Universiteit
Leiden
The Netherlands

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

Kong, J.

Citation
Kong, J. (2023, September 27). *Learning class-imbalanced problems from the perspective of data intrinsic characteristics*. Retrieved from https://hdl.handle.net/1887/3642254

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/3642254

Note: To cite this publication please use the final published version (if applicable).

CHAPTER 3

An Empirical Investigation Comparing Several Oversampling Techniques

Many resampling approaches have been developed in the imbalance learning domain, most empirical studies and application work are still based on the “classical" resampling techniques and do not take newly developed resampling techniques into account. In this chapter, we investigate the effectiveness of six oversampling techniques (both “classical" and new ones) and study the relationship between data complexity measures and the choice of oversampling techniques. This chapter is structured as follows. First, Section 3.1 briefly introduces our work in this chapter. Then, in Section 3.2, the research related to our work is presented including the relevant background knowledge on six resampling approaches and data complexity measures. In Section 3.3, the experiments, including introduction on the datasets, cross-validation and experimental setup are introduced. Section 3.3 also contains the results and discussions of our experiments. Further exploration through data from a real-world inspired digital vehicle model is presented in Section 3.4. Section 3.5 concludes the chapter and outlines further research.

3.1 Introduction

The classification problem under class imbalance has caught growing attention from both, academic and industrial field. Due to recent advances, the progress in technical assets for data storage and management as well as in data science enables

practitioners from industry and engineering to collect a large amount of data with the purpose of extracting knowledge and acquire hidden insights. An example may be illustrated from the field of computational design optimization where product parameters are modified to generate digital prototypes which performances are evaluated by numerical simulations, or based on equations that express human heuristics and preferences. Here, many parameter variations usually result in valid and producible geometries but in the final steps of the optimization, i.e. in the area where the design parameters converge to a local/global optimum, some geometries are generated which violate given constraints. Under this circumstance, a database would contain a large number of designs which are according to specifications (even if some may be of low performance) and a smaller number of designs which eventually violate pre-defined product requirements. By far, the resampling techniques have proven to be effective in handling imbalanced benchmark datasets (López, Fernández, García, Palade, and Herrera, 2013). However, the empirical study and application work in the imbalanced learning domain are mostly focusing on “classical" resampling techniques like SMOTE, ADASYN, and MWMOTE etc (J. Li, L.-s. Liu, Fong, R. K. Wong, Mohammed, Fiaidhi, Sung, and K. K. Wong, 2017; Luengo, Fernández, García, and Herrera, 2011; M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018), although there are many recently developed resampling techniques.

In this chapter, we set up several experiments on 19 benchmark datasets to study the effectiveness of six oversampling techniques (Kong, Rios, Kowalczyk, Menzel, and Bäck, 2020b), including SMOTE, ADASYN, MWMOTE, RACOG, wRACOG and RWO-Sampling. For each data set, we also compute seven data complexity measures to investigate the relationship between data complexity measures and the choice of resampling techniques, since researchers have pointed out that studying the data complexity of imbalanced datasets is of vital importance (Luengo, Fernández, García, and Herrera, 2011) and it may affect the choice of resampling techniques (M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018). We also perform experiments on a real-world inspired vehicle dataset. Results of our experiments demonstrate that in most cases oversampling techniques that take into account the minority class distribution (RACOG, wRACOG, RWO-Sampling) perform better and RACOG exhibits the best performance among the six reviewed approaches. Results on our real-world inspired vehicle dataset further validate this conclusion. No obvious relationship between data complexity measures and the

choice of resampling techniques is found in our experiments. However, we find that the F1v value, a measure for evaluating the overlap between classes which most researchers ignore (Luengo, Fernández, García, and Herrera, 2011; M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018), has a strong negative correlation with the potential after-sampled Area Under curve (AUC) value.

3.2 Related Work

Many effective sampling approaches have been developed in the imbalanced learning domain and the synthetic minority oversampling technique (SMOTE) is the most famous one among all. Currently, more than 90 SMOTE extensions have been published in scientific journals and conferences (Fernández, García, Galar, Prati, Krawczyk, and Herrera, 2018). Most of the review papers and applications are based on the “classical" resampling techniques and do not take new oversampling techniques into account. In this chapter, we briefly review six oversampling approaches, including both, “classical" ones (SMOTE, ADASYN, MWMOTE) and new ones (RACOG, wRACOG, RWO-Sampling) (Barua, Islam, Yao, and Murase, 2012; Chawla, Bowyer, Hall, and Kegelmeyer, 2002; Das, Krishnan, and Cook, 2014; H. He, Bai, E. A. Garcia, and S. Li, 2008; H. Zhang and M. Li, 2014). The six reviewed oversampling techniques can be divided into two groups according to whether they consider the overall minority class distribution. Among the six approaches, RACOG, wRACOG, and RWO-Sampling take into account the overall minority class distribution while the other three do not. Apart from developing new approaches to solve the class-imbalance problem, various studies have pointed out that it is important to study the characteristics of the imbalanced dataset (López, Fernández, García, Palade, and Herrera, 2013; M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018). In (López, Fernández, García, Palade, and Herrera, 2013), authors emphasize the importance of studying the overlap between the two-class samples. In (M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018), authors set up several experiments with the KEEL benchmark datasets (Alcalá-Fdez, Fernández, Luengo, Derrac, García, Sánchez, and Herrera, 2011) to study the relationship between various data complexity measures and the potential AUC value. It is also pointed out in (M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018) that the distinctive inner procedures of oversampling approaches are suitable for particular characteristics of the data. Hence, apart from evaluating

the effectiveness of the six reviewed oversampling approaches, we also aim to investigate the relationship between data complexity measures and the choice of resampling techniques.

3.2.1 Oversampling Techniques

As mentioned above, we investigate six oversampling techniques in two groups: “classical" ones (SMOTE, ADASYN, MWMOTE) and “new" ones (RACOG, wRACOG, RWO-sampling), depending on whether they consider the overall minority class distribution. SMOTE has been introduced in Section 2 and in the following, the remaining five oversampling techniques are introduced.

ADASYN

The adaptive synthetic (ADASYN) sampling technique aims to adaptively generate minority class samples according to their distributions (H. He, Bai, E. A. Garcia, and S. Li, 2008). The samples which are harder to learn are given higher importance and will be oversampled more often in the data generation process. The key point is to determine a weight/sampling importance ($\hat{r}_i$) for each minority class sample. Weight $\hat{r}_i$ of a minority class sample $\boldsymbol{x_i}$ is defined as (H. He, Bai, E. A. Garcia, and S. Li, 2008)

$$\hat{r}_i = \frac{r_i}{\sum_{i=1}^{m_s} r_i}, \quad r_i = \frac{\Delta_i}{K}, \quad i = 1, ..., m_s, \tag{3.1}$$

where m_s is the number of minority class samples, Δ_i is the number of samples in the K Nearest Neighbours (K-NN) of $\boldsymbol{x_i}$ that belong to the majority class. For a specific minority class sample, a higher value of r_i corresponds to a higher difficulty to learn. The number of synthetic samples that will be generated for different minority class samples are proportional to their sampling importance (H. He, Bai, E. A. Garcia, and S. Li, 2008)

$$g_i = \hat{r}_i \cdot G, \tag{3.2}$$

where G is the total number of synthetic minority class samples that need to be produced. Figure 3.1 shows an example of the sampling importance for different minority class samples.

Compared to SMOTE, the only difference in ADASYN oversampling procedure is that more synthetic samples will be generated for harder minority class samples.

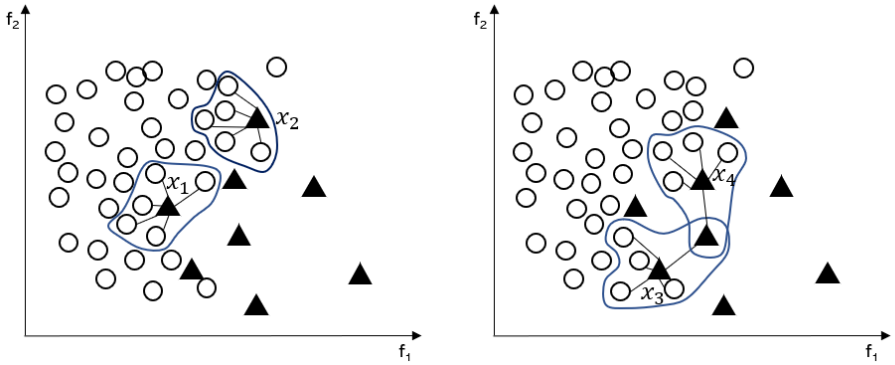

Figure 3.1: Example of sampling importance for different minority class samples. According to definition, $r_1 = r_2 = 1, r_3 = r_4 = 0.8$ and $\hat{r}_1 = \hat{r}_2 > \hat{r}_3 = \hat{r}_4$, indicating the sampling importance of sample $\boldsymbol{x_1}, \boldsymbol{x_2}$ is higher than $\boldsymbol{x_3}, \boldsymbol{x_4}$ and more synthetic samples will be produced for $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$.

In this way, the ADASYN not only provides less learning bias but puts more focus on the difficulty to learn minority class samples.

MWMOTE

Compared to other oversampling techniques, the majority weighted minority oversampling technique (MWMOTE) improves the sample selection scheme and the synthetic sample generation scheme (Barua, Islam, Yao, and Murase, 2012). MWMOTE first finds the informative minority class samples (S_{imin}) by removing the "noise" minority class samples and finding the borderline majority class samples. Then, every sample in S_{imin} is given a selection weight (S_w), according to the distance to the decision boundary, the sparsity of the located minority class cluster and the sparsity of the nearest majority class cluster. These weights are converted into the selection probability (S_p), which will be used in the synthetic sample generation stage. Different from the K-NN-based approach, MWMOTE adopts a clustering algorithm to generate the synthetic samples. The cluster-based synthetic sample generation process proposed in MWMOTE can be described as, 1) cluster all minority class samples into M clusters; 2) select a minority class sample x from S_{imin} according to S_p and randomly select another sample y from the same cluster of x; 3) use the same equation (Eq. 2.1.1) employed in the K-NN-based approach to generate the synthetic sample; 4) repeat 1) – 3) until the required number of

synthetic samples is generated.

RACOG and wRACOG

The oversampling approaches can effectively increase the number of minority class samples and achieve a balanced training dataset for classifiers. However, the oversampling approaches introduced above heavily rely on *local* information of the minority class samples and do not take the overall distribution of the minority class into account. Hence, the *global* information of the minority class samples cannot be guaranteed. In order to tackle this problem, Das et al. (Das, Krishnan, and Cook, 2014) proposed RACOG (RApidly COnverging Gibbs) and wRACOG (Wrapper-based RApidly COnverging Gibbs).

In both algorithms, the *n*-dimensional probability distribution of the minority class is optimally approximated by Chow-Liu's dependence tree algorithm and the synthetic samples are generated from the approximated distribution using Gibbs sampling (Das, Krishnan, and Cook, 2014). The minority class data points are chosen as initial values to start the Gibbs sampler. Instead of running an "exhausting" long Markov chain, the two algorithms produce multiple relatively short Markov chains, each starting with a different minority class sample. RACOG selects the new minority class samples from the Gibbs sampler using a predefined *lag* and this selection procedure does not take the usefulness of the generated samples into account. On the other hand, wRACOG considers the usefulness of the generated samples and selects those samples which have the highest probability of being misclassified by the existing learning model (Das, Krishnan, and Cook, 2014).

RWO-Sampling

Inspired by the central limit theorem, Zhang et al. (H. Zhang and M. Li, 2014) proposed the random walk oversampling (RWO-Sampling) approach to generate the synthetic minority class samples which follows the same distribution as the original training data. Given an imbalanced dataset with multiple attributes, the mean and the standard deviation for the ith attribute a_i ($i \in \{1, 2, 3, ..., m\}$) in minority class data can be calculated and denoted by μ_i and σ_i. Under the central limit theorem, as the number of the minority class samples approaches infinity, the

following formula holds:

$$\frac{\mu_i - \mu_i'}{\sigma_i'/\sqrt{n}} \rightarrow N(0,1), \tag{3.3}$$

where μ_i' and σ_i' are the real mean and standard deviation for attribute a_i.

In order to add m synthetic samples to the n original minority class samples, we first select at random m examples from the minority class and then for each of the selected examples $\boldsymbol{x} = (x_1, ..., x_m)$ we generate its synthetic counterpart by replacing $a_i(j)$ (the ith attribute in x_j, $j \in \{1, 2, ..., m\}$) with $\mu_i - r_i \cdot \sigma_i/\sqrt{n}$, where μ_i and σ_i denote the mean and the standard deviation of the ith feature restricted to the original minority class, and r_i is a random value drawn from the standard Gaussian distribution. We can repeat the above process until we reach the required mount of synthetic examples. Since the synthetic sample is achieved by randomly walking from one real sample, this oversampling is called random walk oversampling.

3.2.2 Data Complexity

The motivation for studying the data complexity in imbalanced data is that some researchers (Luengo, Fernández, García, and Herrera, 2011; M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018) find no clear relationship between imbalance ratio (IR) and the classification performance obtained via resampling. From their empirical studies, they conclude that IR is not a sufficient measure to identify the potential performance improvement of the data-level approaches (Luengo, Fernández, García, and Herrera, 2011). Therefore, they analyse the resampling techniques through data complexity measures (detailed introduced in Section 2.3) in further studies. One of the main results is that the Fisher discriminant ratio (F1) is informative in characterising the imbalanced classification performance. Following the idea of studying the data complexity in the context of class imbalance, Chen et al. (L. Chen, Fang, Shang, and Tang, 2018) studied the relationship between class overlap and class imbalance in software defect prediction problems. In (M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018), authors conduct detailed experiments to study the relationship between data complexity measures and imbalanced classification performance. In their regression analysis across a range of datasets, the obtained regression model can predict the AUC performance based on complexity measures with an average $R^2 = 0.72$.

3.3 Experiments

In this section, we introduce the information on the datasets used in our experiments. Then, the cross-validation in imbalanced learning is described. After that, the experimental setup and results are given.

3.3.1 Information on the Datasets

The experiments reported in this chapter are based on 19 two-class imbalanced datasets from the KEEL-collection (Alcalá-Fdez, Fernández, Luengo, Derrac, García, Sánchez, and Herrera, 2011). The 19 collected binary datasets are manually decomposed from four multi-class datasets: *ecoli*, *glass*, *vehicle* and *yeast*. Detailed information on the datasets are given in Table 3.1 & Table 3.2.

Table 3.1: Information on datasets divided into 4 groups.

Datasets	#Attributes	#Samples	Imbalance Ratio (IR)
ecoli{1,2,3,4}	7	336	{ 3.36, 5.46, 8.6, 15.8 }
glass{0,1,2,4,5,6}	9	214	{ 2.06, 1.82, 11.59, 15.47, 22.78, 6.38 }
vehicle{0,1,2,3}	18	846	{ 3.25, 2.9, 2.88, 2.99 }
yeast{1,3,4,5,6}	8	1484	{ 2.46, 8.1, 28.1, 32.73, 41.4 }

Table 3.2: Further description of the datasets (Alcalá-Fdez, Fernández, Luengo, Derrac, García, Sánchez, and Herrera, 2011)

Datasets	Description
ecoli	This is a protein localization sites classification dataset, which contains 8 classes. It is artificially modified into 4 binary datasets, where the sample proportions are { 77:259 ; 52:284 ; 35:301 ; 20:316 }.
glass	This is a glass identification dataset, which contains 6 classes. It is artificially modified into 6 binary datasets, where the sample proportions are { 70:144 ; 76:138 ; 17:197 ; 13:201 ; 9:205 ; 29:185 }.
vehicle	This is a vehicle silhouettes dataset, which contains 4 classes. It is artificially modified into 4 binary datasets, where the sample proportions are { 199:647 ; 217:629 ; 218:628 ; 212:634 }.
yeast	This is a protein localization sites classification dataset, which contains 10 classes. It is artificially modified into 5 binary datasets, where the sample proportions are { 429:1055 ; 163:1321 ; 51:1433 ; 44:1440 ; 35:1449 }.

3.3.2 Cross-Validation in Imbalanced Learning

Cross-validation (CV) is an effective technique to assess classification performance. It allows different portions of the data for training and testing a model (Bishop and Nasrabadi, 2006). In traditional k-fold CV, the original dataset is randomly partitioned into k folds, where k-1 folds are used to train the model and the left one is retained as a validation fold to test the model performance. Then, every fold iterates to be the validation fold to ensure that all folds are used for training and testing the model. After k iterations, the final performance can be estimated by averaging the k results. One significant advantage of this procedure is that the validation fold is unseen in the training process. In the imbalanced learning domain, data-level approaches are commonly used to deal with the imbalance in the datasets. Some researchers emphasize the importance of correctly understanding the joint use of CV and data-level approaches. They point out that a poorly designed CV procedure for imbalanced datasets will result in overfitting and overoptimism problems (Lusa et al., 2015; M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018).

According to *Oxford English Dictionary* [1], overfitting is a statistical term with definition “the production of an analysis which corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably". This term is then extended to machine learning, which means the learning model is highly fitted to the training data and, therefore, has poor ability to generalise on unseen data. The CV technique can alleviate the overfitting problem in most cases. However, when learning from imbalanced data, some oversampling techniques produce exact replicas of some samples (Lusa et al., 2015). Too many same patterns in the training set will result in overfitting of the model even with CV technique.

Overoptimism occurs when the training and test sets contain exact or similar replicas of some patterns (M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018). For example, suppose we first obtain a balanced dataset through oversampling approaches and then perform cross-validation when dealing with imbalanced datasets. In this way, since the synthetic samples share similar patterns with the original sample, samples with similar patterns may appear in both training and test set, which will lead to the overoptimism problem. In our experiments, we perform k-fold stratified CV before applying the six introduced oversampling techniques.

[1] https://www.oed.com/

The stratified folds ensure that the imbalance ratio in the training set is consistent with the original dataset.

3.3.3 Experimental Setup

In this chapter, six oversampling approaches (using the R package `imbalance` (Cordón, García, Fernández, and Herrera, 2018)), which have been reviewed in Section 3.2.1, are applied to the 19 two-class imbalanced datasets in Table 3.1. Every collected dataset is divided into 5 stratified folds for cross-validation and only the training set is oversampled, where the stratified fold ensures that the imbalance ratio in the training set is consistent with the original dataset and only oversampling the training set avoids the over-optimism problem (Lorena, L. P. Garcia, Lehmann, Souto, and Ho, 2019).

In this chapter, we aim to study the effectiveness of different oversampling approaches and investigate the relationship between data complexity measures and the choice of oversampling techniques. Therefore, we calculate the 7 data complexity measures (Table 2.2) for each dataset. In our 30 experiments for each dataset, we calculate the 7 data complexity measures for every training set using the R package `ECoL` (Lorena, L. P. Garcia, Lehmann, Souto, and Ho, 2019) (Table 3.3). Since we use 5 stratified cross-validations, we average each data complexity measure for these 5 training sets and define it to be the data complexity measure for the dataset.

In a binary classification problem, the confusion matrix can provide intuitive classification results. In the class imbalance domain, it is widely admitted that *Accuracy* tends to result in a deceptive evaluation of the performance. Instead of *Accuracy*, the Area Under the ROC Curve (AUC) and geometric mean (GM) are used to evaluate the performance (details can be checked in Section 2.1).

Table 3.3: Data complexity for 19 collected datasets.

Dataset	F1	F1v	F2	F3	L1	L2	L3
ecoli1	0.8785	0.1248	0.0229	0.5814	0.0523	0.0955	0.0586
ecoli2	0.9154	0.1323	0.0000	0.7175	0.0514	0.0806	0.0588
ecoli3	0.9248	0.1557	0.0058	0.4257	0.0516	0.0771	0.0629
ecoli4	0.9291	0.0614	0.0005	0.3584	0.0088	0.0163	0.0152
glass0	0.9525	0.3728	0.0000	0.7002	0.1181	0.2232	0.1873
glass1	0.9808	0.5749	0.0068	0.8896	0.2046	0.3409	0.3378
glass2	0.9913	0.3540	0.0000	0.5279	0.0732	0.0794	0.0778
glass4	0.9497	0.0956	0.0027	0.2784	0.0312	0.0441	0.0378
glass5	0.9753	0.1312	0.0000	0.1402	0.0061	0.0186	0.0154
glass6	0.8373	0.0435	0.0095	0.3775	0.0252	0.0260	0.0185
vehicle0	0.9156	0.0812	0.0001	0.5425	0.0103	0.0261	0.0082
vehicle1	0.9720	0.2606	0.0003	0.9362	0.0929	0.1758	0.1397
vehicle2	0.9735	0.0760	0.0024	0.7702	0.0172	0.0300	0.0142
vehicle3	0.9730	0.3075	0.0006	0.9595	0.1041	0.1818	0.1595
yeast1	0.9638	0.4407	0.0000	0.9587	0.1553	0.2496	0.2418
yeast3	0.9554	0.1343	0.0000	0.4588	0.0433	0.0510	0.0365
yeast4	0.9802	0.2013	0.0000	0.8734	0.0332	0.0344	0.0338
yeast5	0.9580	0.1049	0.0000	0.1139	0.0142	0.0224	0.0182
yeast6	0.9791	0.1468	0.0000	0.6514	0.0225	0.0232	0.0238

3.3.4 Experimental Results and Discussion

The AUC results for C5.0 decision tree and SVM in our experiments are presented in Table 3.4 and Table 3.5. Geometric mean results can be found in Table 3.6 and Table 3.7. In the experimental results of the decision tree, we observe that RACOG outperforms the other 5 oversampling techniques in 8 out of 19 datasets. The same conclusion can also be drawn from the experimental results of SVM. It is worth mentioning that RACOG costs more time than the other five considered oversampling techniques due to the execution of the Markov chain in its data generation process. From our experimental results, we conclude that, in most cases, oversampling approaches which consider the minority class distribution (RACOG, wRACOG and RWO-Sampling) perform better.

It was expected that data complexity can provide some guidance for choosing the oversampling technique, however, from our experimental results, no obvious relationship between data complexity and the choice of oversampling approaches can be concluded. This is because the 6 introduced oversampling approaches are designed for common datasets and do not take a specific data characteristic into account.

According to our experimental results, although the data complexity measures cannot provide guidance for choosing the most promising oversampling approaches, we find that there is a strong correlation between the potential best AUC (after oversampling) and some of the data complexity measures. From Figure 3.2 and Table 3.8, it can be concluded that the potentially best AUC value that can be achieved through C5.0 decision tree and oversampling techniques has an extreme negative correlation with the F1v value and the linearity measures. In the imbalanced learning domain, many researchers focus on studying data complexity measures. In (Lorena, L. P. Garcia, Lehmann, Souto, and Ho, 2019), the authors propose that the potentially best AUC value after resampling can be predicted through various data complexity measures. However, they did not consider the F1v measure, which has the strongest correlation with AUC value according to our findings. Hence, we recommend using F1v to evaluate the overlap in imbalanced datasets.

Table 3.4: AUC results for C5.0 decision tree.

Dataset	Baseline	SMOTE	ADASYN	MWMOTE	RACOG	wRACOG	RWO
ecoli1	0.9408	0.9428	0.9342	0.9414	**0.9453**	0.9384	0.9432
ecoli2	0.8736	**0.9190**	0.9102	0.9112	0.9133	0.8987	0.9143
ecoli3	0.7765	0.9170	0.9013	0.9049	**0.9204**	0.8648	0.9126
ecoli4	0.8403	**0.9271**	0.8832	0.9235	0.9244	0.8896	0.9020
glass0	0.8179	0.8328	0.8254	0.8345	**0.8470**	0.8391	0.8364
glass1	0.6995	0.7391	0.7440	0.7473	**0.7588**	0.7493	0.6944
glass2	0.7309	0.8189	**0.8201**	0.7995	0.8159	0.7960	0.7125
glass4	0.8461	0.9227	0.9203	0.9126	0.9216	0.8542	**0.9252**
glass5	0.9950	0.9927	0.9931	0.9935	0.9940	**0.9952**	0.9932
glass6	0.9341	0.9357	0.9306	0.9385	**0.9388**	0.9386	0.9354
vehicle0	0.9722	0.9730	0.9736	0.9723	0.9737	**0.9739**	0.9679
vehicle1	0.7430	0.7993	0.7916	0.7977	0.7970	**0.8000**	0.7738
vehicle2	0.9735	0.9722	0.9748	0.9757	0.9803	**0.9815**	0.9766
vehicle3	0.7858	0.8001	0.7954	0.8115	**0.8158**	0.8117	0.7907
yeast1	0.7318	0.7380	0.7282	0.7473	**0.7536**	0.6766	0.7279
yeast3	0.9335	0.9594	0.9580	0.9602	**0.9642**	0.9551	0.9422
yeast4	0.7769	**0.9020**	0.8989	0.8884	0.8549	0.8142	0.8367
yeast5	0.9555	0.9769	**0.9773**	**0.9773**	0.9761	0.9688	0.9772
yeast6	0.7307	0.8792	0.8850	0.8789	0.8806	0.7815	**0.8868**

Table 3.5: AUC results for SVM.

Dataset	Baseline	SMOTE	ADASYN	MWMOTE	RACOG	wRACOG	RWO
ecoli1	**0.9518**	0.9483	0.9435	0.9471	0.9458	0.9513	0.9455
ecoli2	0.9580	0.9563	0.9590	0.9564	0.9958	**0.9602**	0.9552
ecoli3	0.9459	0.9508	0.9462	0.9502	**0.9518**	0.9512	0.9485
ecoli4	**0.9949**	0.9922	0.9905	0.9908	0.9907	0.9948	0.9900
glass0	0.8390	0.8515	0.8475	0.8489	**0.8535**	0.8461	0.8527
glass1	0.7741	0.7765	0.7764	0.7749	**0.7802**	0.7777	0.7770
glass2	0.8206	0.8483	0.8471	0.8414	0.8609	0.8296	**0.8626**
glass4	**0.9863**	0.9855	0.9862	0.9853	0.9836	0.9862	0.9856
glass5	0.9698	**0.9807**	0.9806	0.9797	0.9785	0.9708	0.9776
glass6	0.9800	0.9773	0.9744	0.9739	0.9766	**0.9809**	0.9755
vehicle0	0.9956	**0.9959**	0.9954	0.9948	0.9950	0.9951	0.9906
vehicle1	0.8609	0.8889	0.8886	**0.8913**	0.8822	0.8812	0.8487
vehicle2	0.9952	0.9953	**0.9960**	0.9949	0.9948	0.9955	0.9943
vehicle3	0.8492	**0.8724**	0.8717	0.8709	0.8676	0.8611	0.8492
yeast1	0.7803	0.7874	0.7875	0.7826	**0.7959**	0.7768	0.7911
yeast3	**0.9730**	0.9685	0.9678	0.9689	0.9716	0.9727	0.9686
yeast4	0.8416	0.8843	0.8838	0.8878	**0.8990**	0.8703	0.8853
yeast5	0.9804	0.9867	**0.9871**	0.9868	0.9837	0.9827	0.9865
yeast6	0.8334	0.9264	0.9158	0.9272	**0.9295**	0.8709	0.9191

Table 3.6: Geometric mean results for C5.0 decision tree.

Dataset	Baseline	SMOTE	ADASYN	MWMOTE	RACOG	wRACOG	RWO
ecoli1	0.8319	0.8851	0.8769	0.8861	**0.8865**	0.8727	0.8562
ecoli2	0.8519	0.8829	0.8742	0.8784	**0.8854**	0.8701	0.8720
ecoli3	0.7173	**0.8281**	0.8100	0.8173	0.7762	0.7527	0.7458
ecoli4	0.8276	0.8617	0.8415	0.8610	**0.8681**	0.8442	0.8540
glass0	0.7691	0.7799	0.7727	0.7846	**0.7879**	0.7773	0.7829
glass1	0.7082	0.7179	0.7181	0.7193	0.7205	**0.7233**	0.6879
glass2	0.3966	0.6083	**0.6194**	0.5702	0.4938	0.5286	0.4399
glass4	0.6838	**0.8513**	0.8427	0.8344	0.8047	0.6930	0.8388
glass5	0.8868	0.9121	0.9030	0.9087	0.8850	**0.9199**	0.9076
glass6	0.8828	0.9069	0.8853	**0.9078**	0.8969	0.8792	0.8947
vehicle0	0.9158	0.9155	0.9201	0.9240	**0.9249**	0.9215	0.9228
vehicle1	0.6271	0.7104	0.7031	0.7089	0.7054	**0.7119**	0.6475
vehicle2	0.9455	0.9534	0.9587	0.9569	0.9491	0.9509	**0.9596**
vehicle3	0.6439	0.7119	0.7084	0.7113	**0.7121**	0.7059	0.6454
yeast1	0.6335	0.6893	0.6917	0.6925	**0.7024**	0.6461	0.6307
yeast3	0.8668	0.9106	0.9156	0.9067	**0.9184**	0.8959	0.8853
yeast4	0.5011	0.7006	0.6879	**0.7390**	0.6466	0.5725	0.5000
yeast5	0.8394	0.9305	**0.9399**	0.9288	0.9058	0.8669	0.8629
yeast6	0.6224	0.7688	0.7831	**0.7880**	0.7501	0.7076	0.7060

Table 3.7: Geometric mean results for SVM. "—" means that TP+FN=0 or TP+FP=0 and the performance metric cannot be computed.

Dataset	Baseline	SMOTE	ADASYN	MWMOTE	RACOG	wRACOG	RWO
ecoli1	0.8292	0.8810	0.8844	0.8782	**0.8845**	0.8622	0.8801
ecoli2	0.7278	**0.9326**	0.9179	0.9324	0.9297	0.7399	0.9306
ecoli3	0.6108	0.8722	0.8668	0.8748	**0.8764**	0.6615	0.8729
ecoli4	0.7132	0.9079	0.8987	0.9017	**0.9191**	0.7158	0.8992
glass0	0.7234	0.7900	**0.7909**	0.7850	0.7866	0.7741	0.7881
glass1	0.6419	0.6908	0.6883	0.6894	**0.6951**	0.6942	0.6861
glass2	—	0.7138	0.7080	0.7207	**0.7592**	—	0.7664
glass4	0.7079	0.8606	0.8692	0.8603	0.8658	0.7181	**0.8776**
glass5	0.0283	0.6663	0.6664	0.6644	0.6899	0.0679	**0.7630**
glass6	0.8374	0.8862	**0.8926**	0.8799	0.8889	0.8459	0.8818
vehicle0	0.9525	**0.9731**	0.9730	0.9682	0.9693	0.9677	0.9599
vehicle1	0.5668	0.8176	**0.8199**	0.8183	0.8073	0.8020	0.6520
vehicle2	0.9621	0.9728	**0.9754**	0.9727	0.9657	0.9687	0.9591
vehicle3	0.5115	0.8017	0.8048	**0.8056**	0.7986	0.7943	0.6347
yeast1	0.5888	0.7123	0.7123	0.7107	**0.7193**	0.6864	0.7162
yeast3	0.8428	0.8978	0.9023	0.8956	**0.9141**	0.8658	0.9020
yeast4	0.0084	0.7484	0.7527	0.7560	**0.8021**	0.3774	0.7525
yeast5	0.6463	0.9255	0.9278	0.9245	0.9342	0.7618	**0.9377**
yeast6	0.3701	0.8257	0.8063	0.8279	**0.8541**	0.5605	0.8310

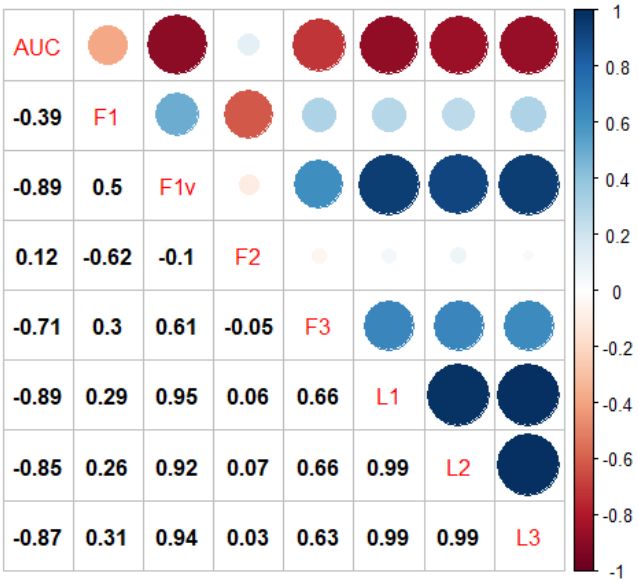

Figure 3.2: Correlation matrix. (Lorena, L. P. Garcia, Lehmann, Souto, and Ho, 2019).

Table 3.8: Results of Hypothesis Test.

Measure	Correlation Coefficient	P-value	Correlation Level
F1	-0.3872	0.1014	medium
F1v	-0.8928	2.736×10^{-7}	extreme
F2	0.1156	0.6374	none
F3	-0.7138	0.0006	high
L1	-0.8876	4.013×10^{-7}	extreme
L2	-0.8523	3.611×10^{-6}	extreme
L3	-0.8699	1.304×10^{-6}	extreme

3.4 Efficient Oversampling for Engineering Vehicle Mesh Dataset

In this section, we propose the application of the reviewed methods on the quality prediction of geometric computer aided engineering (CAE) models. For some CAE applications, like e.g. aerodynamic performance evaluation, engineers discretize the geometric models using surface meshes (undirected graphs). Each mesh consists of a set of nodes (vertices), and a set of edges connecting the nodes to form faces and volumes (elements). In computer simulations, equations describing the physical phenomena are solved with respect to the vertices allowing to approximate the solution between nodes and calculate performance features of a design, e.g. drag values as aerodynamic design quality. The meshes are generated from an initial geometric representation, e.g. non-uniform rational B-Splines (NURBS) or stereolithography (STL) representations, using numerical algorithms, such as sweep-hull for Delaunay triangulation (Sinclair, 2016), polycube (Livesu, Vining, Sheffer, Gregson, and Scateni, 2013) etc.

In most cases, the quality of the mesh plays an important role concerning the accuracy and fidelity of the results (Knupp, 2008). Engineers use different types of metrics to infer the quality of the mesh, but it is common sense that increasing the number and uniformity of the elements in the mesh improves the accuracy of the simulation results. However, the computational effort associated with meshing is proportional to the target level of refinement. Therefore, a match between accuracy and available computational resources is often required, especially for cases that demand iterative geometric modifications, such as shape optimization.

Shape morphing techniques address this issue by operating on the mesh nodes through a polynomial-based lower-dimensional representation. Such techniques avoid re-meshing of the simulation domain, thus, speeding up the optimization process. Several cases of optimization using morphing techniques are published in the literature (Menzel, Olhofer, and Sendhoff, 2005; Menzel and Sendhoff, 2008; Olhofer, Bihrer, Menzel, Fischer, and Sendhoff, 2009; Sieger, Menzel, and Botsch, 2015). For our experiments, we implemented the free form deformation (FFD) method presented in (Sederberg and Parry, 1986). To prepare design deformations based on FFD, the geometry of interest is embedded in a uniform parallelepiped lattice, where a trivariate Bernstein polynomial maps the position of the control points of the lattice to the nodes of the mesh, as an $I\!R^3 \rightarrow I\!R^3$ function. Therefore,

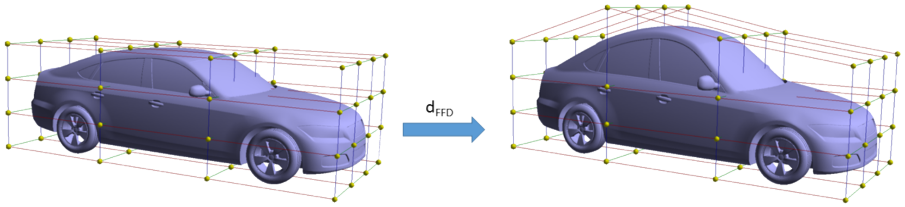

Figure 3.3: Example of free form deformation applied to a configuration of the TUM DrivAer model (Heft, Indinger, and Adams, 2012) using a lattice with four planes in each direction.

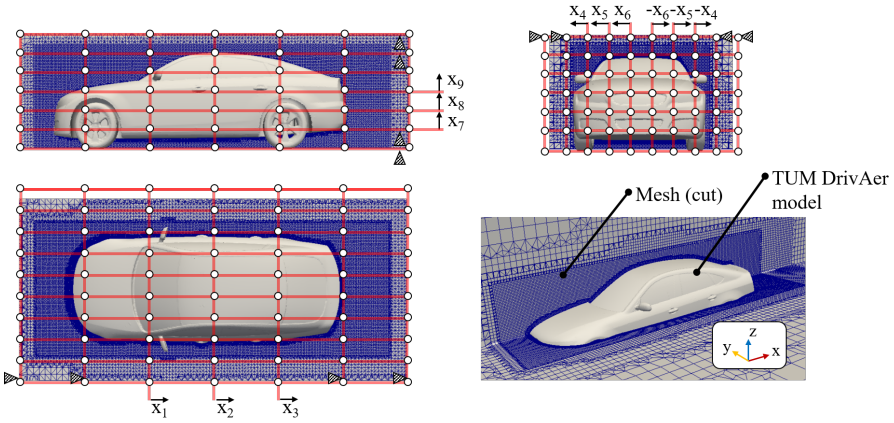

Figure 3.4: Free form deformation lattice used to generate the data set for the experiments.

by deforming the lattice, the nodes of the mesh are moved accordingly (Figure 3.3).

In order to embed the geometry in the lattice, a local coordinate system is defined taking as vector basis the unitary vectors $(\vec{s}, \vec{t}, \vec{u})$, normal to the faces of the parallelepiped and origin in $\mathbf{v_0}$. Then, the coordinates of the mesh nodes are described according to the new basis, using the following linear transformation:

$$\mathbf{v} = \mathbf{v_0} + S\vec{s} + T\vec{t} + U\vec{u} \tag{3.4}$$

where $\mathbf{v}$ is the mesh node described in global coordinate system and the new coordinates S, T and U belong to the interval $[0, 1]$. Given the set that contains the points $\mathbf{p}_{ijk}$ defined by the intersection of the planes that form the lattice, the coordinates of any mesh node can be calculated using the trivariate Bernstein

polynomial, defined as

$$\mathbf{v_{FFD}} = \sum_{i=0}^{l} \binom{l}{i} (1-S)^{l-i} S^i \left\{ \sum_{j=0}^{m} \binom{m}{j} (1-T)^{m-j} T^j \left[\sum_{k=0}^{n} \binom{n}{k} (1-U)^{n-k} U^k \mathbf{p}_{ijk} \right] \right\} \tag{3.5}$$

where $\mathbf{v_{FFD}}$ is the deformed point; l, m, n are respectively the number of control planes in the $\vec{s}$-, $\vec{t}$- and $\vec{u}$-direction.

The continuity of the surfaces is ensured by the mathematical formulation of the FFD up to the order of $k-1$, where k is the number of planes in the direction of interest, but the mesh quality is not necessarily maintained. The designer can either avoid models with ill-defined elements by applying constraints to the deformations, which might be unintuitive, or eliminate them by performing regular quality assessments. Addressing this issue, we propose the classification of the deformation parameters with respect to the quality of the output meshes, based on a data set of labeled meshes. Further than reducing the risk of generating infeasible meshes for CAE applications, our approach avoids unnecessary computation to generate the deformed meshes, which is aligned with the objective of increasing the efficiency of shape optimization tasks.

3.4.1 Generation of a Synthetic Data Set

For the experiments we adopted the computational fluid dynamics (CFD) simulation of a configuration of the TUM DrivAer model (Heft, Indinger, and Adams, 2012). The simulation model is deformed using the discussed FFD algorithm, realized as a lattice with 7 planes in x- and z-directions, and 10 in y-direction (Fig. 3.4). The planes closer to the boundaries of the control volume are not displaced in order to enable a smooth transition from the region affected by the deformations to the original domain. Assuming symmetry of the shape with respect to the vertical plane (xz) and deformations caused by the displacement of entire control planes only in the direction of their normal vectors, it yields a design space with 9 parameters. To generate the data set, the displacements x_i were sampled from a random uniform distribution and constrained to the volume of the lattice, allowing the overlap of planes.

The initial mesh was generated using the algorithms *blockMesh* and *snappyHexMesh* of OpenFOAM®[2]. We automatically generated 994 valid meshes based on the FFD algorithm implemented in Python and evaluated them using the OpenFOAM checkMesh algorithm. The metrics used to define the quality of the meshes were the number of warnings raised by the *checkMesh* algorithm, the maximum skewness and maximum aspect ratio. We manually labeled the feasible meshes according to the rules shown in Table 3.9. The imbalance ratios after manually labelling are also given in Table 3.9. Please note that the input attributes are exactly the same for all three sets of datasets, only the “class" labels are different. In this way, the values of data complexity measures (Table 3.10) for the three datasets vary from each other.

3.4.2 Experimental Results and Discussion

The experimental results on the digital vehicle dataset are given in Table 3.11. In line with our conclusions for the KEEL-dataset experiments (Section 3.3.4), we find that RACOG outperforms the other 5 oversampling techniques in 2 out of 3 datasets. Therefore, combining our experimental results on both benchmark and real-world inspired datasets, we conclude that RACOG performs the best out of the considered 6 oversampling approaches. Moreover, we find that applying the oversampling techniques can improve the performance by around 10% for our digital vehicle datasets. We also calculate the data complexity measures for our digital vehicle datasets and our findings on the correlation between the potential AUC value and the data complexity measures remain consistent with the conclusions in Section.

Table 3.9: Feasible meshes labeling rule.

Dataset	#Attribute	#Sample	#Warnings	Max skewness	Max aspect ratio	IR
set1	9	994	<4	<6	<10	3.76
set2	9	994	<2	<5.8	<10.3	6.83
set3	9	994	<4	<5.6	<10.3	12.43

Table 3.10: Data complexity HRI.

Dataset	F1	F1v	F2	F3	L1	L2	L3
set1	0.9809	0.4360	0.3123	0.9072	0.1737	0.2103	0.2115
set2	0.9950	0.7030	0.1619	0.8900	0.1133	0.1278	0.1325
set3	0.9840	0.2854	0.0962	0.7953	0.0693	0.0744	0.0709

[2]https://www.openfoam.com

Table 3.11: Experimental Results (AUC) on Digital Vehicle Dataset.

Dataset	Baseline	SMOTE	ADASYN	MWMOTE	RACOG	wRACOG	RWO
set1	0.7927	0.8332	0.8279	0.8458	**0.8512**	0.8436	0.8240
set2	0.5864	0.7619	0.7517	0.7590	**0.7633**	0.7437	0.7583
set3	0.6511	0.8215	0.8169	**0.8341**	0.8246	0.8114	0.8065

Table 3.12: Experimental Results (Geometric mean) on Digital Vehicle Dataset.

Dataset	Baseline	SMOTE	ADASYN	MWMOTE	RACOG	wRACOG	RWO
set1	0.6975	0.7557	0.7492	0.7577	**0.7612**	0.7530	0.7295
set2	0.2685	0.6685	0.6622	0.6670	**0.6781**	0.6520	0.6414
set3	0.3373	0.6657	0.6683	**0.6878**	0.6725	0.6573	0.5952

3.5 Conclusions

In this work, we reviewed six oversampling techniques, including “classical" ones (SMOTE, ADASYN and MWMOTE) and new ones (RACOG, wRACOG and RWO-Sampling), in which the new ones consider the minority class distribution while the “classical" ones do not. The six reviewed oversampling approaches were applied to 19 benchmark imbalanced datasets and an imbalanced real-world inspired vehicle dataset to investigate their effectiveness. Seven data complexity measures were considered in order to find the relationship between data complexity measures and the choice of resampling techniques. According to our experimental results, two main conclusions can be derived:

- In our experiment, in most cases, oversampling approaches which consider the minority class distribution (RACOG, wRACOG and RWO-Sampling) perform better. For both benchmark datasets and our real-world inspired dataset, RACOG performs best. However, the trade-off between performance improvement and the time cost should be considered while using RACOG.

- No obvious relationship between data complexity measures and the choice of resampling techniques can be derived from our experimental results. However, we find that the F1v value has a strong correlation with the potential best AUC value (after resampling) while only rarely researchers in

the imbalance learning domain consider F1v value for evaluating the overlap between classes.

In this chapter, we applied the oversampling techniques for benchmark datasets and our digital vehicle dataset and evaluated their effectiveness. In the next chapter, we will study hyperparameter optimisation on class-imbalance problems.

CHAPTER 4

Hyperparameter Optimisation on Class-Imbalance Problems

Although the class-imbalance classification problem has caught a huge amount of attention, hyperparameter optimisation has not been studied in detail in this field. Both classification algorithms and resampling techniques involve some hyperparameters that can be tuned. In this chapter, we study hyperparameter optimisation on class-imbalance problems and investigate the relation between the degree of class overlap and the improvement yielded via hyperparameter tuning. This chapter is divided as follows. First, Section 4.1 shows the motivation and provides a brief introduction on our work. After that, in Section 4.2, the resampling techniques used in this chapter and the background knowledge on hyperparameter optimisation are presented. In Section 4.3, the information on the datasets, the experimental setup as well as the experimental results and discussion are introduced. Section 4.4 concludes the chapter and outlines the further work.

4.1 Introduction

Over years of development, many techniques have proven to be efficient in handling imbalanced datasets. These methods can be divided into data-level approaches and algorithmic-level approaches (Bhowan, Johnston, M. Zhang, and Yao, 2012; Ganganwar, 2012; M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018), where the data-level approaches aim to produce balanced datasets and the algorithmic-level approaches aim to adjust classical classification algorithms in order to make them appropriate for handling imbalanced datasets.

By far, the most commonly used approach for handling imbalanced data

is a combination of resampling techniques and machine learning classification algorithms (López, Fernández, Moreno-Torres, and Herrera, 2012). Research works also focused on these two separate parts, developing new resampling techniques and adjusting machine learning algorithms to be more appropriate for imbalanced datasets. Both resampling techniques and machine learning algorithms involve some hyperparameters that are set to some default values and could be tuned. A minor variation of these hyperparameters might influence the performance significantly. However, hyperparameter optimisation has not been studied yet in detail in the context of learning from imbalanced data, where both components could be tuned simultaneously.

Previous research has considered the hyperparameters for the classifiers for class-imbalance problems (Thai-Nghe, Busche, and Schmidt-Thieme, 2009), but the hyperparameters in resampling techniques are not included. Agrawal et al. (Agrawal and Menzies, 2018) take the hyperparameters in SMOTE into account and propose an auto-tuning version of SMOTE. In this chapter, we explore the potential of applying hyperparameter optimisation for the automatic construction of high-quality classifiers for imbalanced data. In our research, we experiment with a small collection of imbalanced datasets and two classification algorithms: Random Forest and SVM. In each experiment we consider six scenarios for hyperparameter optimisation (see Table 4.1). For classification algorithms, we consider two conditions, algorithms with default hyperparameters (A_d) and algorithms with optimised hyperparameters (A_o). For resampling approaches, we consider three conditions, no resampling applied (R_n), resampling applied with default hyperparameters (R_d) and resampling applied with optimised hyperparameters (R_o).

Table 4.1: Six scenarios in our experiments.

Scenario	**Classification Algorithms**	**Resampling Approaches**
(1) $A_d + R_n$	Default hyperparameters	No
(2) $A_o + R_n$	Optimised hyperparameters	No
(3) $A_d + R_d$	Default hyperparameters	Default hyperparameters
(4) $A_o + R_d$	Optimised hyperparameters	Default hyperparameters
(5) $A_d + R_o$	Default hyperparameters	Optimised hyperparameters
(6) $A_o + R_o$	Optimised hyperparameters	Optimised hyperparameters

Apart from developing new techniques to deal with imbalanced datasets, the data complexity in the dataset itself has caught an increasing attention in recent studies of class-imbalance problems. As we stated in Chapter 3.2.2, it has been shown that the degradation of machine learning algorithms for imbalanced datasets is not directly caused by class imbalance, but is also related to the degree of class overlapping (Prati, Batista, and Monard, 2004), and the classification algorithms are more sensitive to noise than to class imbalance (López, Fernández, García, Palade, and Herrera, 2013). It is also concluded that data complexity may influence the choice of resampling methods (M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018). Hence, in this chapter, we consider the hyperparameter optimisation for both resampling techniques and classification algorithms. Furthermore, the relation between the degree of class overlap and the improvement achieved via hyperparameter tuning is investigated.

The results of our experiments demonstrate that an improvement can be obtained by applying hyperparameter tuning. In the six scenarios, optimising the hyperparameters for both classification algorithms and resampling approaches gives the best performance for all six datasets. Further study shows that the data complexity of the original data, especially the overlap between classes, influences whether a significant improvement can be achieved through hyperparameter optimisation. Compared to imbalanced datasets with high class overlap, hyperparameter optimisation works more efficiently for imbalanced datasets with low class overlap. In addition, we point out that resampling techniques are not effective for all datasets, and their effectiveness is also affected by data complexity in the original datasets. Hence, we recommend studying the data complexity of imbalanced datasets before resampling the samples and optimising the hyperparameters. Our work in this chapter has received more than 20 citations from other researchers till the end of 2022, which indicates our contributions to this topic.

4.2 Related Works

This section first introduces the resampling techniques used in this chapter. Then, the definition of hyperparameter optimisation and the related literature in the class-imbalance domain are given in Section 4.2.2.

4.2.1 Resampling Techniques

This section describes four resampling techniques in our experiments, two oversampling and two hybrid approaches. The two oversampling techniques, SMOTE and ADASYN, have been introduced in detail in the previous chapters. Therefore, we only provide details on the two hybrid approaches, SMOTETL and SMOTEENN.

SMOTETL

In a classification problem, a Tomek link is defined as follows (Tomek, 1976): given two samples $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ from different classes, $d(\boldsymbol{x_i}, \boldsymbol{x_j})$ the distance between $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$, and $\boldsymbol{x_l}$ is a random sample in the dataset. The pair $(\boldsymbol{x_i}, \boldsymbol{x_j})$ is defined as a Tomek link if the following requirements hold,

$$\forall \boldsymbol{x_l},\ d(\boldsymbol{x_i}, \boldsymbol{x_j}) < d(\boldsymbol{x_i}, \boldsymbol{x_l}) \text{ and } d(\boldsymbol{x_i}, \boldsymbol{x_j}) < d(\boldsymbol{x_j}, \boldsymbol{x_l}). \tag{4.1}$$

From the definition, a Tomek link is a pair of samples from different classes that are the nearest neighbours for each other, and the samples in Tomek links are either noise or borderline (Batista, Prati, and Monard, 2004).

Oversampling techniques aim to balance the class distribution via expanding the minority class space. However, some synthetic minority class samples may invade the majority class space, making the decision boundary blur. To alleviate this problem, Batista et al. (Batista, Prati, and Monard, 2004) proposed to apply Tomek links as an additional data cleaning method after SMOTE, and named the new technique SMOTETL. In the SMOTETL technique, the first step is (1) to oversample the minority classes using SMOTE and then (2) to identify the Tomek links. After that, (3) the Tomek links for the oversampled samples are removed. In this way, the SMOTETL technique provides a more clear decision boundary by removing part of the samples in the overlapping region. Figure 4.1 gives an example of clearing Tomek links for oversampled samples.

SMOTEENN

Similar to SMOTETL, SMOTEENN is also a hybrid method that combines oversampling and data cleaning techniques. SMOTEENN uses Wilson's Edited Nearest Neighbours (ENN) (D. L. Wilson, 1972) to remove any sample that has a different class from at least two of its three nearest neighbours (Lorena, L. P.

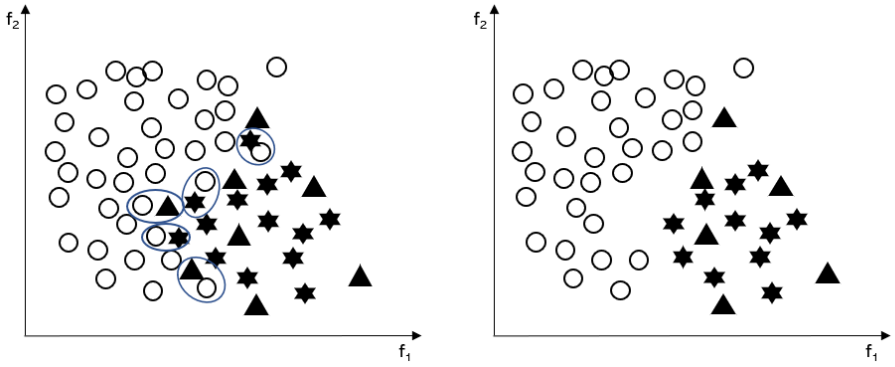

Figure 4.1: Example of clearing Tomek links for oversampled samples. ◯ and the black △ indicate the majority and minority class samples respectively. The star indicates the synthetic samples and each blue circle indicates a Tomek link.

Garcia, Lehmann, Souto, and Ho, 2018). For a binary class-imbalance problem, SMOTEENN is implemented as follows: (1) the training set is oversampled via SMOTE, then (2) for each sample in the training set, its three nearest neighbours are found. After that, (3) any sample whose label contradicts the label of at least two of its three nearest neighbours is removed. According to the ENN procedure, more samples are removed than the Tomek links, i.e. ENN provides a deeper data cleaning (Lorena, L. P. Garcia, Lehmann, Souto, and Ho, 2018).

4.2.2 Hyperparameter Optimisation

Most machine learning algorithms involve several hyperparameters, which have to be set before the training process. Compared with randomly selecting the hyperparameters in a learning algorithm, choosing a set of optimal hyperparameters can improve the performance of the algorithm. For instance, in Random Forest, the choice of the depth of a decision tree and the number of trees in a forest will have an influence on the performance. To determine the optimal combination of hyperparameters for a given problem/dataset naturally leads to the well-established hyperparameter optimisation (or hyperparameter tuning) task.

Let $\mathcal{A}$ denote a typical machine learning algorithm with n hyperparameters, $\boldsymbol{\lambda}$ denote a vector of hyperparameters and $\boldsymbol{\Lambda}$ denote the hyperparameter configuration space, i.e. $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$. A learning algorithm with hyperparameters $\boldsymbol{\lambda}$ is represented by

$\mathcal{A}_{\boldsymbol{\lambda}}$ (Feurer and Hutter, 2019). Given a dataset $\boldsymbol{X}$, the goal to find the optimal set of hyperparameters $\boldsymbol{\lambda}^*$ so as to minimize the predefined loss function $\mathcal{L}(\cdot)$ can be represented by (Bergstra, Bardenet, Bengio, and Kégl, 2011; Claesen and De Moor, 2015)

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}}{\arg\min}\ \mathcal{L}(\boldsymbol{X}^{(te)}; \mathcal{A}_{\boldsymbol{\lambda}}(\boldsymbol{X}^{(tr)}), \tag{4.2}$$

where $\boldsymbol{X}^{(tr)}$ and $\boldsymbol{X}^{(te)}$ are the training set and validation set, which are given.

There are many approaches for performing hyperparameter optimisation. *Grid search* is a traditional way of tuning hyperparameters. It starts with dividing the search space into a discrete grid. Then, grid search performs an exhaustive search on every combination of the hyperparameters, which always requires much time. *Random search* is similar to grid search but replaces the exhaustive searching on every combination with randomly selecting the combinations to test. *Bayesian hyperparameter optimisation* approaches provide a less expensive way to optimise the hyperparameters. Its strategy keeps tracking previously evaluated results and uses the obtained information to form a surrogate probabilistic model of the objective function (Bergstra, Bardenet, Bengio, and Kégl, 2011; Bergstra, Yamins, and Cox, 2013). The hyperparameters for evaluation by the objective function are selected by applying a criterion to the surrogate function, and this criterion is defined by a selection function, e.g. Expected Improvement. The optimisation procedure is described below.

- Form a surrogate probabilistic model of the objective function;
- Optimise the selection function over the surrogate model;
- Find the hyperparameter values which maximise the Expected Improvement;
- Evaluate these hyperparameters on the objective function;
- Update the surrogate according to the new performance;
- Iterate the 2nd - 5th step until time or other constraint is met.

Compared to the original objective function, the surrogate model is less expensive to optimise because it chooses the next candidate hyperparameters worth evaluating instead of wasting time on unworthy hyperparameters. In practice, there are many software packages based on Bayesian hyperparameter optimisation,