Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Improving Imbalanced Classification by Anomaly Detection

Jiawen Kong, Wojtek Kowalczyk, Stefan Menzel, Thomas Bäck

2020

Preprint:

This is a post-peer-review, pre-copyedit version of an article published in Parallel Problem Solving from Nature (PPSN). The final authenticated version is available online at: https://doi.org/10.1007/978-3-030-58112-1_35



Improving Imbalanced Classification by Anomaly Detection

Jiawen Kong[1(✉)], Wojtek Kowalczyk[1], Stefan Menzel[2], and Thomas Bäck[1]

[1] Leiden University, Leiden, The Netherlands
{j.kong,w.j.kowalczyk,t.h.w.baeck}@liacs.leidenuniv.nl
[2] Honda Research Institute Europe GmbH, Offenbach, Germany
stefan.menzel@honda-ri.de

Abstract. Although the anomaly detection problem can be considered as an extreme case of class imbalance problem, very few studies consider improving class imbalance classification with anomaly detection ideas. Most data-level approaches in the imbalanced learning domain aim to introduce more information to the original dataset by generating synthetic samples. However, in this paper, we gain additional information in another way, by introducing additional attributes. We propose to introduce the outlier score and four types of samples (safe, borderline, rare, outlier) as additional attributes in order to gain more information on the data characteristics and improve the classification performance. According to our experimental results, introducing additional attributes can improve the imbalanced classification performance in most cases (6 out of 7 datasets). Further study shows that this performance improvement is mainly contributed by a more accurate classification in the overlapping region of the two classes (majority and minority classes). The proposed idea of introducing additional attributes is simple to implement and can be combined with resampling techniques and other algorithmic-level approaches in the imbalanced learning domain.

Keywords: Class imbalance · Anomaly detection · Borderline samples

1 Introduction

The imbalanced classification problem has caught growing attention from many fields. In the field of computational design optimization, product parameters are modified to generate digital prototypes and the performances are usually evaluated by numerical simulations which often require minutes to hours of computation time. Here, some parameter variations (minority number of designs) would result in valid and producible geometries but violate given constraints in the final steps of the optimization. Under this circumstance, performing proper imbalanced classification algorithms on the design parameters could save computation time. In the imbalanced learning domain, many techniques have proven to be efficient in handling imbalanced datasets, including resampling techniques

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 766186 (ECOLE).

© The Author(s) 2020
T. Bäck et al. (Eds.): PPSN 2020, LNCS 12269, pp. 512–523, 2020.
https://doi.org/10.1007/978-3-030-58112-1_35

and algorithmic-level approaches [5,9,15], where the former aims to produce balanced datasets and the latter aims to make classical classification algorithms appropriate for handling imbalanced datasets. The resampling techniques are standard techniques in imbalance learning since they are simple and easily configurable and can be used in synergy with other learning algorithms [4]. The main idea of most oversampling approaches is to introduce more information to the original dataset by creating synthetic samples. However, very few studies consider the idea of introducing additional attributes to the imbalanced dataset.

The anomaly detection problem can be considered as an extreme case of the class imbalance problem. In this paper, we propose to improve the imbalanced classification with some anomaly detection techniques. We propose to introduce the outlier score, which is an important indicator to evaluate whether a sample is an outlier [2], as an additional attribute of the original imbalanced datasets. Apart from this, we also introduce the four types of samples (safe, borderline, rare and outlier), which have been emphasized in many studies [14,16], as another additional attribute. In our experiments, we consider four scenarios, i.e. four different combinations using the additional attributes and performing resampling techniques. The results of our experiments demonstrate that introducing the two proposed additional attributes can improve the imbalanced classification performance in most cases. Further study shows that this performance improvement is mainly contributed by a more accurate classification in the overlapping region of the two classes (majority and minority classes).

The remainder of this paper is organized as follows. In Sect. 2, the research related to our work is presented, also including the relevant background knowledge on four resampling approaches, outlier score and the four types of samples. In Sect. 3, the experimental setup is introduced in order to understand how the results are generated. Section 4 gives the results and further discussion of our experiments. Section 5 concludes the paper and outlines further research.

2 Related Work

As mentioned in the Introduction, we propose to introduce two additional attributes into the imbalanced datasets in order to gain more information on the data characteristics and improve the classification performance. Introducing additional attributes can be regarded as a data preprocessing method, which is independent of resampling techniques and algorithmic-level approaches, and can also be combined with these two approaches. In this section, the background knowledge related to our experiment is given, including resampling techniques (Sect. 2.1), the definition of four types of samples in the imbalance learning domain (Sect. 2.3) and the outlier score (Sect. 2.2).

2.1 Resampling Techniques

In the following, we introduce two oversampling techniques (SMOTE and ADASYN) and two undersampling techniques (NCL and OSS).

Oversampling Techniques. The synthetic minority oversampling technique (SMOTE) is the most famous resampling technique [3]. SMOTE produces synthetic minority samples based on the randomly chosen minority samples and their K-nearest neighbours. The new synthetic sample can be generated by interpolation between the selected minority sample and one of its K-nearest neighbours. The main improvement in the adaptive synthetic (ADASYN) sampling technique is that the samples which are harder to learn are given higher importance and will be oversampled more often in ADASYN [7].

Undersampling Techniques. One-Sided Selection (OSS) is an undersampling technique which combines Tomek Links and the Condensed Nearest Neighbour (CNN) Rule [4,11]. In OSS, noisy and borderline majority samples are removed with so-called Tomek links [17]. The safe majority samples which have limited contribution for building the decision boundary are then removed with CNN. Neighbourhood Cleaning Rule (NCL) emphasizes the quality of the retained majority class samples after data cleaning [12]. The cleaning process is first performed by removing ambiguous majority samples through Wilson's Edited Nearest Neighbour Rule (ENN) [19]. Then, the majority samples which have different labels from their three nearest neighbours are removed. Apart from this, if a minority sample has different labels from its three nearest neighbours, then the three neighbours are removed.

2.2 Four Types of Samples in the Imbalance Learning Domain

Napierala and Stefanowski proposed to analyse the local characteristics of minority examples by dividing them into four different types: safe, borderline, rare examples and outliers [14]. The identification of the type of an example can be done through modeling its k-neighbourhood. Considering that many applications involve both nominal and continuous attributes, the HVDM metric (Appendix A) is applied to calculate the distance between different examples. Given the number of neighbours k (odd), the label to a minority example can be assigned through the ratio of the number of its minority neighbours to the total number of neighbours ($R_{\frac{min}{all}}$) according to Table 1. The label for a majority example can be assigned in a similar way.

Table 1. Rules to assign the four types of minority examples.

Type	**S**afe (S)	**B**orderline (B)	**R**are (R)	**O**utlier (O)
Rule	$\frac{k+1}{2k} < R_{\frac{min}{all}} \leqslant 1$	$\frac{k-1}{2k} \leqslant R_{\frac{min}{all}} \leqslant \frac{k+1}{2k}$	$0 < R_{\frac{min}{all}} < \frac{k-1}{2k}$	$R_{\frac{min}{all}} = 0$
E.G. given the neighbourhood of a fixed size $k = 5$				
Rule	$\frac{3}{5} < R_{\frac{min}{all}} \leqslant 1$	$\frac{2}{5} \leqslant R_{\frac{min}{all}} \leqslant \frac{3}{5}$	$0 < R_{\frac{min}{all}} < \frac{2}{5}$	$R_{\frac{min}{all}} = 0$

2.3 Outlier Score

Many algorithms have been developed to deal with anomaly detection problems and the experiments in this paper are mainly performed with the nearest-neighbour based local outlier score (LOF). Local outlier factor (LOF), which indicates the degree of a sample being an outlier, was first introduced by Breunig et al. in 2000 [2]. The LOF of an object depends on its relative degree of isolation from its surrounding neighbours. Several definitions are needed to calculate the LOF and are summarized in the following Algorithm 1.

Algorithm 1: Local Outlier Factor (LOF) algorithm [2]

Input : $\mathbf{X}$ - input data $\mathbf{X} = (X_1, ..., X_n)$
n - the number of input examples
k - the number of neighbours
Output: LOF score of every X_i

initialization;
calculate the distance $d(\cdot)$ between every two data points;
for $i = 1$ **to** n **do**
  calculate k-*distance*(X_i): the distance between X_i and its kth neighbour;
  find out k-distance neighbourhood $N_k(X_i)$: the set of data points whose distance from X_i is not greater than k-*distance*(X_i);
  for $j = 1$ **to** n **do**
    calculate reachability distance:

$$reach\text{-}dist_k(X_i, X_j) = \max\{k\text{-}distance(X_j), d(X_i, X_j)\};$$

    calculate local reachability density:

$$\begin{aligned} lrd_k(X_i) &= 1/avg\text{-}reach\text{-}dist_k(X_i) \\ &= 1/\left(\frac{\sum_{o \in N_k(X_i)} reach\text{-}dist_k(X_i, X_j)}{|N_k(X_i)|}\right); \end{aligned}$$

    intuitively, the local reachability density of X_i is the inverse of the average reachability distance based on the k-nearest neighbours of X_i;
    calculate LOF:

$$\begin{aligned} LOF_k(X_i) &= \frac{\sum_{o \in N_k(X_i)} lrd_k(X_j)}{|N_k(X_i)| \cdot lrd_k(X_i)} \\ &= \frac{\sum_{o \in N_k(X_i)} \frac{lrd_k(X_j)}{lrd_k(X_i)}}{|N_k(X_i)|} \end{aligned}$$

    the LOF of X_i is the average local reachability density of X_i's k-nearest neighbours divided by the local reachability density of X_i.
  end
end

According to the definition of LOF, a value of approximately 1 indicates that the local density of data point X_i is similar to its neighbours. A value below 1 indicates that data point X_i locates in a relatively denser area and does not seem to be an anomaly, while a value significantly larger than 1 indicates that data point X_i is alienated from other points, which is most likely an outlier.

3 Experimental Setup

In this paper, we propose to introduce the *four types of samples* and the *outlier score* as additional attributes of the original imbalanced dataset, where the former can be expressed as $R_{\frac{min}{all}}$ (Table 1) and the latter can be calculated through Python library `PyOD` [20].

The experiments reported in this paper are based on 7 two-class imbalanced datasets, including 6 imbalanced benchmark datasets (given in Table 2) and a 2D imbalanced chess dataset, which is commonly used for visualising the effectiveness of the selected techniques in the imbalanced learning domain [4]. Imbalance ratio (IR) is the ratio of the number of majority class samples to the number of minority class samples. For each dataset, we consider four scenarios, whether to perform resampling techniques on the original datasets and whether to perform resampling techniques on the datasets with additional attributes. For each scenario of each dataset, we repeat the experiments 30 times with different random seeds. After that, the paired t-tests were performed on each of the 30 performance metric values to test if there is significant difference between the results of each scenario on a 5% significance level. Each collected dataset is divided into 5 stratified folds (for cross-validation) and only the training set is oversampled, where the stratified fold is to ensure that the imbalance ratio in the training set is consistent with the original dataset and only oversampling the training set is to avoid over-optimism problem [15]. Our code is available on Github (https://github.com/FayKong/PPSN2020) for the convenience of reproducing the main results and figure.

Table 2. Information on benchmark datasets [1].

Datasets	#Attributes	#Samples	Imbalance ratio (IR)
glass1	9	214	1.82
ecoli4	7	336	15.8
vehicle1	18	846	2.9
yeast4	8	1484	28.1
wine quality	11	1599	29.17
page block	10	5472	8.79

In this paper, we evaluate the performance through several different measures, including Area Under the ROC Curve (AUC), precision, recall, F-Measure

(F1) and Geometric mean (Gmean) [13]. These performance measures can be calculated as follows.

$$AUC = \frac{1 + TP_{rate} - FP_{rate}}{2}, \quad TP_{rate} = \frac{TP}{TP + FN}, \quad FP_{rate} = \frac{FP}{FP + TN};$$
$$GM = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{FP + TN}}; \quad FM = \frac{(1 + \beta)^2 \times Recall \times Precision}{\beta^2 \times Recall + Precision}$$
$$Recall = TP_{rate} = \frac{TP}{TP + FN}, \quad Precision = \frac{TP}{TP + FP}, \quad \beta = 1;$$

where TP, FN, FP, TN indicate True Positives, False Negatives, False Negatives and True Negatives in the standard confusion matrix for binary classification.

4 Experimental Results and Discussion

Like other studies [7,13], we also use `SVM` and `Decision Tree` as the base classifiers in our experiments to compare the performance of the proposed method and the existing methods. Please note that we did not tune the hyperparameters for the classification algorithms and the resampling techniques [9]. The experimental results with the two additional attributes (four types of samples and LOF score) are presented in Table 3. We can observe that introducing outlier score and four types of samples as additional attributes can significantly improve the imbalanced classification performance in most cases. For 5 out of 7 datasets (*2D chess dataset, glass1, yeast4, wine quality and page block*), only introducing additional attributes (with no resampling) gives better results than performing resampling techniques.

According to our experimental setup, we notice that introducing the outlier score focuses on dealing with the minority samples since the outlier score indicates the degree of a sample being an outlier. Meanwhile, introducing four types of samples (safe, borderline, rare and outlier) puts emphasis on separating the overlapping region and safe region. The visualisation of different scenarios for the *2D chess* dataset is given in Fig. 1 in order to further study the reason for the performance improvement.

From both the experimental results in Table 3 and the visualisation in Fig. 1, we can conclude that, for the *2D chess* dataset, the experiment with the two additional attributes outperforms the experiment with the classical resampling technique SMOTE. The figure also illustrates that the proposed method has a better ability to handle samples in the overlapping region.

Table 3. Experimental results with `SVM` and `Decision Tree`. "**Add** = YES" means we introduce the two additional attributes to the original datasets; gray cells indicate that the proposed method (**Add** = YES) significantly outperforms the existing methods (**Add** = NO); "—" means that TP + FN = 0 or TP + FP = 0 and the performance metric cannot be computed.

2D chess dataset											
Methods	**Add**	**Decision Tree**					**SVM**				
		AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	NO	0.8482	0.5743	0.6992	0.6208	0.8047	0.8285	—	—	—	—
	YES	0.9771	0.9557	0.9070	0.9226	0.9469	0.9859	0.9846	0.9485	0.9643	0.9723
SMOTE	NO	0.8584	0.6422	0.7102	0.6646	0.8183	0.5921	0.1636	0.5004	0.2437	0.5855
	YES	0.9704	0.9191	0.9061	0.9064	0.9453	0.9933	0.9633	0.9667	0.9622	0.9801
ADASYN	NO	0.8482	0.5743	0.6992	0.6208	0.8047	0.6172	0.1434	0.5904	0.2299	0.5892
	YES	0.9771	0.9557	0.9070	0.9226	0.9469	0.9925	0.8546	0.9667	0.8999	0.9721
NCL	NO	0.5786	0.1245	0.6652	0.2092	0.5541	0.5290	0.1076	0.4212	0.1693	0.4802
	YES	0.9715	0.8542	0.9667	0.8988	0.9716	0.9946	0.9119	0.9667	0.9337	0.9766
OSS	NO	0.7569	0.4197	0.5227	0.4554	0.6813	0.6262	0.3050	0.0295	0.0535	0.0958
	YES	0.9743	0.9321	0.9391	0.9316	0.9640	0.9937	0.9532	0.9564	0.9524	0.9745
glass1 dataset											
Methods	**Add**	**Decision Tree**					**SVM**				
		AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	NO	0.7029	0.6099	0.6235	0.6044	0.6806	0.6779	0.6394	0.5533	0.5828	0.6633
	YES	0.7328	0.6283	0.6344	0.6227	0.6956	0.7779	0.6506	0.65917	0.6430	0.7089
SMOTE	NO	0.7008	0.5750	0.6561	0.6060	0.6782	0.7140	0.5125	0.7236	0.5785	0.6111
	YES	0.7595	0.6393	0.6988	0.6589	0.7273	0.8288	0.6537	0.8802	0.7369	0.7760
ADASYN	NO	0.7095	0.5922	0.6728	0.6187	0.6842	0.7338	0.5159	0.7982	0.6103	0.6271
	YES	0.7799	0.6614	0.7106	0.6780	0.7419	0.8388	0.6545	0.8996	0.7456	0.7845
NCL	NO	0.5926	0.4401	0.9302	0.5843	0.3761	0.6750	0.4124	1.0000	0.5765	0.2177
	YES	0.5897	0.3976	0.9239	0.5527	0.3806	0.7790	0.4299	1.0000	0.5948	0.3403
OSS	NO	0.7010	0.5688	0.6841	0.6132	0.6804	0.6810	0.5850	0.5837	0.5683	0.6444
	YES	0.7611	0.6342	0.7136	0.6637	0.7295	0.7784	0.6085	0.7382	0.6543	0.7128
ecoli4 dataset											
Methods	**Add**	**Decision Tree**					**SVM**				
		AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	NO	0.8446	0.7241	0.6433	0.6432	0.7694	0.9919	0.8889	0.8000	0.7993	0.8797
	YES	0.8525	0.6435	0.6017	0.5734	0.6920	0.9889	0.9143	0.7500	0.7835	0.8512
SMOTE	NO	0.8824	0.7938	0.7233	0.7102	0.8328	0.9894	0.8290	0.8000	0.7268	0.8457
	YES	0.8629	0.8315	0.7300	0.7262	0.8303	0.9931	0.8824	0.9500	0.8881	0.9639
ADASYN	NO	0.8719	0.8407	0.7083	0.7221	0.8236	0.9903	0.7813	0.8000	0.7034	0.8389
	YES	0.8747	0.7833	0.6717	0.6623	0.7822	0.9934	0.8800	0.9500	0.8857	0.9634
NCL	NO	0.8007	0.6080	0.6333	0.5651	0.7380	0.9869	0.8258	0.9000	0.7886	0.8976
	YES	0.8523	0.7297	0.7550	0.6499	0.7982	0.9914	0.8533	0.9500	0.8556	0.9549
OSS	NO	0.8398	0.6284	0.7250	0.5958	0.7872	0.9877	0.8458	0.8133	0.7580	0.8668
	YES	0.9115	0.6858	0.8350	0.6787	0.8586	0.9890	0.8830	0.9117	0.8626	0.9408
vehicle1 dataset											
Methods	**Add**	**Decision Tree**					**SVM**				
		AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	NO	0.6699	0.5018	0.4301	0.4575	0.6004	0.8673	0.7074	0.3593	0.4747	0.5824
	YES	0.7385	0.5855	0.5329	0.5573	0.6794	0.9081	0.6873	0.6266	0.6536	0.7500
SMOTE	NO	0.7241	0.5398	0.5557	0.5458	0.6796	0.8945	0.5538	0.9237	0.6913	0.8264
	YES	0.7403	0.5825	0.5629	0.5704	0.6938	0.9204	0.5808	0.9745	0.7272	0.8582
ADASYN	NO	0.7211	0.5359	0.5570	0.5446	0.6791	0.8995	0.5485	0.9465	0.6937	0.8303
	YES	0.7481	0.5842	0.5789	0.5797	0.7025	0.9206	0.5800	0.9809	0.7284	0.8597
NCL	NO	0.7411	0.4153	0.9506	0.5769	0.7093	0.8411	0.4108	0.9768	0.5776	0.7059
	YES	0.7781	0.4560	0.9392	0.6118	0.7529	0.8752	0.5076	1.0000	0.6728	0.8139
OSS	NO	0.7125	0.4857	0.6066	0.5370	0.6837	0.8702	0.5745	0.7014	0.6293	0.7560
	YES	0.7531	0.5524	0.6286	0.5859	0.7174	0.9062	0.6088	0.9117	0.7290	0.8515
yeast4 dataset											
Methods	**Add**	**Decision Tree**					**SVM**				
		AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	NO	0.6736	0.3619	0.2217	0.2653	0.4482	0.8469	—	—	—	—
	YES	0.8647	0.8320	0.6708	0.7260	0.8132	0.9910	0.8628	0.8036	0.8270	0.8920
SMOTE	NO	0.7320	0.2632	0.4029	0.3082	0.6082	0.9052	0.2112	0.6769	0.3160	0.7773
	YES	0.9115	0.7665	0.6892	0.7171	0.8235	0.9922	0.7096	0.9442	0.8079	0.9639
ADASYN	NO	0.7226	0.2494	0.3958	0.2963	0.6041	0.9011	0.2061	0.6902	0.3104	0.7815
	YES	0.9114	0.7531	0.6553	0.6906	0.8036	0.9923	0.6951	0.9618	0.8051	0.9727
NCL	NO	0.8176	0.1929	0.6819	0.2992	0.7772	0.9063	0.2552	0.5745	0.3516	0.7256
	YES	0.9785	0.6733	0.9772	0.7928	0.9791	0.9917	0.7512	0.9436	0.8337	0.9649
OSS	NO	0.7066	0.2899	0.3561	0.3020	0.5713	0.8488	0.2094	0.0258	0.0447	0.0781
	YES	0.9130	0.7637	0.7699	0.7532	0.8708	0.9892	0.8312	0.8390	0.8310	0.9121
wine quality dataset											
Methods	**Add**	**Decision Tree**					**SVM**				
		AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	NO	0.5844	0.1180	0.1275	0.1132	0.2817	0.9790	0.9653	0.9113	0.9333	0.9525
	YES	0.9790	0.9653	0.9113	0.9333	0.9525	0.9944	0.9636	0.8274	0.8761	0.9031
SMOTE	NO	0.5597	0.0648	0.1801	0.0930	0.3704	0.6935	0.1065	0.4223	0.1680	0.5941
	YES	0.9685	0.9715	0.8630	0.9031	0.9239	0.9942	0.8809	0.9055	0.8890	0.9488
ADASYN	NO	0.5601	0.0654	0.1909	0.0953	0.3800	0.6920	0.1039	0.4231	0.1650	0.5933
	YES	0.9859	0.9709	0.8467	0.8917	0.9141	0.9944	0.8805	0.9055	0.8888	0.9488
NCL	NO	0.5922	0.1037	0.2593	0.1423	0.4817	0.7207	0.2582	0.1891	0.1818	0.3755
	YES	0.9845	0.8567	0.9492	0.8949	0.9703	0.9939	0.9359	0.8818	0.8890	0.9308
OSS	NO	0.5733	0.0729	0.2158	0.1054	0.4135	0.5078	—	—	—	—
	YES	0.9859	0.9636	0.9818	0.9723	0.9901	0.9941	0.9282	0.9424	0.9307	0.9690
page block dataset											
Methods	**Add**	**Decision Tree**					**SVM**				
		AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	NO	0.9083	0.8108	0.7442	0.7687	0.8519	0.9723	0.8743	0.7046	0.7663	0.8304
	YES	0.9369	0.8535	0.8289	0.8350	0.9014	0.9880	0.8481	0.8460	0.8379	0.9091
SMOTE	NO	0.9122	0.7485	0.7910	0.7620	0.8735	0.9646	0.6815	0.8792	0.7536	0.9099
	YES	0.9300	0.8216	0.8404	0.8245	0.9051	0.9847	0.7404	0.9496	0.8251	0.9533
ADASYN	NO	0.9130	0.7302	0.7990	0.7558	0.8763	0.9613	0.5716	0.9277	0.6983	0.9194
	YES	0.9328	0.8452	0.8321	0.8356	0.9032	0.9843	0.7529	0.9726	0.8435	0.9661
NCL	NO	0.9338	0.6528	0.9091	0.7502	0.9223	0.9669	0.6628	0.8960	0.7412	0.9127
	YES	0.9563	0.7318	0.9400	0.8156	0.9474	0.9844	0.7355	0.9606	0.8255	0.9577
OSS	NO	0.9071	0.7297	0.7936	0.7473	0.8711	0.9555	0.8375	0.6755	0.7310	0.8107
	YES	0.9248	0.7820	0.8349	0.7957	0.8972	0.9808	0.7845	0.8655	0.8111	0.9137

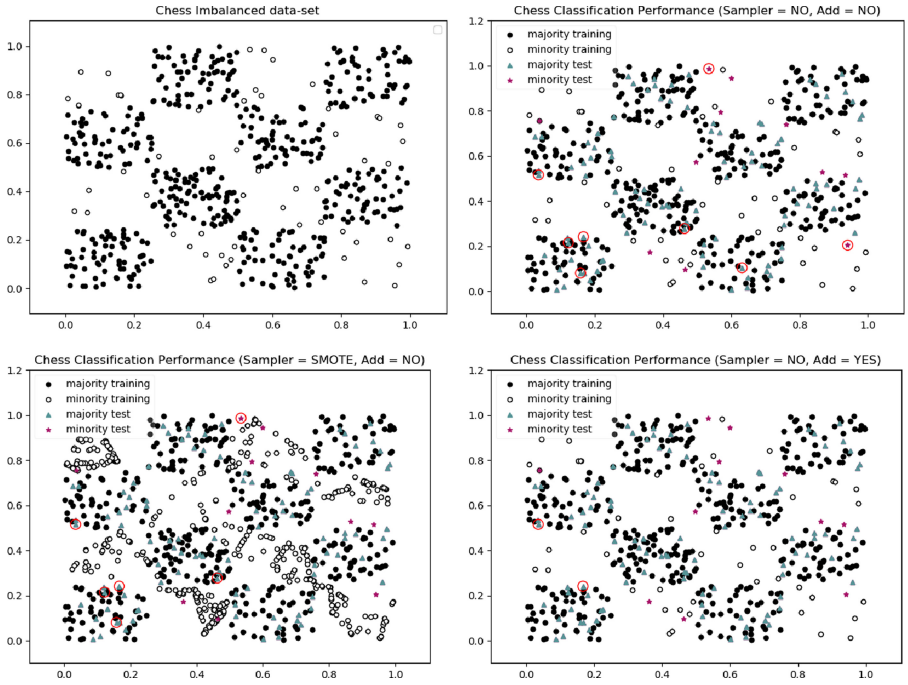


Fig. 1. [top left]. Original imbalanced 2D chess dataset. [top right]. Classification performance for original chess dataset. The red-circled points indicate the misclassified points. [bottom left]. Classification performance for SMOTE-sampled chess dataset. [bottom right]. Classification performance for chess dataset with additional attributes. (Color figure online)

Apart from the visualisation, the feature importance (with `Decision Tree`) is also analysed in order to get an additional insight into the usefulness of the new attributes. Detailed importance score of each attribute is shown in Table 4. According to the feature importance analysis, we can conclude that the introduced "four types of samples" attribute plays an important role in the decision tree classification process for all datasets in our experiment. For 3 out of 7 datasets, the introduced "outlier score" attribute provides useful information during the classification process. The conclusions above show that the two introduced attributes are actually used in the decision process and the "four types of samples" attribute is more important than the "outlier score" attribute.

Table 4. Feature importance analysis with `Decision Tree`. The higher the "score" is, the more the feature contributes during the classification; "org" indicates the original attribute while "add" indicates the added attribute; grey cells indicate the three most useful attributes (after adding the two proposed attributes) in the decision tree classification process.

2D chess dataset (2 original & 2 added attributes)													
Score \ Attr / Add	**org1**	**org2**	**add1**	**add2**	—	—	—	—	—	—	—	—	—
NO	0.4636	0.5364	—	—	—	—	—	—	—	—	—	—	—
YES	0.0101	0.0097	0.8152	0.1650	—	—	—	—	—	—	—	—	—
glass1 dataset (9 original & 2 added attributes)													
Score \ Attr / Add	**org1**	**org2**	**org3**	**org4**	**org5**	**org6**	**org7**	**org8**	**org9**	**add1**	**add2**	—	—
NO	0.2063	0.0213	0.2354	0.1291	0.0302	0.0418	0.2634	0.0000	0.0726	—	—	—	—
YES	0.1770	0.0056	0.1527	0.1099	0.0000	0.0110	0.1892	0.0000	0.0056	0.2413	0.1077	—	—
ecoli4 dataset (7 original & 2 added attributes)													
Score \ Attr / Add	**org1**	**org2**	**org3**	**org4**	**org5**	**org6**	**org7**	**add1**	**add2**	—	—	—	—
NO	0.1093	0.0587	0.0000	0.0000	0.6591	0.1729	0.0000	—	—	—	—	—	—
YES	0.0000	0.0337	0.0000	0.0000	0.6119	0.0000	0.0808	0.1742	0.0994	—	—	—	—
vehicle1 dataset (18 original & 2 added attributes)													
Score \ Attr / Add	**org1**	**org2**	**org3**	**org4**	**org5**	**org6**	**org7**	**org8**	**org9**	**org10**	**org11**	**org12**	**org13**
NO	0.1304	0.0654	0.0892	0.0403	0.0563	0.0233	0.0028	0.0707	0.0000	0.0635	0.0172	0.0416	0.0438
YES	0.0248	0.0216	0.1317	0.0426	0.0227	0.0205	0.0179	0.0024	0.0000	0.0338	0.0260	0.1291	0.0828
Score \ Attr / Add	**org14**	**org15**	**org16**	**org17**	**org18**	**add1**	**add2**	—	—	—	—	—	—
NO	0.0862	0.0414	0.0498	0.0516	0.1265	—	—	—	—	—	—	—	—
YES	0.0146	0.0310	0.0325	0.0291	0.0471	0.2413	0.0485	—	—	—	—	—	—
yeast4 dataset (8 original & 2 added attributes)													
Score \ Attr / Add	**org1**	**org2**	**org3**	**org4**	**org5**	**org6**	**org7**	**org8**	**add1**	**add2**	—	—	—
NO	0.3301	0.2446	0.1839	0.0720	0.0106	0.0000	0.1233	0.0355	—	—	—	—	—
YES	0.0385	0.0297	0.0483	0.0116	0.0000	0.0000	0.0248	0.0053	0.7771	0.0646	—	—	—
wine quality dataset (11 original & 2 added attributes)													
Score \ Attr / Add	**org1**	**org2**	**org3**	**org4**	**org5**	**org6**	**org7**	**org8**	**org9**	**org10**	**org11**	**add1**	**add2**
NO	0.0466	0.1402	0.1215	0.1194	0.0806	0.0635	0.0428	0.1287	0.0483	0.0841	0.1244	—	—
YES	0.0000	0.0098	0.0000	0.0000	0.0000	0.0000	0.0263	0.0000	0.0000	0.0000	0.0000	0.9639	0.0000
page block dataset (10 original & 2 added attributes)													
Score \ Attr / Add	**org1**	**org2**	**org3**	**org4**	**org5**	**org6**	**org7**	**org8**	**org9**	**org10**	**add1**	**add2**	—
NO	0.5452	0.0096	0.0117	0.1899	0.0530	0.0285	0.0983	0.0382	0.0122	0.0134	—	—	—
YES	0.5282	0.0006	0.0036	0.1745	0.0205	0.0223	0.0833	0.0129	0.0007	0.0082	0.1288	0.0164	—

5 Conclusions and Future Research

In this paper, we propose to introduce additional attributes to the original imbalanced datasets in order to improve the classification performance. Two additional attributes, namely four types of samples and outlier score, and the resampling techniques (SMOTE, ADASYN, NCL and OSS) are considered and experimentally tested on seven imbalanced datasets. According to our experimental results, two main conclusions can be derived:

1) In most cases, introducing these two additional attributes can improve the class imbalance classification performance. For some datasets, only introducing additional attributes gives better classification results than only performing resampling techniques.
2) An analysis of the experimental results also illustrates that the proposed method has a better ability to handle samples in the overlapping region.

In this paper, we only validate our idea with four resampling techniques and seven benchmark datasets. As future work, other anomaly detection techniques,

such as the clustering-based local outlier score (CBLOF) [8] and histogram-based outlier score (HBOS) [6] could be included in the analysis. Future work could also consider an extension of this research for engineering datasets [10], especially for the design optimization problems mentioned in our Introduction. Detailed analysis of the feature importance and how the proposed method affects the classification performance in the overlapping region would also be worth studying.

Appendix A

Heterogeneous Value Difference Metric (HVDM)

HVDM is a heterogeneous distance function that returns the distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ [18], where the vectors can involve both nominal and numerical attributes. The HVDM distance can be calculated by [18]:

$$HVDM(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{a=1}^{n} {d_a}^2(x_a, y_a)}, \tag{1}$$

where n is the number of attributes. The function $d_a(\cdot)$ returns the distance between x_a and y_a, where x_a, y_a indicate the ath attribute of vector x and y respectively. It is defined as follows:

$$d_a(x, y) = \begin{cases} 1, & \text{if } x \text{ or } y \text{ is unknown} \\ \text{norm_vdm}_a(x, y), & \text{if } a\text{th attribute is nominal} \\ \text{norm_diff}_a(x, y), & \text{if } a\text{th attribute is continuous} \end{cases} \tag{2}$$

where

$$\text{norm_vdm}_a(x, y) = \sqrt{\sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^2}, \quad \text{norm_diff}_a(x, y) = \frac{|x - y|}{4\sigma_a}, \tag{3}$$

where

- C is the number of total output classes,
- $N_{a,x,c}$ is the number of instances which have value x for the ath attribute and output class c and $N_{a,x} = \sum_{c=1}^{C} N_{a,x,c}$,
- σ_a is the standard deviation of values of the ath attribute.

References

1. Alcalá-Fdez, J., et al.: KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J. Multiple-Valued Log. Soft Comput. **17** (2011)

2. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 93–104 (2000)
3. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
4. Fernández, A., García, S., Galar, M., Prati, R.C., Krawczyk, B., Herrera, F.: Learning from Imbalanced Data Sets. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98074-4
5. Ganganwar, V.: An overview of classification algorithms for imbalanced datasets. Int. J. Emerg. Technol. Adv. Eng. **2**(4), 42–47 (2012)
6. Goldstein, M., Dengel, A.: Histogram-based outlier score (HBOS): a fast unsupervised anomaly detection algorithm. KI-2012: Poster and Demo Track, pp. 59–63 (2012)
7. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322–1328. IEEE (2008)
8. He, Z., Xu, X., Deng, S.: Discovering cluster-based local outliers. Pattern Recogn. Lett. **24**(9–10), 1641–1650 (2003)
9. Kong, J., Kowalczyk, W., Nguyen, D.A., Bäck, T., Menzel, S.: Hyperparameter optimisation for improving classification under class imbalance. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 3072–3078. IEEE (2019)
10. Kong, J., Rios, T., Kowalczyk, W., Menzel, S., Bäck, T.: On the performance of oversampling techniques for class imbalance problems. In: Lauw, H.W., Wong, R.C.-W., Ntoulas, A., Lim, E.-P., Ng, S.-K., Pan, S.J. (eds.) PAKDD 2020. LNCS (LNAI), vol. 12085, pp. 84–96. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-47436-2_7
11. Kubat, M., Matwin, S., et al.: Addressing the curse of imbalanced training sets: one-sided selection. In: ICML, Nashville, USA, vol. 97, pp. 179–186 (1997)
12. Laurikkala, J.: Improving identification of difficult small classes by balancing class distribution. In: Quaglini, S., Barahona, P., Andreassen, S. (eds.) AIME 2001. LNCS (LNAI), vol. 2101, pp. 63–66. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-48229-6_9
13. López, V., Fernández, A., García, S., Palade, V., Herrera, F.: An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics. Inf. Sci. **250**, 113–141 (2013)
14. Napierala, K., Stefanowski, J.: Types of minority class examples and their influence on learning classifiers from imbalanced data. J. Intell. Inf. Syst. **46**(3), 563–597 (2015). https://doi.org/10.1007/s10844-015-0368-1
15. Santos, M.S., Soares, J.P., Abreu, P.H., Araujo, H., Santos, J.: Cross-validation for imbalanced datasets: avoiding overoptimistic and overfitting approaches [research frontier]. IEEE Comput. Intell. Mag. **13**(4), 59–76 (2018)
16. Skryjomski, P., Krawczyk, B.: Influence of minority class instance types on SMOTE imbalanced data oversampling. In: First International Workshop on Learning with Imbalanced Domains: Theory and Applications, pp. 7–21 (2017)
17. Tomek, I.: Two modifications of CNN. IEEE Trans. Syst. Man Cybern. **6**, 769–772 (1976)
18. Wilson, D.R., Martinez, T.R.: Improved heterogeneous distance functions. J. Artif. Intell. Res. **6**, 1–34 (1997)

19. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans. Syst. Man Cybern. **SMC-2**(3), 408–421 (1972)
20. Zhao, Y., Nasrullah, Z., Li, Z.: PyOD: a python toolbox for scalable outlier detection. arXiv preprint arXiv:1901.01588 (2019)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

