Learning class-imbalanced problems from the perspective of data intrinsic characteristics

Kong, J.

CHAPTER 5

Improving Imbalanced Classification via Adding Additional Attributes

The anomaly detection problem can be considered as an extreme case of class imbalance problem, however, 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. In this chapter, we introduce our proposed idea on improving imbalanced classification via adding additional attributes. First, Section 5.1 shows the motivation and provides a brief introduction on our work. After that, in Section 5.2, the background knowledge on anomaly detection and four types of samples in imbalanced datasets are presented. In Section 5.3, the information on the datasets, the experimental setup as well as the experimental results and discussion are introduced. Section 5.4 concludes the chapter and outlines the further work.

5.1 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 and algorithmic-level approaches (Ganganwar, 2012; Kong, Kowalczyk, D. A. Nguyen, Bäck, and Menzel, 2019; M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018), 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 (Fernández, García, Galar, Prati, Krawczyk, and Herrera, 2018). 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 chapter, 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 (Breunig, Kriegel, R. T. Ng, and Sander, 2000), 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 (Napierala and Stefanowski, 2016; Skryjomski and Krawczyk, 2017), as another additional attribute. The paper contributed to this chapter has been published in *Parallel Problem Solving from Nature–PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part I 2020 Aug 31 (pp. 512-523)*, titled "*Improving imbalanced classification by anomaly detection*". 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).

5.2 Related Works

This section first introduces the resampling techniques used in this chapter. Then, the anomaly detection problem is introduced.

5.2.1 Resampling Techniques

This work is based on five resampling techniques in our experiments, one oversampling, two undersampling and two hybrid approaches. The oversampling technique SMOTE and the two hybrid approaches, SMOTETL and SMOTEENN; have been introduced in detail in the previous chapters. Therefore, we only provide details on the two undersampling approaches, OSS and NCL.

OSS

One-Sided Selection (OSS) (Kubat, Matwin, et al., 1997) is an undersampling technique which combines Tomek Links and the Condensed Nearest Neighbour (CNN) Rule. Detailed information on Tomek Links is given in 4.2.1. CNN was first introduced by Hart in 1968 (Hart, 1968) together with the concept of a consistent subset. By definition, a subset $\hat{E}$ is consistent with E ($\hat{E} \subseteq E$), if the 1-NN rule (K-NN rule, where $K = 1$) built with samples in $\hat{E}$ can correctly classify samples in E. In OSS, the following three groups of samples are removed (Kubat, Matwin, et al., 1997):

- Majority class samples which suffer from class-label noise.
- Majority class samples which are close to the decision boundary. They are susceptible to variations, and even a tiny variation in training data or classification model can make them fall on the wrong side of the decision boundary.
- Majority class samples which have limited contribution for building the decision boundary. Although they are harmless but they increase the classification costs.

The first two groups of samples are removed with so-called Tomek links. The third group of samples are removed with CNN. The remainder of the majority class samples and all the minority class samples are used to construct the classifiers. Algorithm 1 summarizes the OSS procedure.

Algorithm 1: One-Side Selection (OSS) (Kubat, Matwin, et al., 1997)

Input : S - Original training set
Output : T - Undersampled training set

1 Select a subset C ($C \subseteq S$), which contains all minority class samples and one randomly selected majority class sample;
2 Classify S using the 1-NN rule built with C. Add all misclassified samples in S to subset C and now C is a consistent subset of S;
3 Remove from C all majority class samples belonging to Tomek links. The remaining set is referred to as T.

NCL

Neighbourhood Cleaning Rule (NCL) (Laurikkala, 2001) emphasizes the quality of the retained samples after data cleaning and can be used for multi-class problems. Suppose C are the classes of interest, and the rest of the data are referred as R. The cleaning process is first performed by removing any ambiguous sample in R whose label differs from the class of at least two of its three neighbours through the Wilson's Edited Nearest Neighbours (ENN, introduced in 4.2.1) (D. L. Wilson, 1972). In addition, NCL performs a deeper cleaning in the neighbourhoods of samples in C. For a sample in C, if its label differs from the classification given by its three nearest neighbours, the neighbours belonging to R are removed. In this step, special considerations are paid to small-size classes (details in Algorithm 2). In the binary scenario, NCL can be described as follows: if a majority class sample has a different label from the classification given by its three nearest neighbours, this majority class sample is removed. Additionally, if the label of a minority class sample contradicts the classification given by its three nearest neighbours, then the neighbours belonging to the majority class are removed.

5.2.2 Anomaly Detection

Anomaly detection, also referred to as outlier detection, is the process of identifying irregular patterns in the datasets (Chandola, Banerjee, and Kumar, 2009). The behaviours of these patterns deviate significantly from the majority of the data. Such examples can be found in various applications, including fraud detection in credit cards, medical diagnosis in health care, quality control in the manufacturing field, etc.

Many algorithms have been developed to deal with anomaly detection problems

Algorithm 2: Neighbourhood Cleaning Rule (NCL) (Laurikkala, 2001)

Input : S - Original training set
Output : T - Undersampled training set

Split training set S into the classes of interest C and the rest R;
Identify the noisy data D_1 in R with ENN;
Identify the samples in C which are misclassified by their 3 nearest neighbours and referred to as C_m;
for *each class* $R_i \in R$ **do**
    if $x \in Ri \cap C_m$ *and* $|R_i| \geqslant \frac{1}{2} \times |C|$ **then**
        move x into D_1;
end
Remove D_1 from S and the undersampled training set is $T = S - D_1$.

and the experiments in this chapter 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 in (Breunig, Kriegel, R. T. Ng, and Sander, 2000). 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 3.

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.

5.2.3 Four Types of Samples in Imbalanced Datasets

Napierala and Stefanowski proposed to analyse the local characteristics of minority class samples by dividing them into four different types: *safe, borderline, rare* samples and *outliers* (Napierala and Stefanowski, 2016). The idea has been introduced in detail in Section 2.3.2.

Algorithm 3: Local Outlier Factor (LOF) algorithm (Breunig, Kriegel, R. T. Ng, and Sander, 2000)

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 $k\text{-}distance(x_i)$: 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 $k\text{-}distance(x_i)$;
  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

5.3 Experiments

5.3.1 Information on the Datasets

The experiments reported in this chapter are based on 7 two-class imbalanced datasets, including 6 imbalanced benchmark datasets (given in Table 5.1) and a 2D imbalanced chess dataset, which is commonly used for visualising the effectiveness of the selected techniques in the imbalanced learning domain (Fernández, García, Galar, Prati, Krawczyk, and Herrera, 2018). Figure 5.1 shows the 2D imbalanced chess dataset.

Table 5.1: Information on benchmark datasets (Alcalá-Fdez, Fernández, Luengo, Derrac, García, Sánchez, and Herrera, 2011).

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

5.3.2 Experimental Setup

In this chapter, we propose introducing the *outlier score* and the *four types of samples* as additional attributes of the original imbalanced dataset. The LOF algorithm is an unsupervised anomaly detection method which computes the local density deviation of a given data point relative to its neighbours. Hence, calculating the *outlier score* does not require the information of class labels on either training or test samples. In our experiments, we calculate the LOF values for all samples (before splitting the training and test set). The Python library `PyOD` (Y. Zhao, Nasrullah, and Z. Li, 2019) is used directly to calculate the LOF values. Unlike computing LOF values, computing different types of samples requires the information of class labels, see Table 2.3. However, the labels of test samples should be assumed unknown in

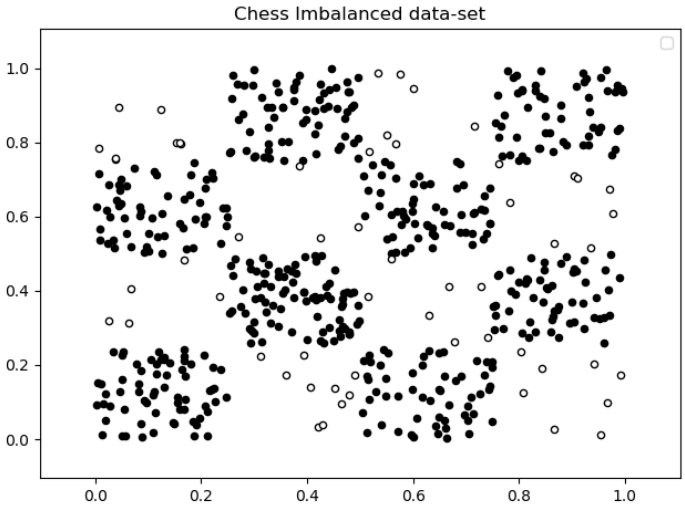


Figure 5.1: Original imbalanced 2D chess dataset. Black points indicate the majority class samples and white points indicate minority class samples.

the training process. Therefore, we use the steps below to add *four types of samples* as an additional attribute "type".

1. Split the data into training and test set.

2. Compute the types for samples in training set. We use positive numbers ($R_{\frac{min}{all}}$) to indicate the types of minority class samples, and negative numbers ($-R_{\frac{maj}{all}}$) for types of majority class samples. For example, for a safe minority class sample with $R_{\frac{min}{all}} = 1$, we add "1" as the "type" value. For a borderline majority class sample with $R_{\frac{maj}{all}} = \frac{3}{5}$, we add "$-\frac{3}{5}$" as the "type" value.

3. Treat training set and test set as a whole. Then, given the number of neighbours k, for each sample in the test set, find the k nearest neighbours belonging to the training set.

4. For a sample in the test set, average the "type" values of its k nearest neighbours belonging to the training set. The average is the "type" value for this sample.

Each dataset was experimented with five resampling techniques and our proposed method. For each method 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. We perform 5-fold stratified cross validation in our experiments.

5.3.3 Experimental Results and Discussion

Like other studies (H. He, Bai, E. A. Garcia, and S. Li, 2008; López, Fernández, García, Palade, and Herrera, 2013), 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. Our purpose in this chapter is not to achieve the best performance of a certain method under fine hyperparameter tuning. Hence, we did not tune the hyperparameters for the classification algorithms and the resampling techniques (Kong, Kowalczyk, D. A. Nguyen, Bäck, and Menzel, 2019). The experimental results with the two additional attributes (four types of samples and LOF score) are presented in Table 5.2, 5.3, 5.4 and 5.5. Before discussing the experimental results, it is worth mentioning that NCL will not be effective if no samples meet the removal conditions. In this case, NCL will produce the same results as dealing with the original dataset, i.e. row “None" and row “NCL" can be exactly the same in the tables. The tables contain much information, and we will discuss them separately below.

- Scenarios where adding additional attributes performs significantly better than resampling techniques:
  - *2D chess* dataset with `SVM`;
  - *glass1* dataset with `SVM`;
  - *ecoli4* dataset with `Decision Tree`.
- Scenarios where adding additional attributes produces competitive performances to resampling techniques:
  - *vehicle1* dataset with `SVM`;
  - *yeast4* dataset with `Decision Tree` and `SVM`;
  - *wine quality* dataset with `Decision Tree` and `SVM`.
- Scenarios where both resampling techniques and our proposed method do not improve the imbalanced classification performances:

  - *glass1* dataset with `Decision Tree`;
  - *ecoli4* dataset with `SVM`;
  - *page block* dataset with `Decision Tree` and `SVM`.

- Scenarios where adding additional attributes degrades the imbalanced classification performance:
  - *2D chess* dataset with `Decision Tree`;
  - *vehicle1* dataset with `Decision Tree`.

We conclude that in most cases, adding additional attributes produces significantly better or competitive classification performance, except for two scenarios *2D chess* dataset with `Decision Tree` and *vehicle1* dataset with `Decision Tree`. Further feature importance analysis shows that due to the high correlation between the added “type" attribute and class labels, `Decision Tree` uses only the added “type" attribute for classification when dealing with these two datasets. This results in the degradation of these two scenarios. Hence, it is recommended to implement the proposed method with feature-insensitive classifiers.

According to our experimental setup, we notice that introducing the local outlier factor focuses on dealing with the minority samples since the local outlier factor 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 with `SVM` is given in Figure 5.2 in order to further study the reason for the performance improvement.

From both the experimental results and the visualisation in Figure 5.2, 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.

5.4 Conclusions and Future Work

In this chapter, we propose to introduce additional attributes to the original imbalanced datasets in order to improve the classification performance. Two

Table 5.2: Experimental results on *2D chess* and *glass1* datasets with `SVM` and `Decision Tree`.
Row "**Add**" indicates our proposed methods. Bold numbers indicate that the performance is statistically better than the unbold ones. We only bold up to two statistically best results; if more than two competitive results exist, no numbers will be bolded. "—" means that TP+FN=0 or TP+FP=0 and the performance metric cannot be computed.

2D chess dataset										
Methods	**Decision Tree**					**SVM**				
	AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	0.8232	0.7775	0.5324	0.6203	0.7131	0.8284	—	—	—	—
SMOTE	**0.8584**	0.6422	**0.7102**	**0.6646**	**0.8183**	0.5848	0.1564	0.4847	0.2340	0.5743
NCL	0.8232	0.7775	0.5324	0.6203	0.7131	0.8284	—	—	—	—
OSS	0.7569	0.4197	0.5227	0.4554	0.6813	0.6385	0.2611	0.0266	0.0479	0.08478
SMOTEENN	0.8135	0.5519	0.6534	0.5885	0.7763	0.5932	0.1503	**0.5371**	0.2331	**0.5846**
SMOTETL	**0.8586**	0.6581	**0.7018**	**0.6696**	**0.8150**	0.5818	0.1644	0.4824	0.2414	0.5777
Add	0.6033	**0.9000**	0.2106	0.3374	0.4479	**0.8422**	**0.8533**	0.3333	**0.4750**	0.5673
glass1 dataset										
Methods	**Decision Tree**					**SVM**				
	AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	0.6987	0.6056	0.6199	0.5998	0.6764	0.6765	**0.6351**	0.5400	0.5735	**0.6554**
SMOTE	0.6973	0.5650	0.6455	0.5954	0.6700	0.7159	0.5104	0.7255	0.5787	0.6097
NCL	0.7050	0.5850	0.6635	0.6087	0.6772	0.6887	0.5983	0.5800	0.5783	**0.6572**
OSS	0.6885	0.5532	0.6707	0.5999	0.6701	0.6779	0.5882	0.5734	0.5639	0.6397
SMOTEENN	0.6812	0.5407	**0.6905**	0.5955	0.6578	0.7239	0.5098	**0.7596**	**0.5834**	0.5943
SMOTETL	0.6942	0.5793	0.6401	0.5995	0.6723	0.7097	0.5071	0.7192	0.5748	0.6064
Add	0.6954	**0.6633**	0.6050	**0.6139**	0.6773	**0.7885**	—	—	—	—

Table 5.3: Experimental results on *ecoli4* and *vehicle1* datasets with `SVM` and `Decision Tree`.
Row "**Add**" indicates our proposed methods. Bold numbers indicate that the performance is statistically better than the unbold ones. We only bold up to two statistically best results; if more than two competitive results exist, no numbers will be bolded. "—" means that TP+FN=0 or TP+FP=0 and the performance metric cannot be computed.

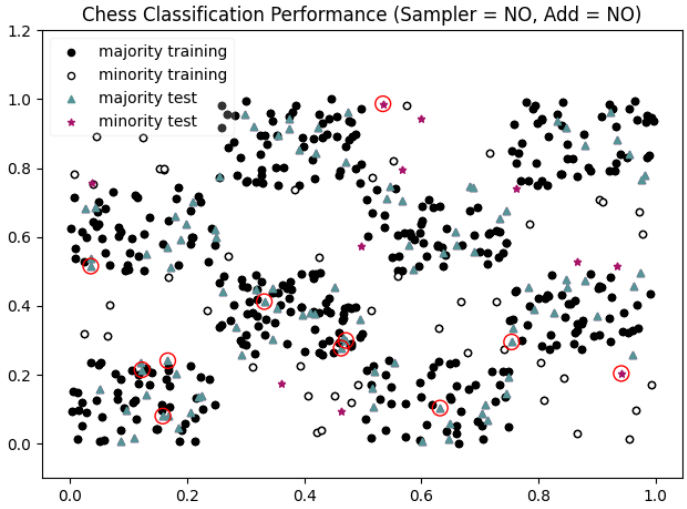
ecoli4 dataset										
Methods	**Decision Tree**					**SVM**				
	AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	0.8446	0.7241	0.6433	0.6432	0.7694	0.9919	**0.8889**	0.8000	0.7993	0.8797
SMOTE	0.8803	0.7894	0.7233	0.7080	0.8323	0.9898	0.8299	0.8000	0.7281	0.8470
NCL	0.8446	0.7241	0.6433	0.6432	0.7694	0.9919	**0.8889**	0.8000	0.7993	0.8797
OSS	0.8398	0.6276	0.7267	0.5827	0.7788	0.9867	0.8389	0.8133	0.7504	0.8648
SMOTEENN	0.8105	0.7115	0.7133	0.6251	0.7626	0.9908	0.8238	0.8400	0.7492	0.8585
SMOTETL	0.8685	0.7867	0.7083	0.7001	0.8247	0.9896	0.8295	0.8000	0.7275	0.8465
Add	**0.9058**	**0.8571**	**0.8500**	**0.8032**	**0.9031**	0.9439	0.8571	**0.8500**	0.8032	**0.9031**
vehicle1 dataset										
Methods	**Decision Tree**					**SVM**				
	AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	0.6699	0.5018	0.4301	0.4575	0.6004	0.8673	**0.7074**	0.3593	0.4747	0.5824
SMOTE	0.7172	0.5293	0.5407	0.5323	0.6687	0.8950	0.5560	0.9269	0.6940	0.8287
NCL	0.7298	**0.5573**	0.5734	0.5635	0.6932	0.8617	0.6063	0.5623	0.5808	0.6987
OSS	0.7055	0.4717	0.5952	0.5241	0.6739	0.8698	0.5747	0.6961	0.6271	0.7537
SMOTEENN	**0.7624**	0.5004	**0.7521**	**0.5993**	**0.7436**	0.8654	0.4949	**0.9388**	0.6467	0.7890
SMOTETL	0.7169	0.5331	0.5377	0.5333	0.6686	0.8945	0.5548	0.9226	0.6919	0.8268
Add	0.6755	0.5210	0.4973	0.5065	0.6440	0.8935	0.5584	0.9077	0.6902	0.8239

Table 5.4: Experimental results on *yeast4* and *wine quality* datasets with `SVM` and `Decision Tree`.
Row "**Add**" indicates our proposed methods. Bold numbers indicate that the performance is statistically better than the unbold ones. We only bold up to two statistically best results; if more than two competitive results exist, no numbers will be bolded. "—" means that TP+FN=0 or TP+FP=0 and the performance metric cannot be computed.

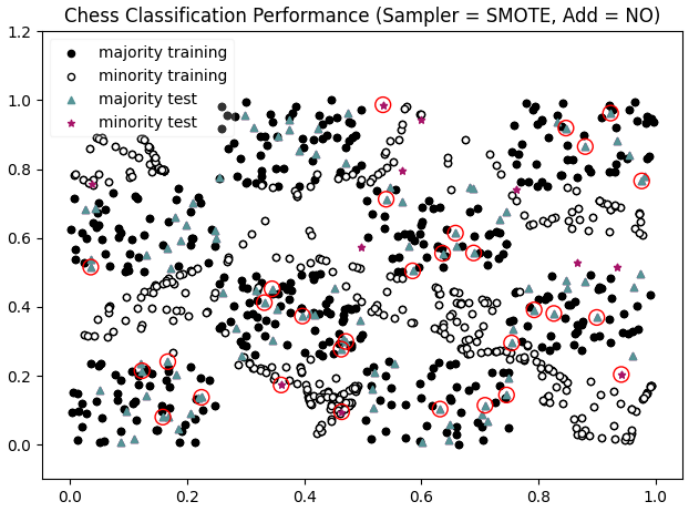
yeast4 dataset										
Methods	**Decision Tree**					**SVM**				
	AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	0.6845	**0.2959**	0.2245	0.2465	0.4503	0.8481	—	—	—	—
SMOTE	0.7338	0.2598	0.3968	0.3026	0.6042	0.9033	0.2170	0.6693	0.3182	0.7735
NCL	0.6845	**0.2959**	0.2245	0.2465	0.4503	0.8481	—	—	—	—
OSS	0.7050	0.2838	0.3727	0.3044	0.5827	0.8452	**0.2794**	0.0310	0.0551	0.0961
SMOTEENN	**0.7663**	0.2170	**0.5552**	0.3071	**0.7101**	0.9083	0.1937	**0.7247**	0.2982	**0.7953**
SMOTETL	0.7258	0.2769	0.4042	**0.3166**	0.6112	0.9046	0.2166	0.6674	**0.3174**	0.7724
Add	0.7388	0.2610	0.3709	0.2990	0.5891	0.8977	0.2050	0.6491	0.3009	0.7589
wine quality dataset										
Methods	**Decision Tree**					**SVM**				
	AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	0.5760	**0.1739**	0.1292	0.1283	0.2910	0.6415	—	—	—	—
SMOTE	0.5539	0.0698	0.1720	0.0970	0.3627	0.6934	**0.1180**	0.4292	**0.1825**	0.5948
NCL	0.5760	**0.1739**	0.1292	0.1283	0.2910	0.6415	—	—	—	—
OSS	0.5534	0.0796	0.1735	0.1035	0.3610	0.4466	—	—	—	—
SMOTEENN	**0.6043**	0.0902	**0.3413**	**0.1404**	**0.5262**	0.6995	0.1006	**0.4722**	0.1642	0.6097
SMOTETL	0.5545	0.0709	0.1793	0.0991	0.3669	0.6942	**0.1169**	0.4253	**0.1811**	0.5926
Add	**0.6126**	0.0910	**0.3418**	**0.1429**	**0.5320**	0.7007	—	—	—	—

Table 5.5: Experimental results on *page block* dataset with `SVM` and `Decision Tree`.
Row "**Add**" indicates our proposed methods. Bold numbers indicate that the performance is statistically better than the unbold ones. We only bold up to two statistically best results; if more than two competitive results exist, no numbers will be bolded. "—" means that TP+FN=0 or TP+FP=0 and the performance metric cannot be computed.

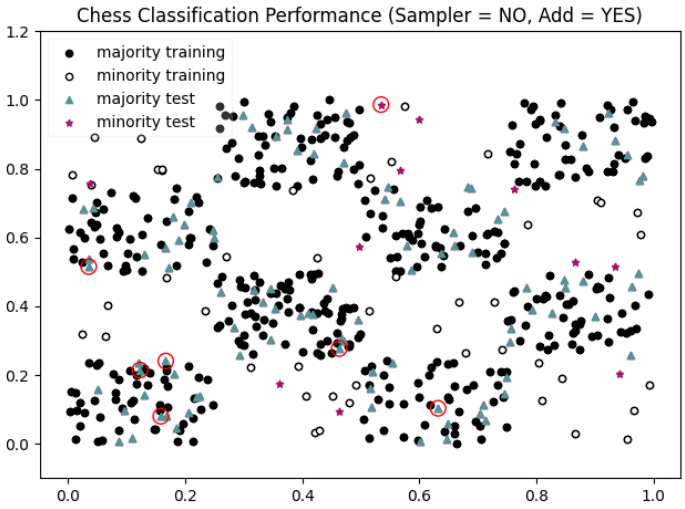
page block dataset										
Methods	**Decision Tree**					**SVM**				
	AUC	**Precision**	**Recall**	**F1**	**Gmean**	**AUC**	**Precision**	**Recall**	**F1**	**Gmean**
NONE	0.9104	0.7840	0.7340	0.7498	0.8451	0.9753	0.8625	0.7025	0.7596	0.8287
SMOTE	0.9198	0.7434	0.8150	0.7708	0.8871	0.9728	0.6621	0.9057	0.7533	0.9239
NCL	0.9163	0.7978	0.7884	0.7857	0.8760	0.9728	0.8471	0.7182	0.7606	0.8364
OSS	0.9228	0.7203	0.8092	0.7514	0.8810	0.9552	0.8280	0.6649	0.7208	0.8044
SMOTEENN	0.9280	0.6994	0.8744	0.7692	0.9137	0.9721	0.6384	0.9120	0.7409	0.9246
SMOTETL	0.9209	0.7425	0.8220	0.7737	0.8908	0.9729	0.6620	0.9061	0.7530	0.9239
Add	0.8491	0.8269	0.7126	0.7559	0.8349	0.9709	0.5669	0.9537	0.7027	0.9340



(a) Classification performance for original chess dataset with SVM.



(b) Classification performance for SMOTE-sampled chess dataset with SVM.



(c) Classification performance for chess dataset with additional attributes with SVM.

Figure 5.2: Classification performance for chess dataset under different scenarios. The red-circled points indicate the misclassified points.

additional attributes, namely four types of samples and outlier score, and the resampling techniques (SMOTE, NCL, OSS, SMOTEENN and SMOTETL) 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 or produce competitive class imbalance classification performance. For some datasets, only introducing additional attributes gives better classification results than only performing resampling techniques.

2. The proposed additional attribute “type" is highly correlated with class labels in some datasets. Hence, it is recommended to implement the proposed method with feature-insensitive classifiers.

3. 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 chapter, we only validate our newly proposed method with five resampling techniques and seven benchmark datasets. As future work, other

anomaly detection techniques, such as the clustering-based local outlier score (CBLOF) (Z. He, Xu, and Deng, 2003) and histogram-based outlier score (HBOS) (Goldstein and Dengel, 2012) could be included in the analysis. Future work could also consider an extension of this research for engineering datasets (Kong, Rios, Kowalczyk, Menzel, and Bäck, 2020a), 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.