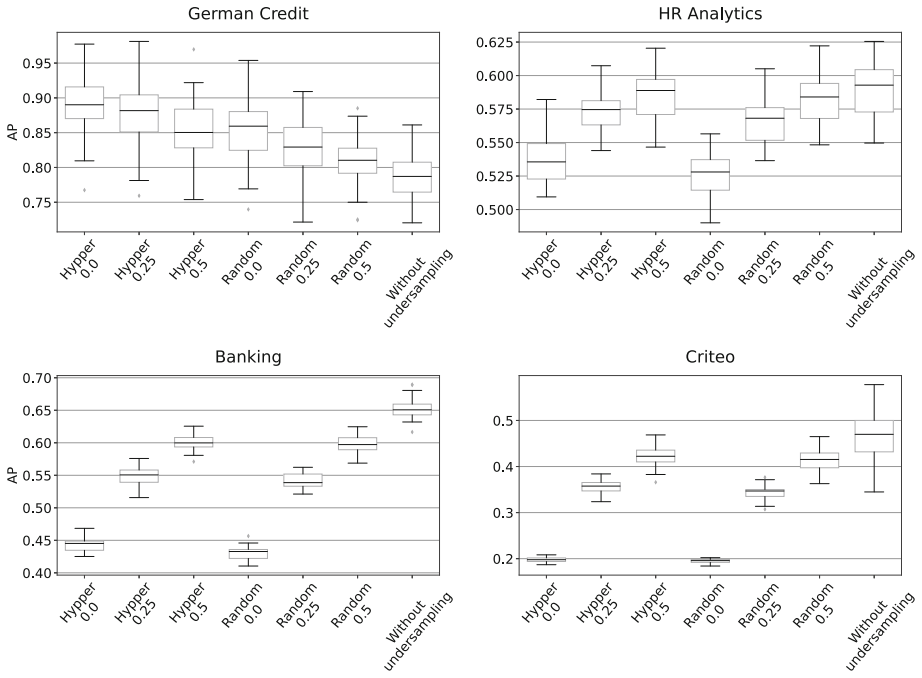
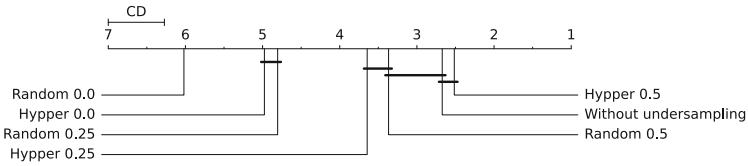
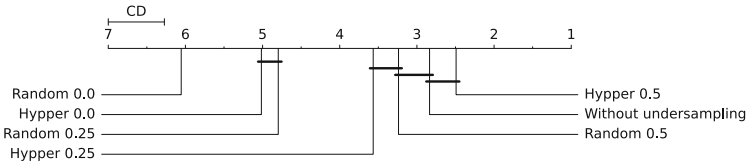


Fig. 6 The undersampling scenario: the performance comparison of the *Hypper* method against the random undersampling approach in terms of the AP metric for the case of the CatBoost classifier

The straightforward strategy R_2 to remove these majority samples, which are closest to the minority class, has proven to be better than the remaining ones. On the other hand, the results for bigger datasets are different. The efficiency improvement is smaller for their case. Firstly, they are less academic and more real-world datasets with more outliers, some missing values, and class overlapping. Secondly, in their case, the undersampling procedure has to remove a large number of samples and remain only a small part of the majority class in a way preserving the global dependencies in data. For these datasets, as optimal hyperparameters settings, we have obtained bigger values of α—which means a more significant impact of



(a) CatBoost classifier



(b) LightGBM classifier

Fig. 7 The undersampling scenario: visualization of Nemenyi's test results for the AP measure (Hypper vs Random comparison)

random factor—and R_0 or R_1 HI ratings types. The choice of the rating type is possibly related to the existence of many data outliers in the majority class, which are not valuable for model training.

5.2 Feature selection

The outcomes of experiments for the feature selection scenario are collected in Table 7 and illustrated in Fig. 8.

Similar to the case of undersampling, the results are consistent for both classifiers applied to build the model for final evaluation, namely for CatBoost and LightGBM. It may be noticed (see Table 7 and Figs. 8, 9 and 10) that the *Hypper* method outperforms the baseline approaches used for the comparison, especially for the case of $\beta = 15\%$. Additionally, one can observe that the selection method utilizing the feature importance ranking based on Random Forest Classifier leads to better results than the method using feature importance derived from Logistic Regression coefficients.

We complement the result presentation using visualizations of post hoc Nemenyi's tests for both CatBoost and LightGBM classifiers and each investigated setting of the parameter β, i.e., $\beta \in \{0.15, 0.30, 0.50\}$ (see Figs. 9 and 10).

The results have proven that the Hypper feature selection method is more efficient for smaller datasets, namely Brest Cancer and German Credit. The reason for that is related to their characteristics as academic datasets with categorical features of various distributions usually correlated to classification labels (see Table 2 from Sect. 3.4 for an example). For the remaining larger datasets, the efficiency improvement of the proposed method is not evident. In the case of the Phishing dataset, which consists only of binary and trinary features, the underlying hypergraph model contains only very large hyperedges, which can be a difficulty factor for our approach. Similarly, the biggest Banking and Criteo datasets also lead to specific hypergraph structure with hyperedges of very big size corresponding to most popular or default values and a very big number of small hyperedges corresponding to the values occurring only a few times in data.

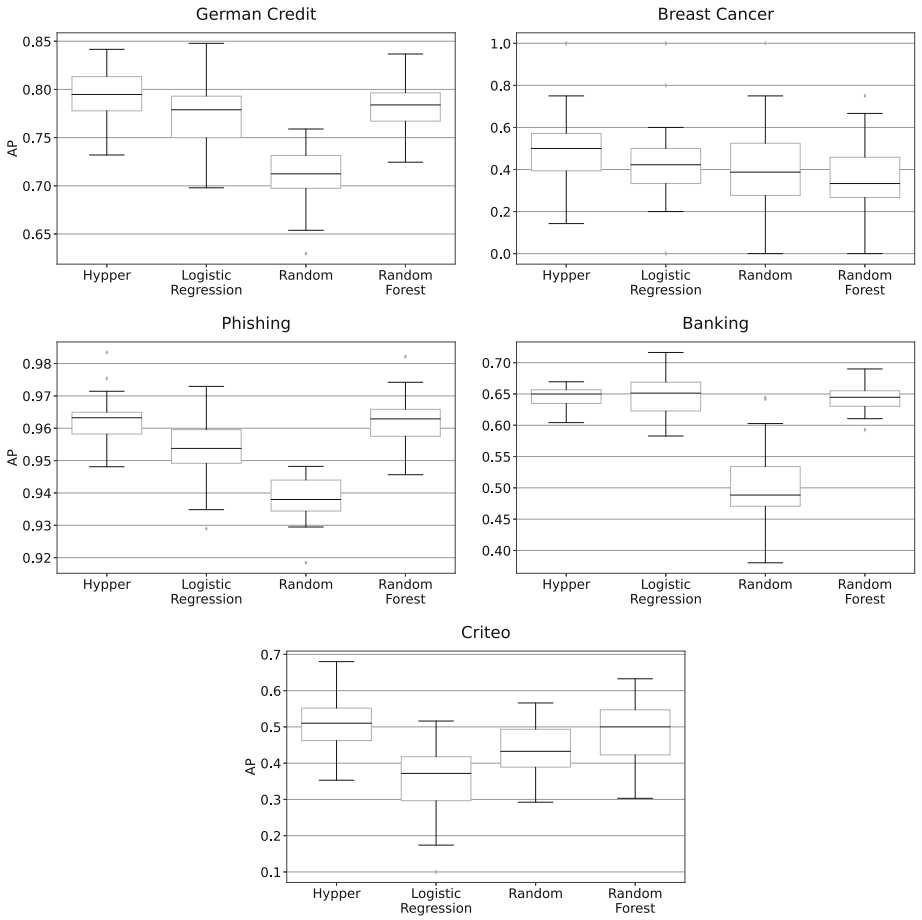


Fig. 8 The features selection scenario: the performance comparison in terms of the AP metric for the case of the CatBoost classifier and $\beta = 0.5$

6 Conclusions and future work

In our studies, we show that hypergraphs, which are regarded as a lossless representation from the perspective of higher-order relationships in data, can be successfully applied to model binary classification datasets. Moreover, our results indicate that such a representation may be applied to rate the importance of data elements from the perspective of their relevance to classification labels. The effectiveness of the proposed algorithms in the undersampling and feature selection scenarios has been confirmed by the results of experiments involving several datasets of various characteristics.

One of the key findings of the research presented in this paper is that the practical value of hypergraph modeling depends heavily on the characteristics of a given dataset, especially for the scenario aimed at reducing the number of samples in the majority class. For this scenario, the adaptive mechanism enabling controlling the influence of hypergraph-based rates on the algorithm outcome was critical in ensuring that the solution leads to satisfactory results for datasets with different characteristics. From this perspective, our approach is consistent with

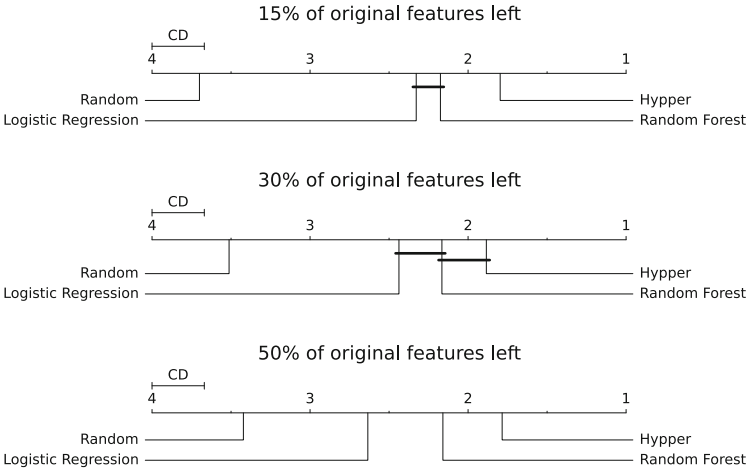


Fig. 9 The features selection scenario: visualization of Nemenyi's test results for the AP measure and the CatBoost classifier

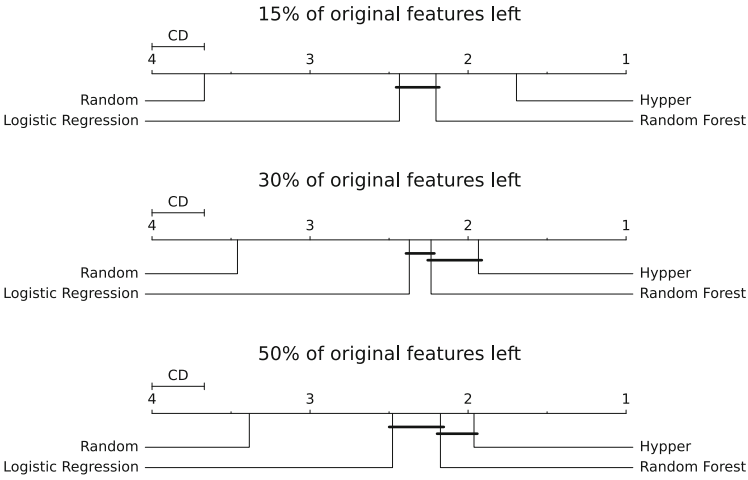


Fig. 10 The features selection scenario: visualization of Nemenyi's test results for the AP measure and the LightGBM classifier

the recent postulate of Andrew Ng for more data-centric rather than model-centric approaches to machine learning, confirming the validity of his observations [51].

Our research opens several future work directions. One interesting way to proceed is to investigate additional undersampling algorithms based on the proposed adaptive mechanism for sampling from uniform distribution but using modification factors different from the HI rates. Other directions for future research may focus on the case of multi-label or multi-class classification, for which the hypergraph cut conductance minimization approach may be replaced by the hypergraph modularity maximization technique [3]. Also, it would be

interesting to build the classifier based on HI rates and compare its efficiency with similar machine learning techniques such as the k-nearest neighbors algorithm.

Finally, it has to be admitted that focusing on processing only the categorical features is a limitation of our approach. Therefore, the research on introducing the numerical features to the hypergraph model is planned, involving the application of discretization techniques. This enhancement is necessary to extend the solution to a general binary classification problem. Nevertheless, it involves new non-trivial modeling and research problems. Firstly, to ensure the fair impact of various types of features on the method outcome, the proper discretization technique should be chosen with parameters adjusted to the characteristics of categorical attributes in a given dataset. Secondly, the discretization of numeric attributes leads to ordinal features. Therefore, an auxiliary modeling technique, possibly using additional weighted hyperedges, is needed to reflect the order of discretized values for a given numeric feature.

Supplementary information

Hypper standalone open-source library has been implemented for the purpose of this research. Hypper is working with Python 3.7 or above. The repository for the project is publicly available at https://github.com/hypper-team/hypper. The library can be installed using PyPI (https://pypi.org/project/hypper/). Documentation for the project is located at https://hypper-team.github.io/hypper.html. Specifically, the methods of importance assessment described in this paper are implemented within the library. Furthermore, python scripts and notebooks used for undersampling and feature selection benchmarks are located in the *benchmarks/* directory. Paper results can be easily reproducible using datasets provided in public repositories.

Acknowledgements This work was supported by the Polish National Science Centre, Grant DEC-2019/03/X/ST6/01023.

Author contributions Pawel Misiorek wrote the main manuscript text including all sections contained in a paper. Szymon Janowski contributed to the section concerning experimental settings by providing descriptions of datasets, evaluation methodology, and compared methods. He prepared all figures and tables. Both authors reviewed the manuscript and prepared its revised version.

Declarations

Conflict of interest The authors declare no conflict of interest.

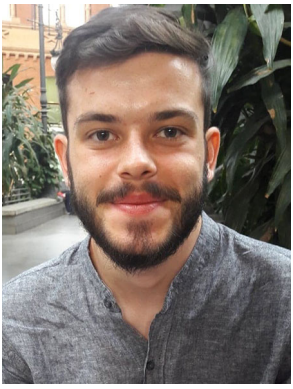
References

1. Aksoy SG, Joslyn C, Ortiz Marrero C, Praggastis B, Purvine E (2020) Hypernetwork science via high-order hypergraph walks. EPJ Data Sci. https://doi.org/10.1140/epjds/s13688-020-00231-0

2. Feng S, Heath E, Jefferson BA, Joslyn CA, Kvinge H, Mitchell HD, Praggastis B, Eisfeld AJ, Sims AC, Thackray LB, Fan S, Walters KB, Halfmann PJ, Westhoff-Smith D, Tan Q, Menachery VD, Sheahan TP, Cockrell AS, Kocher JF, Stratton KG, Heller NC, Bramer LM, Diamond MS, Baric RS, Waters KM, Kawaoka Y, McDermott JE, Purvine E (2021) Hypergraph models of biological networks to identify genes critical to pathogenic viral response. BMC Bioinform. 22(1):287. https://doi.org/10.1186/s12859-021-04197-2
3. Kaminski B, Poulin V, Pralat P, Szufel P, Théberge F (2019) Clustering via hypergraph modularity. PLoS One 14(11):1–15. https://doi.org/10.1371/journal.pone.0224307
4. Kumar T, Vaidyanathan S, Ananthapadmanabhan H, Parthasarathy S, Ravindran B (2020) Hypergraph clustering by iteratively reweighted modularity maximization. Appl Netw Sci 5(1):52. https://doi.org/10.1007/s41109-020-00300-3
5. Li J, He J, Zhu Y (2018) E-tail product return prediction via hypergraph-based local graph cut. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '18. Association for Computing Machinery: New York, NY, pp. 519–527. https://doi.org/10.1145/3219819.3219829
6. Leevy JL, Khoshgoftaar TM, Bauder RA, Seliya N (2018) A survey on addressing high-class imbalance in big data. J Big Data 5:42
7. Lango M, Stefanowski J (2017) Multi-class and feature selection extensions of roughly balanced bagging for imbalanced data. J Intell Inf Syst 50:97–127
8. Saarela M, Jauhiainen S (2021) Comparison of feature importance measures as explanations for classification models. SN Appl Sci 3(2):272. https://doi.org/10.1007/s42452-021-04148-9
9. Urkullu A, Pérez A, Calvo B (2021) Statistical model for reproducibility in ranking-based feature selection. Knowl Inf Syst 63(2):379–410. https://doi.org/10.1007/s10115-020-01519-3
10. Zhang Z, Hancock ER (2011) A hypergraph-based approach to feature selection. In: Real P, Diaz-Pernil D, Molina-Abril H, Berciano A, Kropatsch W (eds) Computer analysis of images and patterns. Springer, Berlin, Heidelberg, pp 228–235
11. Stefanowski J (2016) Dealing with data difficulty factors while learning from imbalanced data. In: Matwin S, Mielniczuk J (eds) Challenges in computational statistics and data mining. Springer, Cham, pp 333–363. https://doi.org/10.1007/978-3-319-18781-5_17
12. Yadati N, Nimishakavi M, Yadav P, Nitin V, Louis A, Talukdar PP (2019) Hypergcn: a new method for training graph convolutional networks on hypergraphs. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R (eds.) Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., Red Hook, NY, USA, pp. 1–12
13. Feng Y, You H, Zhang Z, Ji R, Gao Y (2019) Hypergraph neural networks. Proceedings of the AAAI conference on artificial intelligence 33(01):3558–3565. https://doi.org/10.1609/aaai.v33i01.33013558
14. Pessach D, Shmueli E (2022) A review on fairness in machine learning. ACM Comput Surv. https://doi.org/10.1145/3494672
15. Chitra U, Raphael BJ (2019) Random walks on hypergraphs with edge-dependent vertex weights. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Proceedings of machine learning research, vol. 97, pp. 1172–1181. PMLR, USA. http://proceedings.mlr.press/v97/chitra19a.html
16. Zhang S, Ding Z, Cui S (2020) Introducing hypergraph signal processing: theoretical foundation and practical applications. IEEE Int Things J 7:639–660
17. Chodrow PS (2020) Configuration models of random hypergraphs. J Complex Netw **8**(3) https://academic.oup.com/comnet/article-pdf/8/3/cnaa018/33559166/cnaa018.pdf. https://doi.org/10.1093/comnet/cnaa018
18. Chen RC, Dewi C, Huang S-W, Caraka RE (2020) Selecting critical features for data classification based on machine learning methods. J Big Data 7:1–26
19. Flach P (2012) Machine learning: the art and science of algorithms that make sense of data. Cambridge University Press, New York, NY
20. Brzezinski D, Minku LL, Pewinski T, Stefanowski J, Szumaczuk A (2021) The impact of data difficulty factors on classification of imbalanced and concept drifting data streams. Knowl Inf Syst 63(6):1429–1469. https://doi.org/10.1007/s10115-021-01560-w
21. Kwon O, Sim JM (2013) Effects of data set features on the performances of classification algorithms. Expert Syst Appl 40(5):1847–1857. https://doi.org/10.1016/j.eswa.2012.09.017
22. Dorogush AV, Gulin A, Gusev G, Kazeev N, Prokhorenkova LO, Vorobev A (2017) Fighting biases with dynamic boosting. CoRR abs/1706.09516. 1706.09516
23. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu T-Y (2017) Lightgbm: a highly efficient gradient boosting decision tree. Adv Neural Inf Process Syst 30:3146–3154

24. Dörpinghaus J, Stefan A, Schultz B, Jacobs M (2022) Context mining and graph queries on giant biomedical knowledge graphs. Knowl Inf Syst 64(5):1239–1262. https://doi.org/10.1007/s10115-022-01668-7
25. Hu T, Liu C, Tang Y, Sun J, Xiong H, Sung SY (2014) High-dimensional clustering: a clique-based hypergraph partitioning framework. Knowl Inf Syst 39(1):61–88. https://doi.org/10.1007/s10115-012-0609-3
26. Bai S, Zhang F, Torr PHS (2021) Hypergraph convolution and hypergraph attention. Pattern Recognit 110:107637. https://doi.org/10.1016/j.patcog.2020.107637
27. Qu R, Feng H, Xu C, Hu B (2022) Analysis of hypergraph signals via high-order total variation. Symmetry. https://doi.org/10.3390/sym14030543
28. He W, Cheng X, Hu R, Zhu Y, Wen G (2017) Feature self-representation based hypergraph unsupervised feature selection via low-rank representation. Neurocomputing 253:127–134. https://doi.org/10.1016/j.neucom.2016.10.087
29. University of California, Irvine (UCI), Machine learning repository: statlog (German Credit Data) Dataset. https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data) (2022)
30. Kaggle: German credit risk dataset. https://www.kaggle.com/kabure/german-credit-data-with-risk/ (2022)
31. Tallis M, Yadav P (2018) Reacting to variations in product demand: an application for conversion rate (cr) prediction in sponsored search. arXiv preprint arXiv:1806.08211
32. Kaggle: banking dataset. https://www.kaggle.com/prakharrathi25/banking-dataset-marketing-targets (2022)
33. Kaggle: HR analytics dataset. https://www.kaggle.com/arashnic/hr-analytics-job-change-of-data-scientists (2022)
34. Kaggle: phishing dataset. https://www.kaggle.com/shashwatwork/phishing-dataset-for-machine-learning (2022)
35. University of California, Irvine (UCI), machine learning repository: breast cancer dataset. https://archive.ics.uci.edu/ml/datasets/breast+cancer (2022)
36. Yandex: Catboost - open-source gradient boosting library. https://catboost.ai/) (2022)
37. CatBoost: Transforming categorical features to numerical features. https://catboost.ai/en/docs/concepts/algorithm-main-stages_cat-to-numberic (2022)
38. Microsoft corporation: LightGBM. https://lightgbm.readthedocs.io/ (2022)
39. LightGBM: optimal split for categorical features. https://lightgbm.readthedocs.io/en/latest/Features.html#optimal-split-for-categorical-features (2022)
40. Scikit-learn: machine learning in python. https://scikit-learn.org (2022)
41. Zhu R, Guo Y, Xue J-H (2020) Adjusting the imbalance ratio by the dimensionality of imbalanced data. Pattern Recogn Lett 133:217–223. https://doi.org/10.1016/j.patrec.2020.03.004
42. Tomek I (1976) Two Modifications of CNN. IEEE Trans Syst Man Cybern 7(2):679–772
43. Imbalanced-learn: Tomek links. https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.TomekLinks.html (2022)
44. Wilson DL (1972) Asymptotic properties of nearest neighbor rules using edited data. Syst Man Cybern IEEE Trans 2(3):408–421. https://doi.org/10.1109/TSMC.1972.4309137
45. Imbalanced-learn: edited nearest neighbours. https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.EditedNearestNeighbours.html (2022)
46. Imbalanced-learn: random undersampler. https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html (2022)
47. Scikit-learn: random forest classifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (2022)
48. Scikit-learn: logistic regression. https://scikit-learn.org/stable/mod-ules/generated/sklearn.linear_model.LogisticRegression.html (2022)
49. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
50. Wang H, Xu Q, Zhou L (2015) Large unbalanced credit scoring using lasso-logistic regression ensemble. PloS One 10(2):0117844
51. Ng, A.: MLOps: From model-centric to data-centric AI. DeepLearning.AI https://www.deeplearning.ai/wp-content/uploads/2021/06/MLOps-From-Model-centric-to-Data-centric-AI.pdf (2021)

Pawel Misiorek received his MSc degree in computer science and Ph.D. degree in mathematics from Adam Mickiewicz University Poznan, Poland, in 2002 and 2008, respectively. He is currently working as an assistant professor at the Institute of Computing Science, Poznan University of Technology, Poland. His research interests include data science and machine learning, in particular the application of hypergraphs in data modeling and analysis, random graphs, big data processing, classification algorithms, and semantic models.



Szymon Janowski is a machine learning engineer with a data science background. He received his BSc and MSc degrees in computer science from Poznan University of Technology, Poland, in 2020 and 2021, respectively. His research interests include graph theory, machine learning applications, and data mining.