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

Kong, J.

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.

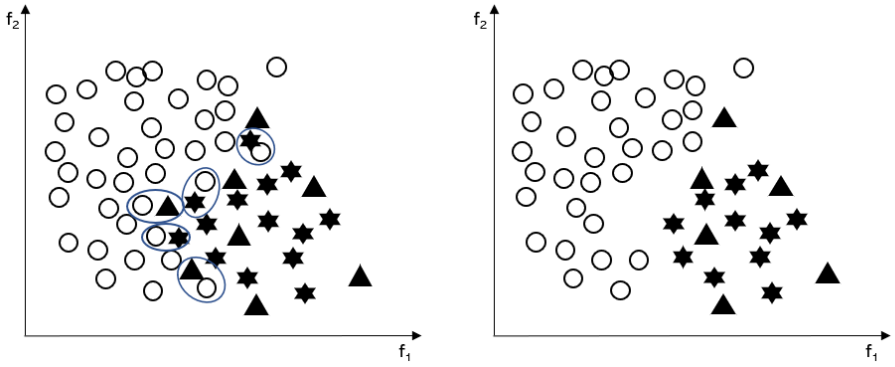

Figure 4.1: Example of clearing Tomek links for oversampled samples. ◯ and the black △ indicate the majority and minority class samples respectively. The star indicates the synthetic samples and each blue circle indicates a Tomek link.

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,

e.g. Spearmint, SMAC, HyperOpt, SPOT, etc. In this chapter, a python library[1], HyperOpt (Bergstra, Komer, Eliasmith, Yamins, and Cox, 2015), is used to perform the hyperparameter optimisation for classification algorithms.

In the field of imbalanced learning, the most basic methods are combining the resampling techniques and machine learning classification algorithms; both involve some hyperparameters that could be tuned. Hyperparameters in classifiers are widely considered in classification tasks and this is also true in the imbalanced learning domain. For example, (Thai-Nghe, Busche, and Schmidt-Thieme, 2009) searches the best hyperparameters for their classifiers when improving academic performance prediction by dealing with class imbalance. In (Shekar and Dagnew, 2019), researchers perform a grid search-based hyperparameter tuning on Random Forest classifier when their imbalanced microarray cancer data. Some studies also take the hyperparameters in resampling techniques into account. In (Douzas, Bacao, and F. Last, 2018), authors tune the k nearest neighbours in SMOTE-related resampling techniques. A representative study on hyperparameters in oversampling techniques is (Agrawal and Menzies, 2018), They take the hyperparameters in SMOTE into account and propose SMOTUNED, an auto-tuning version of SMOTE. In their experiments, SMOTUNED improved the performance dramatically, e.g. improvements in AUC up to 60% compared to SMOTE. In this chapter, we perform a detailed study on hyperparameter optimisation for class imbalance problems, i.e. considering six combinations of hyperparameters in both classification algorithm and resampling techniques (see Table 4.1 in Section 4.1).

4.3 Experiments

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

4.3.1 Information on the Datasets

The experiments reported in this chapter are based on six imbalanced datasets from the KEEL-collection (Alcalá-Fdez, Sánchez, S. Garcia, Jesus, Ventura, Garrell, Otero, Romero, Bacardit, Rivas, et al., 2009). Detailed information on the datasets are

[1] available at: http://hyperopt.github.io/hyperopt/

shown in Table 4.2. The overlap between classes is calculated by the Directional-vector Maximum Fisher's Discriminant Ratio (*F1v*). Lower *F1v* value indicates higher overlap between classes (M. S. Santos, Soares, Abreu, Araujo, and J. Santos, 2018).

Table 4.2: Information on the datasets.

Dataset	#Attributes	#Examples	#Classes	IR	F1v value
glass1	9	214	2	1.82	0.57
glass6	9	214	2	6.38	0.04
yeast3	8	1484	2	8.1	0.13
yeast4	8	1484	2	28.1	0.20
ecoli3	7	336	2	8.6	0.16
abalone19	8	4174	2	129.44	0.31

4.3.2 Experimental Setup

As mentioned in Section 4.1, we experiment with six imbalanced datasets, two algorithms and four resampling techniques. Thus, in our experiment, we have $6 \cdot 2 \cdot 5 = 60$ settings tested on each data set, with 6 scenarios, 2 classifiers, and 5 resampling approaches (including none). My co-authored work (D. A. Nguyen, Kong, H. Wang, Menzel, Sendhoff, Kononova, and Bäck, 2021) studies hyperparameter optimisation on class-imbalance problems more extensively, it includes experiments with more imbalanced datasets.

The hyperparameter optimisation for the classification algorithm is done through HyperOpt. Hyperparameters in resampling approaches includes the number of neighbours, imbalance ratio after resampling and etc. In our experiment, hyperparameter optimisation for resampling approaches is done through grid search. Whenever we optimise hyperparameters with "HyperOpt", the AUC loss (1-AUC) is set as the objective function to minimise and the number of iterations is set to 500. For each experiment, we repeated 30 times with different random seeds. After that, the paired t-tests were performed on each 30 AUC values to test if there is significant difference between the results of each scenario on a 5% significance level.

4.3.3 Experimental Results and Discussions

The experimental results are presented in Table 4.3 to investigate the importance of hyperparameter optimisation for imbalanced datasets. For all the six datasets in our experiment, we observe that optimising the hyperparameters for both classifiers and resampling approaches gives the best performance. The statistical hypothesis tests mentioned in Section 4.3.2 are performed on the AUC values of scenario $(A_d + R_d)$ and $(A_o + R_o)$. The test results indicate that there is enough statistical evidence showing the performance improvements are significant for datasets "glass1", "yeast4" and "abalone19". In other words, applying the hyperparameter optimisation does not bring significant improvement for datasets "glass6", "yeast3" and "ecoli3". This experimental result demonstrates that significant improvement can be achieved by performing hyperparameter optimisation for datasets with high *F1v* values. That is to say, hyperparameter optimisation works efficiently for datasets with low overlap between classes.

Furthermore, comparing the AUC values of scenario $(A_d + R_n)$ and $(A_d + R_d)$, for datasets "glass6", "yeast3" and "ecoli3, resampling techniques does not improve the classification performance. Thus, we can conclude that oversampling techniques are not effective for datasets with high overlap. The generated synthetic samples might bring additional noise and make the class overlap even higher. Another point worth mentioning is that, compared to datasets with high overlap, we expected the classification algorithms would perform better on datasets with low overlap. However, the experimental results are contrary to our presupposition. This is because the complexity of a classification problem is not only determined by the overlap between classes but also related to other types of complexity, such as linearity measures.

In the end, we can also observe that there is no specific combination of classifiers and resampling techniques that can provide the best performance for all datasets. For a given dataset, the best combination of classifiers and resampling approaches might depend on the data complexity itself.

4.4 Conclusions and Future Work

In this chapter we considered six scenarios of hyperparameter optimisation for classification algorithms and resampling approaches. Two main conclusions can be derived according to our experimental results:

Table 4.3: Experimental results (AUC) for two classification algorithms regarding six scenarios.
The grey shade and no shade indicate the experimental results for SVM and Random Forest respectively.
p-values indicate the statistical evidence of t-tests between experimental results of scenario $(A_o + R_o)$ and $(A_d + R_d)$.
Dataset with * indicates the results of scenario $(A_o + R_o)$ is significantly higher than results of scenario $(A_d + R_d)$.

		Resampling Approaches (SVM *vs.* Random Forest)									
Scenarios	Dataset	NONE		SMOTE		ADASYN		SMOTETL		SMOTEENN	
$A_d + R_n$		0.6753	0.8301	—	—	—	—	—	—	—	—
$A_o + R_n$		0.8309	0.8345	—	—	—	—	—	—	—	—
$A_d + R_d$		—	—	0.7165	0.8401	0.7253	0.8456	0.7416	0.8420	0.7484	0.8126
$A_o + R_d$	glass1*	—	—	0.8360	0.8537	0.8390	0.8527	0.8423	0.8479	0.8435	0.8278
$A_d + R_o$		—	—	0.7322	0.8599	0.7370	0.8498	0.7437	0.8463	0.7518	0.8216
$A_o + R_o$		—	—	0.8508	0.8649	0.8592	0.8631	0.8659	0.8527	0.8673	0.8379
p-value		—	—	$\ll 0.05$	0.0060	$\ll 0.05$	0.0133	$\ll 0.05$	0.0022	$\ll 0.05$	0.0100
$A_d + R_n$		0.9768	0.9884	—	—	—	—	—	—	—	—
$A_o + R_n$		0.9848	0.9892	—	—	—	—	—	—	—	—
$A_d + R_d$		—	—	0.9749	0.9862	0.9727	0.9849	0.9768	0.9880	0.9761	0.9870
$A_o + R_d$	glass6	—	—	0.9807	0.9893	0.9787	0.9877	0.9832	0.9886	0.9840	0.9883
$A_d + R_o$		—	—	0.9796	0.9888	0.9744	0.9870	0.9805	0.9896	0.9795	0.9905
$A_o + R_o$		—	—	0.9850	0.9897	0.9833	0.9883	0.9861	0.9917	0.9857	0.9910
p-value		—	—	0.0693	0.1633	0.1819	0.1166	0.3067	0.1513	0.0603	0.1279
$A_d + R_n$		0.9688	0.9624	—	—	—	—	—	—	—	—
$A_o + R_n$		0.9712	0.9700	—	—	—	—	—	—	—	—
$A_d + R_d$		—	—	0.9642	0.9662	0.9601	0.9670	0.9659	0.9653	0.9649	0.9693
$A_o + R_d$	yeast3	—	—	0.9663	0.9731	0.9655	0.9727	0.9701	0.9669	0.9689	0.9743
$A_d + R_o$		—	—	0.9671	0.9693	0.9628	0.9696	0.9684	0.9705	0.9668	0.9722
$A_o + R_o$		—	—	0.9704	0.9759	0.9683	0.9756	0.9733	0.9742	0.9716	0.9787
p-value		—	—	0.3890	0.1529	0.1256	0.0567	0.6166	0.0585	0.2084	0.0573

Table 4.4: Experimental results (AUC) for two classification algorithms regarding six scenarios.
The grey shade and no shade indicate the experimental results for SVM and Random Forest respectively.
p-values indicate the statistical evidence of t-tests between experimental results of scenario $(A_o + R_o)$ and $(A_d + R_d)$.
Dataset with * indicates the results of scenario $(A_o + R_o)$ is significantly higher than results of scenario $(A_d + R_d)$.

		Resampling Approaches (SVM *vs.* Random Forest)									
Scenarios	Dataset	NONE		SMOTE		ADASYN		SMOTETL		SMOTEENN	
$A_d + R_n$		0.8479	0.9211	—	—	—	—	—	—	—	—
$A_o + R_n$		0.8739	0.9389	—	—	—	—	—	—	—	—
$A_d + R_d$		—	—	0.9025	0.9165	0.8998	0.9123	0.9019	0.9257	0.9079	0.9237
$A_o + R_d$	yeast4*	—	—	0.9132	0.9300	0.9076	0.9293	0.9089	0.9312	0.9093	0.9327
$A_d + R_o$		—	—	0.9098	0.9345	0.9059	0.9319	0.9102	0.9327	0.9122	0.9291
$A_o + R_o$		—	—	0.9178	0.9393	0.9105	0.9346	0.9147	0.9389	0.9201	0.9364
p-value		—	—	$\ll 0.05$	0.0075	0.0133	0.0013	0.0061	0.0036	0.0385	0.0355
$A_d + R_n$		0.9540	0.9359	—	—	—	—	—	—	—	—
$A_o + R_n$		0.9551	0.9535	—	—	—	—	—	—	—	—
$A_d + R_d$		—	—	0.9528	0.9310	0.9505	0.9303	0.9508	0.9300	0.9514	0.9329
$A_o + R_d$	ecoli3	—	—	0.9559	0.9338	0.9519	0.9395	0.9549	0.9384	0.9529	0.9385
$A_d + R_o$		—	—	0.9562	0.9419	0.9528	0.9396	0.9569	0.9417	0.9571	0.9416
$A_o + R_o$		—	—	0.9581	0.9432	0.9543	0.9407	0.9573	0.9444	0.9598	0.9450
p-value		—	—	0.4507	0.1337	0.3408	0.1532	0.4436	0.0773	0.3596	0.0575
$A_d + R_n$		0.7373	0.7239	—	—	—	—	—	—	—	—
$A_o + R_n$		0.7687	0.8077	—	—	—	—	—	—	—	—
$A_d + R_d$		—	—	0.8051	0.7934	0.8053	0.7971	0.8051	0.7946	0.8060	0.8034
$A_o + R_d$	abalone19*	—	—	0.8478	0.8328	0.8484	0.8347	0.8473	0.8331	0.8496	0.8395
$A_d + R_o$		—	—	0.8088	0.8095	0.8097	0.8023	0.8089	0.8077	0.8108	0.8090
$A_o + R_o$		—	—	0.8494	0.8389	0.8503	0.8402	0.8488	0.8391	0.8511	0.8414
p-value		—	—	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$

1. In our experiment, the results of scenario ($A_o + R_o$) outperform the other five scenarios. Especially for imbalanced datasets with low class overlap, applying hyperparameter optimisation for both classification algorithms and resampling approaches can significantly improve the performance. Nevertheless, the time consumption caused by hyperparameter optimisation is not negligible. Therefore, we recommend studying the data complexity and considering the trade-off between time cost and potential improvement before optimising the hyperparameters.

2. Based on our experimental results, we find oversampling techniques does not give performance improvement for imbalanced datasets with high class overlap. This further emphasizes the importance of learning the data complexity before dealing with the imbalanced datasets.

In future work, more data complexity measures will be considered in order to study the relation between hyperparameter optimisation and data complexity in detail. Additionally, more attention should be put on developing techniques which can efficiently handle complex imbalanced datasets. Finally, we observe the best choice of classifiers and oversampling techniques depends on the dataset itself. Therefore, another study worth exploring would be to produce a semi-automatic approach which can help choosing the best combination of resampling approaches, machine learning algorithms and hyperparameter optimisation strategies.