Ref. Ares(2021)2210237 - 30/03/2021

Project Number: 766186
Project Acronym: ECOLE
Project title: Experienced-based Computation: Learning to Optimise

Deliverable D3.1
Novel Techniques for Class Imbalance Problems

Authors:
Stephen Friess, Shuo Wang, Xin Yao – University of Birmingham
Jiawen Kong, Thomas Bäck – Universiteit Leiden

Project Coordinator: Professor Xin Yao, University of Birmingham
Beneficiaries: Universiteit Leiden, Honda Research Institute Europe, NEC Laboratories Europe

H2020 MSCA-ITN
Date of the report: 31.03.2021

Contents

Acknowledgment

While originally D3.1 and WP3.1 were proposed to strongly consider aspects of semi-supervised and active learning, the focus of our research shifted due to being more readily available rather towards the investigation of resampling and anomaly detection methods through additional attributes. In regards, to the direction proposed on semi-supervised learning, one may consider the found resampling techniques within the literature as variations to these. Therefore, the title of this deliverable was changed from "Semi-supervised learning for class imbalance problems" to "Novel techniques for class imbalance problems". This report is mainly based upon published papers from ESR5, with additional contributions from ESR4.

1. Executive summary

The objective of WP3.1 is to develop novel techniques for industrial class-imbalance optimisation problems, and therefore to save the computation time and simulation cost. Within the imbalance research in ECOLE project, data complexity in the imbalanced datasets is studied. The particular focus was on whether significant performance improvement can be achieved on any given imbalanced datasets through performing hyperparameter optimisation techniques. After that, we further studied the performance of several resampling techniques and investigated the relationship between data complexity measures and different resampling techniques, based on both benchmark datasets and a real-world inspired vehicle mesh dataset (detailed information provided in deliverable D1.2). According to our experimental results, it is shown that applying resampling techniques on the proposed industrial dataset can improve the classification performance by around 10%. Moreover, we also proposed to improve imbalanced classification by introducing additional attributes, which gives significant improvement on imbalanced classification performance and is simple to implement and can be combined with resampling techniques and other algorithmic-level approaches in the imbalanced learning domain.

2. Major achievements

Major scientific achievements concerning the research invested in this deliverable are presented. In particular, short answers to some of the most important research questions − practical issues − are described:

Research Questions	Discussion
Does hyperparameter optimisation bring improvement to the imbalanced classification performance?	Our experimental results demonstrate that significant improvement can be achieved by performing hyperparameter optimisation for datasets with low overlap between classes.
Is there any relationship between data complexity measures and the choice of resampling techniques?	Although no obvious relationship can be abstracted in our experiments, we find $\boldsymbol{F1v}$ value, a measure for evaluating the overlap which most researchers ignore, has a strong negative correlation with the potential AUC value (after resampling).
Does introducing additional attributes bring improvement to the imbalanced classification performance?	According to our experimental results, introducing additional attributes can improve the imbalanced classification performance in most cases (6 out of 7 datasets). Further study shows that this performance improvement is mainly contributed by a more accurate classification in the overlapping region of the two classes (majority and minority classes). The proposed idea of introducing additional attributes is simple to implement and can be combined with resampling techniques and other algorithmic-level approaches in the imbalanced learning domain.

3. Introduction

Experience-Based Computation: Learning to Optimize (ECOLE) is a Marie-Curie ITN project, in collaboration with University of Birmingham, Honda Research Institute, Leiden University and NEC Labs Europe GmbH. The project has been further divided into several subprojects, where each early stage researcher (ESR) is responsible for each subproject. The research aims of ECOLE include shortening the product cycle, reducing the resource consumption during the complete process, and creating more balanced and innovative products. Instead of just developing technologies to solve a given problem, it will take a bold step forward and propose to use knowledge automatically across different problem domains. Referring to knowledge, skill, and practice derived from problem solving processes in time, the experience of optimizing one product or process will be learned and transferred automatically to solve other optimization problems.

Within many optimization processes, solutions are produced which can be classified according to their feasibility. However, the ratio of the number of feasible to infeasible solutions might be strongly imbalanced. Thus, rendering the employment of any classifier to automatically detect unwanted solutions ineffective due to performance degradation. For example, in aerodynamics optimization, most car shapes generated are not feasible or novel. Only a small number of designs are worth exploring. Within the ECOLE project, deliverable D3.1 “Novel Techniques for Class Imbalance Problems” is dedicated towards advancing this topic by exploring techniques for handling class imbalance problems. This report on deliverable D3.1 is therefore a summary on the conducted research and the achievements within work package WP3 related to this line of investigation.

Specifically, we report in the following about completed activities from work package 3.1 “Class imbalance classification through semi-supervised and active learning for class imbalance problems”. In the following, Section 2 provides a formal description of the class imbalance problem as well as provides an overview on basic techniques for handling class imbalance problems. In Section 3, hyperparameter optimization as well as the relationship between data complexity measures and different resampling techniques are investigated. Section 4 explains the studies on anomaly detection with additional attributes. Finally, Section 5 provides a summary and outlook of the discussed results.

4. The Class Imbalance Classification Problem

The section first introduces the challenge of class imbalance classification (Section 4.1). After that, common-used techniques to handle class imbalance problems are introduced (Section 4.2).

4.1. The Challenge of Class Imbalance Classification

The imbalanced classification problem has caught growing attention from both the academic and industrial fields. Technically, any dataset with an unequal class distribution is imbalanced. However, only datasets with a significantly skewed distribution are regarded to be imbalanced in the imbalanced learning domain **[1]**. Here, the one or more classes being underrepresented are called *minority* class(es) and the other class(es) are called *majority* classes.

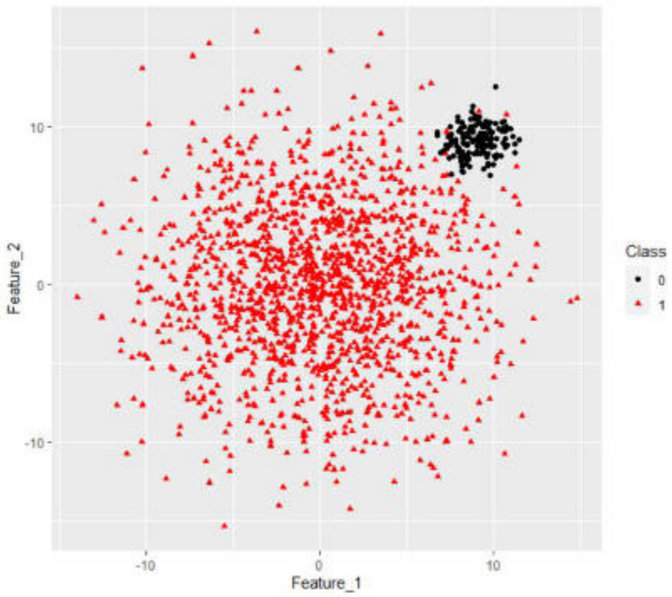


Figure 1: Example of a two-class imbalanced problem [1] with imbalanced ratio of 100:1.

This type of data will result in a reduction in the effectiveness of classical machine learning classification algorithms, because these algorithms assume that the distributions of the classes in the dataset are roughly equal **[2]**. When faced with significantly imbalanced data, these algorithms will be heavily biased towards the majority class and give deceptive accuracy. For example, suppose there are two classes in the dataset (shown in Figure 1), and the ratio of the majority to minority class is 100:1. In this situation, the classification algorithms will tend to predict all the samples as the majority class and give a 99% accuracy. This high accuracy is deceptive since the algorithms neglect the minority class. When learning from the imbalanced data, the algorithms pay more attention to the majority class, whereas, usually the minority class is the one which catches more attention in real life, such as disease recognition and fraud detection. This contradiction brings the challenges of dealing with imbalanced data. The main challenge of class-imbalance problems is to improve the accuracy of predicting the minority class without losing so much accuracy of the majority class.

4.2. Techniques to Handle Class Imbalance Problems

The main techniques to handle class imbalance problems can be divided into four categories, data-level approaches, algorithm-level approaches, cost-sensitive learning and ensemble-based learning **[1]**.

Data-level approaches (also known as resampling techniques) are straight-forward ways to deal with the class imbalance problems and they aim at rebalancing the class distribution by resampling the data space. Resampling techniques can be categorized into three groups **[3]**:

- **oversampling approaches**: replicating the existing minority samples or creating synthetic minority samples based on the original ones
- **undersampling approaches**: deleting the existing majority samples
- **hybrid approaches**: combining both sampling approaches above

Algorithm-level approaches adapt the existing classification learning algorithm to bias the learning towards the minority class. This always requires a deep understanding of the working mechanism of the selected classifier. Many popular machine learning classifiers have alternative versions for dealing with imbalanced problems **[1]**, including SVM, Decision Trees, Bayesian classifiers and etc. In the imbalanced learning domain, cost-sensitive learning can be regarded as a specific type of algorithm-level approach **[1]**. A misclassification cost is introduced to eliminate the classifier degradation brought by the skewed distribution in the imbalanced dataset. In terms of ensemble-based learning, it is a combination between an ensemble learning algorithm and one of the techniques above.

5. Studying Data Complexity in Imbalanced Datasets

Apart from developing new approaches to solve class-imbalance problems, various studies have pointed out that it is important to study the characteristics of the imbalanced dataset **[4] [5]**. In **[4]**, authors emphasize the importance to study the overlap between the two-class samples. In **[5]**, authors set up several experiments with the KEEL benchmark datasets **[6]** to study the relationship between various data complexity measures and the potential AUC value. It is also pointed out in **[5]** that the distinctive inner procedures of oversampling approaches are suitable for particular characteristics of data. However,

In the following, we will therefore study data complexity in the context of class imbalance learning. We first begin by investigating of hyperparameter tuning in the context of class imbalance learning in Section 5.1. Therefore, we introduce the considered scenarios of interest, studied resampling techniques, performance and data complexity measures, as well as subsequently present the experimental results, which clearly indicate the beneficial effect of previous hyperparameter tuning. However, we also acknowledge some associated disadvantages. Subsequently, in Section 5.2 we report on results of our study on the relationship between data complexity measures and different resampling techniques.

5.1. Hyperparameter Optimisation for Improving Classification under Class Imbalance

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. We explore the potential of applying hyperparameter optimisation for 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: RandomForest and SVM. In each experiment we consider six scenarios for hyperparameter optimisation (see Table 1).

Table 1: Six scenarios in our experiments.

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

5.1.1 Resampling Techniques

SMOTE The synthetic minority over-sampling technique (SMOTE), proposed in 2002, is the most popular resampling technique **[7]**. SMOTE produces balanced data through creating artificial data based on the randomly chosen minority samples and their *K*-nearest neighbors **[7]**. A new synthetic sample $\boldsymbol{x_s}$ can be generated according to the following equation **[8]**

$$\boldsymbol{x_s = x_i + \delta \cdot (\hat{x}_i - x_i)},$$

where $\boldsymbol{x_i}$ is the minority sample to oversample, $\boldsymbol{\hat{x}_i}$ is a randomly selected neighbor from its *K*-nearest minority class neighbors and $\boldsymbol{\delta}$ is a random number, where $\boldsymbol{\delta} \in [\mathbf{0}, \mathbf{1}]$. Figure 2 illustrates how the synthetic samples are created in the SMOTE technique.

ADASYN The adaptive synthetic (ADASYN) sampling technique is a method that aims to adaptively generate minority samples according to their distributions **[9]**. The samples which are harder to learn are given higher importance and will be oversampled more often **[5]**. The key point in ADASYN is to determine a weight ($\boldsymbol{\hat{r}_i}$) for each minority sample and use $\boldsymbol{\hat{r}_i}$ as the sampling importance. Weight $\boldsymbol{\hat{r}_i}$ of a minority sample $\mathbf{x_i}$ is defined as **[9]**

$$\boldsymbol{\hat{r}_i = \frac{r_i}{\sum_{i=1}^{m_s} r_i}}, \quad \boldsymbol{r_i = \frac{\Delta_i}{K}}, \quad \boldsymbol{i = 1, \dots, m_s},$$

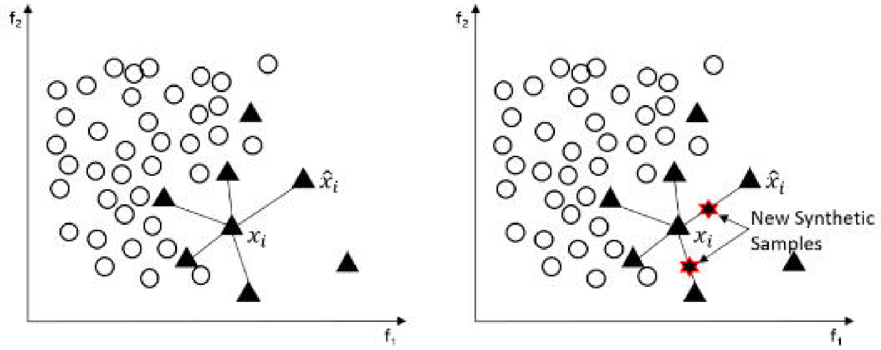


Figure 2: An illustration of how to generate synthetic samples through SMOTE. Example of K-nearest minority class neighbors for sample x_i (K=5) (left) and new synthetic samples generated through SMOTE (right).

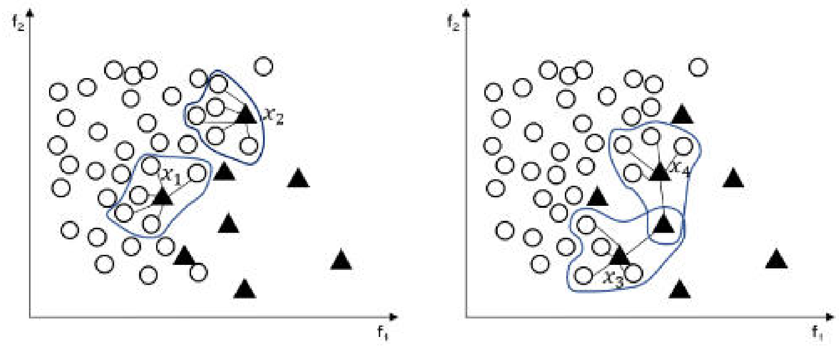


Figure 3: Example of sampling importance for different minority samples. According to definition, $r_1 = r_2 = 1$, $r_3 = r_4 = 0.8$ *and* $\hat{r}_1 = \hat{r}_2 > \hat{r}_3 = \hat{r}_4$, *indicating the sampling importance of sample* x_1, x_2 *is higher than* x_3, x_4 *and more synthetic samples will be produced for* x_1, x_2.

where $\boldsymbol{m_s}$ is the number of minority samples, $\boldsymbol{\Delta_i}$ is the number of neighbors of $\mathbf{x_i}$ that belong to majority class. For a specific minority sample, if the value of $\boldsymbol{r_i}$ is close to 1, it indicates a high level of difficulty to learn it. Then, the synthetic samples that will be generated for a minority sample can be calculated by **[9]**

$$\boldsymbol{g_i = \widehat{r_i} \cdot G},$$

where $\boldsymbol{G}$ is the total number of synthetic minority samples that need to be produced. Figure 3 shows an example of the sampling importance for different minority samples.

SMOTETL and **SMOTEENN** In a binary classification problem, a Tomek link is defined as a pair of samples from different classes which are the nearest neighbors for each other **[10]**. In the SMOTETL technique, the first step is to oversample the minority classes using SMOTE and then the Tomek links for the oversampled samples are removed **[3]**. In other words, the SMOTETL (Figure 4) technique provides a more clear decision boundary by removing part of the samples in the overlapping region. Similar to SMOTETL, the first step of SMOTEENN is also to oversample the minority class with SMOTE. After that, the Wilson's Edited Nearest Neighbors (ENN) are used to remove the sample who has a different class from at least two of its three nearest neighbors **[11]**. By removing the noisy samples, SMOTEENN (Figure 5) makes the classification algorithm work more efficiently.

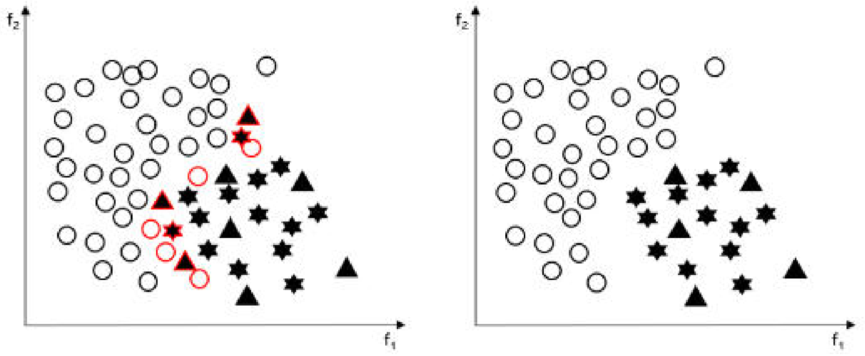


Figure 4: Example of clearing Tomek links for oversampled samples.

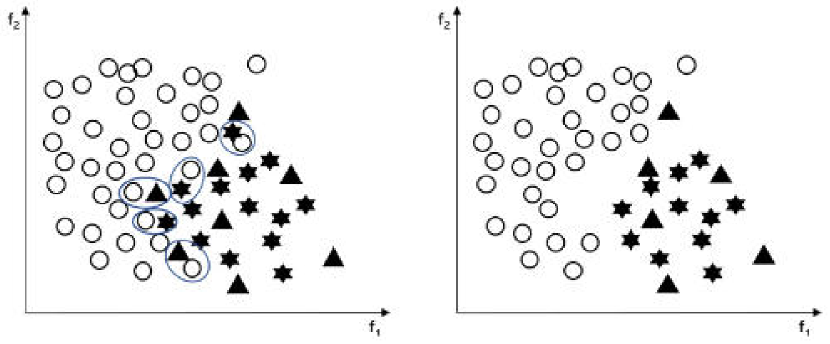


Figure 5: Example of removing the noisy samples using ENN.

5.1.2 *Hyperparameter Optimisation*

The naïve approach to imbalanced learning is just to combine resampling techniques and machine learning classification algorithms. However, compared with randomly selecting the hyperparameters in a learning algorithm, choosing a set of optimal hyperparameters should improve the performance of the algorithm.

Within our studies RandomForest and SVM are considered to do the classification and both algorithms involve various hyperparameters, which affect the performance (e.g., prediction accuracy) significantly. For instance, in RandomForest, 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 best set of hyperparameters for a given problem/dataset naturally leads to the well-established hyperparameter optimisation task. The hyperparameter optimisation problem can be represented by **[12]**

$$\boldsymbol{x}^* = \mathbf{arg}\min_{x \in \chi} \boldsymbol{f}(\boldsymbol{x}),$$

where $\boldsymbol{x}$ can be any combination of hyperparameters in domain χ and $\boldsymbol{x}^*$ is the set of hyperparameters that achieve the lowest value of objective function $\boldsymbol{f}(\boldsymbol{x})$. Typically, it is expensive to evaluate $\boldsymbol{f}(\boldsymbol{x})$ directly.

5.1.3 *Performance Measures*

Accuracy is the most commonly used measure for classification problems. In a binary classification problem, the confusion matrix (see Table 2) provides an intuitive approach towards defining accuracy.

Table 2: Confusion matrix for a binary classification problem.

	Positive Prediction	**Negative Prediction**
Positive Class	True Positives (TP)	False Negatives (FN)
Negative Class	False Positives (FP)	True Negatives (TN)

Based upon the entries, accuracy (Acc) can be subsequently defined as the sum over all true positives (TP) and negatives (TN) averaged over the sum of all samples, such that

$$\boldsymbol{Acc = \frac{TP + TN}{TP + FN + FP + TN}}.$$

However, as mentioned in Section 4.1, accuracy may give a deceptive evaluation and does not reflect the actual effectiveness of an algorithm in imbalanced domains. In imbalance learning domain, the Area Under the ROC Curve (AUC) can be used to evaluate the performance **[13] [4]** and can be computed by

$$\boldsymbol{AUC = \frac{1 + TP_{rate} - FP_{rate}}{2}, TP_{rate} = \frac{TP}{TP + FN}, FP_{rate} = \frac{FP}{FP + TN}},$$

where $\boldsymbol{TP_{rate}}$ is the true positives rate, $\boldsymbol{FP_{rate}}$ is the false positives rate. Apart from the AUC value, there are also some other measures to assess the performance for imbalanced datasets, such as geometric mean (GM) **[14]** and F-measure (FM) **[4]**. These measures are given by

$$\boldsymbol{GM = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{FP + TN}}, FM = \frac{(1 + \beta)^2 \times Recall \times Precision}{\beta^2 \times Precision + Recall}},$$

$$\boldsymbol{Recall = \frac{\mathrm{TP}}{\mathrm{TP + FN}}, Precision = \frac{TP}{TP + FP}},$$

where $\boldsymbol{\beta}$ is a coefficient and normally set to 1.

5.1.4 *Data Complexity Measures*

For these studies only one of the feature overlapping measures, the *maximum Fisher's discriminant ratio* is considered. The *maximum Fisher's discriminant*, denoted by $\boldsymbol{F1}$, measures the overlap between the feature values of different classes and is given by **[11]**

$$\boldsymbol{F1 = \max_{i=1}^{m} r_{f_i}},$$

where $\boldsymbol{m}$ is the number of features, $\boldsymbol{r}_{\boldsymbol{f}_i}$ is the discriminant ratio for each feature $\boldsymbol{f}_i$. In a binary classification problems, $\boldsymbol{r}_{\boldsymbol{f}_i}$ can be calculated as below **[15] [11]**:

$$\boldsymbol{r}_{\boldsymbol{f}_i} = \frac{\sum_{c=1}^{2} \boldsymbol{n}_c \left(\boldsymbol{\mu}_c^{\boldsymbol{f}_i} - \boldsymbol{\mu}^{\boldsymbol{f}_i}\right)^2}{\sum_{c=1}^{2} \sum_{j=1}^{\boldsymbol{n}_c} \left(\boldsymbol{x}_j^c - \boldsymbol{\mu}_c^{\boldsymbol{f}_i}\right)^2},$$

where $\mathbf{n_c}$ is the number of examples in class $\boldsymbol{c}$, $\boldsymbol{\mu}_{\mathbf{c}}^{\mathbf{f_i}}$ is the mean value of feature $\boldsymbol{f}_i$ across class $\boldsymbol{c}$, $\boldsymbol{\mu}^{\mathbf{f_i}}$ is the mean value of feature $\boldsymbol{f}_i$ across all classes, and $\mathbf{x}_{\mathbf{j}}^{\mathbf{c}}$ represents the value of feature $\boldsymbol{f}_i$ for a sample from class $\boldsymbol{c}$ **[11]**. F1 measures the highest discriminant ratio among all the features in the dataset and higher discriminant ratio indicates lower complexity **[11]**. An example of F1 computation is given in Figure 6.

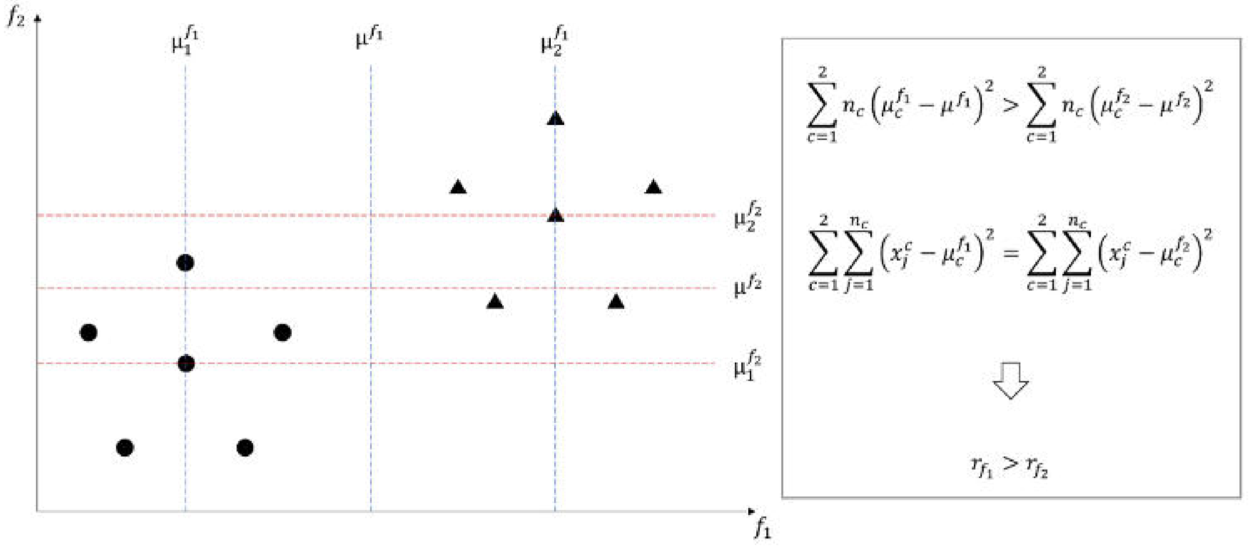

f_2
$\mu_1^{f_1}$
μ^{f_1}
$\mu_2^{f_1}$
$\mu_2^{f_2}$
μ^{f_2}
$\mu_1^{f_2}$
f_1
$\sum_{c=1}^{2} n_c \left(\mu_c^{f_1} - \mu^{f_1}\right)^2 > \sum_{c=1}^{2} n_c \left(\mu_c^{f_2} - \mu^{f_2}\right)^2$
$\sum_{c=1}^{2}\sum_{j=1}^{n_c} \left(x_j^c - \mu_c^{f_1}\right)^2 = \sum_{c=1}^{2}\sum_{j=1}^{n_c} \left(x_j^c - \mu_c^{f_2}\right)^2$
$r_{f_1} > r_{f_2}$

Figure 6: Example of F1 computation for a binary dataset [11].

5.1.5 *Experimental Setup and Results*

The experiments reported here are based on six imbalanced datasets from the KEEL-collection **[6]**. Detailed information on the datasets are shown in Table 3. IR indicates the imbalance ratio, which is the ratio of the number of majority class samples to the number of minority class samples. The overlap between classes is calculated by *Maximum Fisher's Discriminant Ratio* ($\boldsymbol{F1}$). Lower F1 value indicates higher overlap between classes **[5]**.

Table 3: Information on the datasets.

Dataset	**#Attributes**	**#Examples**	**#Classes**	**IR**	**F1**
glass1	9	214	2	1.82	0.92
glass6	9	214	2	6.38	0.53
yeast3	8	1484	2	8.1	0.70
yeast4	8	1484	2	28.1	0.91
ecoli3	7	336	2	8.6	0.84
abalone19	8	4174	2	129.44	0.96

We experiment with six imbalanced datasets, two algorithms and four resampling techniques. Thus, in our experiment, we have 6×2×5 = 60 settings tested on each data set, with 6 scenarios, 2 classifiers, and 5 resampling approaches (including none).

The hyperparameter optimisation for classification algorithm is done through HyperOpt. Hyperparameters in resampling approaches includes the number of neighbors, 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 minimize 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.

*Table 4: Experimental results (AUC) for two classification algorithms regarding six scenarios. The grey shade and no shade indicate the experimental results for SVM and RandomForest 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.* RandomForest)									
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
$A_d + R_n$		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		—	—	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$

For all the six datasets in our experiment (experimental results shown in Table 4), we observe that optimising the hyperparameters for both classifiers and resampling approaches gives the best performance. Nevertheless, the time consumption caused by hyperparameter optimisation is not negligible. Our experimental results also demonstrate that significant improvement can be achieved by performing hyperparameter optimisation for datasets with high F1 values. That is to say, hyperparameter optimisation works efficiently for datasets with low overlap between classes.

5.2. Relationship between Data Complexity and the Choice of Resampling Techniques

Although over 90 sampling approaches have been developed in the imbalance learning domain, most of the empirical study and application work are still based on the "classical" resampling techniques. We setup several experiments on 19 benchmark datasets are set up to study the efficiency of six powerful oversampling approaches, including both "classical" and new ones. We also perform the experiments on our real-world inspired vehicle mesh dataset, which has been detailed introduced in deliverable D1.2.

5.2.1 *Resampling Techniques*

The six resampling techniques can be categorized into two groups:

- Classical ones: which focus on the local information, including SMOTE, ADASYN and MWMOTE (the first two has been introduced in Section 5.1.1).
- New ones: which consider the minority class distribution, including RACOG, wRACOG and RWO-sampling.

MWMOTE Compared to other oversampling techniques, the majority weighted minority oversampling techniques (MWMOTE) improves the sample selection scheme and the synthetic sample generation scheme **[16]**. MWMOTE first finds the informative minority samples ($\mathbf{S_{imin}}$) by removing the "noise" minority samples and finding the borderline majority samples. Then, every sample in $\boldsymbol{S_{imin}}$ is given a selection weight $\boldsymbol{S_w}$, 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 in to selection probability ($\boldsymbol{S_p}$), 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 samples in $\boldsymbol{S_{imin}}$ into $\boldsymbol{M}$ groups; 2). select a minority sample $\boldsymbol{x}$ from $\boldsymbol{S_{imin}}$ according to $\boldsymbol{S_p}$ and randomly select another sample y from the same cluster of $\boldsymbol{x}$; 3). use the same equation employed in 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 reply 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 samples cannot be guaranteed. In order to tackle this problem, Das et al. **[17]** proposed RACOG (RApidy COnverging Gibbs) and wRACOG (Wrapper-based RApidy COnverging Gibbs).

In these two algorithms, the *n*-dimensional probability distribution of 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. The minority class data points are chosen to be the 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 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 **[17]**.

RWO-Sampling Inspired by the central limit theorem, Zhang et al. **[18]** 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 standard deviation for the $\boldsymbol{i}$th attribute $\boldsymbol{a_i}$ ($\mathbf{i} \in \mathbf{1}, \mathbf{2}, \mathbf{3}, \dots, \mathbf{m}$) in minority class data can be calculated and denoted by $\boldsymbol{\mu}_\mathbf{i}$ and $\boldsymbol{\sigma}_\mathbf{i}$. Under the central limit theorem, as the number of the minority samples approaches infinite, the following formula holds

$$\frac{\boldsymbol{\mu}_{i} - \boldsymbol{\mu}'_{i}}{\boldsymbol{\sigma}'_{i}/\sqrt{\boldsymbol{n}}} \to \boldsymbol{N}(\mathbf{0}, \mathbf{1}),$$

where $\boldsymbol{\mu}'_\mathbf{i}$ and $\boldsymbol{\sigma}'_\mathbf{i}$ is the real mean and standard deviation for attribute $\mathbf{a_i}$.

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

5.2.2 *Data Complexity Measures*

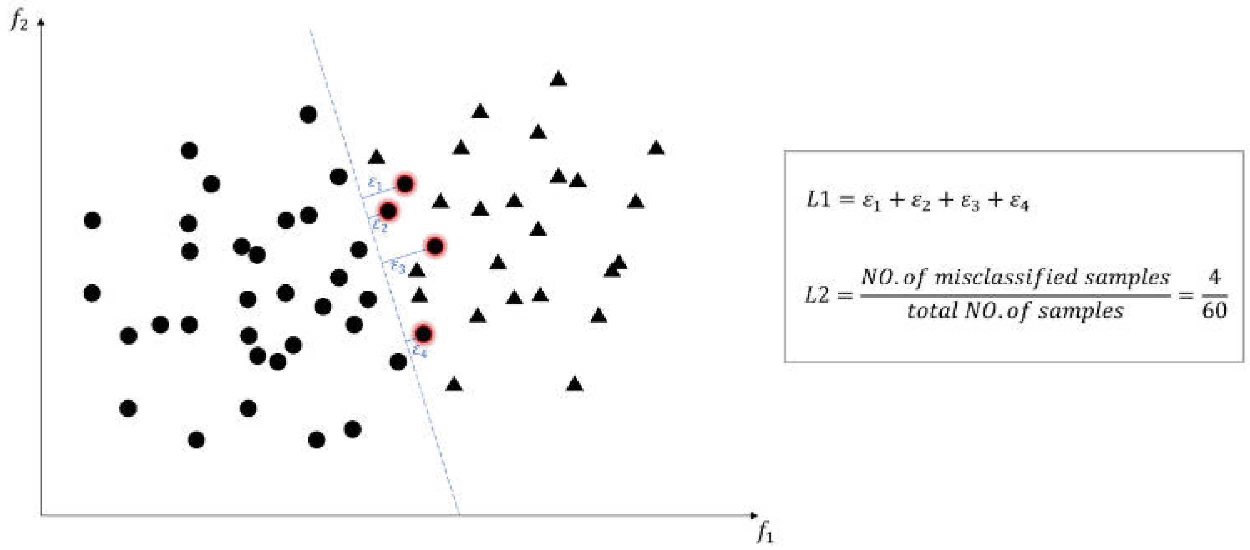
For the data complexity measures in binary classification problems, the measures can be divided into *feature overlapping measures*, *measures of the separability of classes* and *geometry*, *topology* and *density of manifolds measures* **[11] [19]**. Among these measures, we consider *feature overlapping measures* and *linearity measures* in this part of research (Table 5), where the former characterize how informative the features classify the classes and the later ones try to quantify the linear separability of the classes **[11]**.

Table 5: Complexity measures information. "Positive" and "Negative" indicate the positive and negative relation between measure value and data complexity respectively.

Measure	**Description**	**Complexity**
F1	Maximum Fisher's Discriminant Ratio	Negative
F1v	The Directional-vector Maximum Fisher's Discriminant Ratio	Negative
F2	Volume of Overlapping Region	Positive
F3	Maximum Individual Feature Efficiency	Negative
L1	Sum of the Error Distance by Linear Programming	Positive
L2	Error Rate of Linear Classifier	Positive
L3	Non-Linearity of a Linear Classifier	Positive

Feature Overlapping Measures Detailed information on $\boldsymbol{F1}$ has been given in Section 5.1.4. $\boldsymbol{F1v}$ is a complement of $\boldsymbol{F1}$ and a higher value of $\boldsymbol{F1v}$ indicates there exists a vector that can separate different class samples after these samples are projected on it **[20]**. $\boldsymbol{F2}$ calculates the overlap ratio of all features (the width of the overlap interval to the width of the entire interval) and returns the product of the ratios of all features **[20]**. $\boldsymbol{F3}$ measures the individual feature efficiency and returns the maximum value among all features.

Linearity Measures $\boldsymbol{L1}$ and $\boldsymbol{L2}$ both measure to what extent the classes can be linearly separated using an SVM with a linear kernel **[20]**, where $\boldsymbol{L1}$ returns the sum of the distances of the misclassified samples to the linear boundary and $\boldsymbol{L2}$ returns the error rate of the linear classifier. An example of $\boldsymbol{L1}$ and $\boldsymbol{L2}$ computation is given in Figure 7. $\boldsymbol{L3}$ returns the error rate of an SVM with linear kernel on a test set, where the SVM is trained on training samples and the test set is manually created by performing linear interpolation on the two randomly chosen samples from the same class.



$$L1 = \varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4$$

$$L2 = \frac{NO.of\ misclassified\ samples}{total\ NO.of\ samples} = \frac{4}{60}$$

Figure 7: Example of L1 and L2 computation for a binary dataset.

5.2.3 *Experimental Setup and Results*

The experiments in this part of the report are based on 19 two-class imbalanced datasets from the KEEL-collection **[6]** and six powerful oversampling approaches (using **R** package *imbalance* **[21]**), which have been reviewed in the past sections. Every collected dataset is divided into 5 stratified folds (for cross-validation) and only the training set is oversampled, where the stratified fold is to ensure the imbalance ratio in the training set is consistent with the original dataset and only oversampling the training set is to avoid over-optimism problem **[5]**. The 19 collected datasets can be simply divided into 4 groups, ecoli, glass, vehicle and yeast (Table 6).

Table 6: Information on datasets in 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 }

We aim to study the efficiency of different oversampling approaches and investigate the relationship between data complexity measures and the choice of oversampling techniques. Therefore, we need to calculate the 7 data complexity measures (shown in the previous section) for each dataset. In our 30 experiments for each dataset, we calculate the 7 data complexity measures for every training set using **R** package *ECoL* **[21]**. Since we use 5 stratified cross-validations, we average each data complexity measures for these 5 training sets and make it the data complexity measure for the dataset.

Table 7: 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**

The AUC results for C5.0 decision tree in our experiments are presented in Table 7. In the experimental results of decision tree, we can observe that RACOG outperforms the other 5 oversampling techniques in 8 out of 19 datasets. It is worth mentioning that RACOG costs more time than the other five considered oversampling techniques due to the running of Markov chain in its data generation process. From our experimental results, we can 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.

1. According to our experimental results, although the data complexity measures cannot provide guidance for choosing the oversampling approaches, we find there is a strong correlation between the potential best AUC (after oversample) and some of the data complexity measures. From **Error! Reference source not found.** below and

Table **8** below, it can be concluded that the potential best AUC value that can be achieved through C5.0 decision tree and oversampling techniques has an extreme negative correlation with the $\boldsymbol{F1v}$ value and linearity measures. Compared to literature, only in **[22]** the authors demonstrate that $\boldsymbol{F1}$ value has an influence on the potential improvement brought by oversampling approaches. However, they did not consider the $\boldsymbol{F1v}$ measure. Hence, we recommend using $\boldsymbol{F1v}$ to evaluate the overlap in imbalanced dataset.

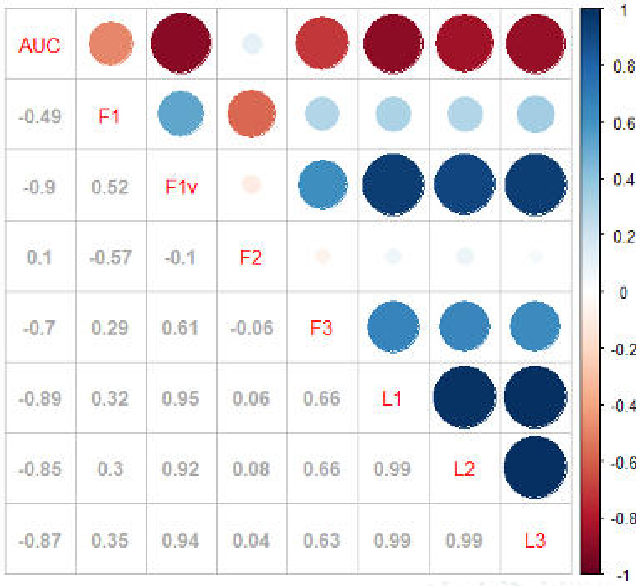
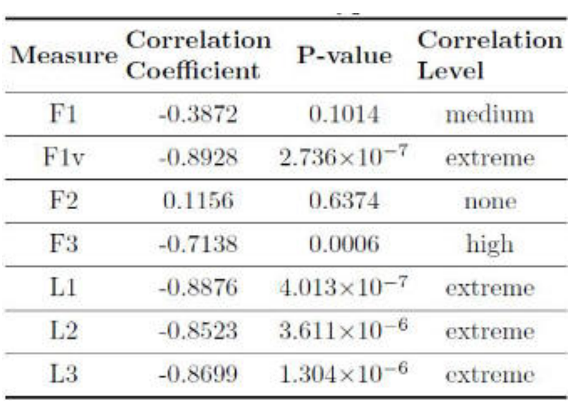


Figure 8 Correlation matrix.



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

Table 8: Results of hypothesis test

6. Improving Imbalanced Classification by Introducing Additional Attributes

Although the anomaly detection problem can be considered as an extreme case of class imbalance problem, very few studies consider improving class imbalance classification with anomaly detection ideas. Most data-level approaches in the imbalanced learning domain aim to introduce more information to the original dataset by generating synthetic samples. However, in this part of research, we gain additional information in another way, by introducing additional attributes. We propose to introduce the outlier score and four types of samples (safe, borderline, rare, outlier) as additional attributes in order to gain more information on the data characteristics and improve the classification performance. The proposed idea of introducing additional attributes is simple to implement and can be combined with resampling techniques and other algorithmic-level approaches in the imbalanced learning domain.

In the following, we will first review the related works in Section 4.1, including selected undersampling techniques, four types of samples in the imbalanced learning domain and the local outlier factor in the anomaly detection domain. After that, the details on the experiments are reported, including the experimental setup in Section 4.2 and experimental results and discussions in Section 4.3

6.1. Related Work

As mentioned above, we propose to introduce two additional attributes into the imbalanced datasets in order to gain more information on the data characteristics and improve the classification performance. Introducing additional attributes can be regarded as a data preprocessing method, which is independent of resampling techniques and algorithmic-level approaches, and can also be combined with these two approaches. In this section, the background knowledge related to our experiment is given, including resampling techniques (Section 4.1.1), the definition of four types of samples in the imbalance learning domain (Section 4.1.2) and the outlier score (Section 4.1.3).

6.1.1 *Resampling Techniques*

We experiment with two oversampling techniques, including SMOTE and ADASYN (given in previous section) and two undersampling techniques, including NCL and OSS.

Undersampling Techniques One-Sided Selection (OSS) is an undersampling technique which combines Tomek Links and the Condensed Nearest Neighbour (CNN) Rule **[1] [23]**. In OSS, noisy and borderline majority samples are removed with so-called Tomek links **[10]**. The safe majority samples which have limited contribution for building the decision boundary are then removed with CNN. Neighbourhood Cleaning Rule (NCL) emphasizes the quality of the retained majority class samples after data cleaning **[24]**. The cleaning process is first performed by removing ambiguous majority samples through Wilson's Edited Nearest Neighbour Rule (ENN) **[25]**. Then, the majority samples which have different labels from their three nearest neighbours are removed. Apart from this, if a minority sample has different labels from its three nearest neighbours, then the three neighbours are removed.

6.1.2 *Four Types of Samples in the Imbalance Learning Domain*

Napierala and Stefanowski proposed to analyse the local characteristics of minority examples by dividing them into four different types: safe, borderline, rare examples and outliers **[26]**. The identification of the type of an example can be done through modeling its k-neighbourhood. Considering that many applications involve both nominal and continuous attributes, the HVDM metric is applied to calculate the distance between different examples. Given the number of neighbours *k* (odd), the label to a minority example can be assigned through the ratio of the number of its minority neighbours to the total number of neighbours ($\boldsymbol{R}_{\frac{min}{all}}$) according to Table 9. The label for a majority all example can be assigned in a similar way.

Table 9: Rules to assign the four types of minority examples.

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

6.1.3 *Outlier Score*

Many algorithms have been developed to deal with anomaly detection problems and the experiments here are mainly performed with the nearest- neighbour based local outlier score (LOF). Local outlier factor (LOF), which indicates the degree of a sample being an outlier, was first introduced by Breunig et al. in 2000 **[27]**. The LOF of an object depends on its relative degree of isolation from its surrounding neighbours. Several definitions are needed to calculate the LOF and are summarized in the following Algorithm 1.

According to the definition of LOF, a value of approximately 1 indicates that the local density of data point $\boldsymbol{X_i}$ is similar to its neighbours. A value below 1 indicates that data point $\boldsymbol{X_i}$ 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 $\boldsymbol{X_i}$ is alienated from other points, which is most likely an outlier.

Algorithm 1: Local outlier factor (LOF) algorithm **[27].**

```
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 $X_i$

1  initialization;
2  calculate the distance $d(\cdot)$ between every two data points;
3  for $i = 1$ to $n$ do
4      calculate $k\text{-}distance(X_i)$: the distance between $X_i$ and its $k$th neighbour;
5      find out $k$-distance neighbourhood $N_k(X_i)$: the set of data points whose
         distance from $X_i$ is not greater than $k\text{-}distance(X_i)$;
6      for $j = 1$ to $n$ do
7          calculate reachability distance:
               $reach\text{-}dist_k(X_i, X_j) = \max\{k\text{-}distance(X_j), d(X_i, X_j)\};$
8          calculate local reachability density:
               $lrd_k(X_i) = 1/avg\text{-}reach\text{-}dist_k(X_i) = 1/\left(\frac{\sum_{o \in N_k(X_i)} reach\text{-}dist_k(X_i, X_j)}{|N_k(X_i)|}\right);$
           intuitively, the local reachability density of $X_i$ is the inverse of the
           average reachability distance based on the $k$-nearest neighbours of $X_i$;
9          calculate LOF:
               $LOF_k(X_i) = \frac{\sum_{o \in N_k(X_i)} lrd_k(X_j)}{|N_k(X_i)| \cdot lrd_k(X_i)} = \frac{\sum_{o \in N_k(X_i)} \frac{lrd_k(X_j)}{lrd_k(X_i)}}{|N_k(X_i)|}$
           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 $X_i$.
10     end
11 end
```

6.2. Experimental Setup

In this part of research, we propose to introduce the four types of samples and the outlier score as additional attributes of the original imbalanced dataset, where the former can be expressed as $\boldsymbol{R}_{\frac{min}{all}}$ and the latter can be calculated through Python library *PyOD* **[28]**. The experiments reported here are based on 7 two-class imbalanced datasets, including 6 imbalanced benchmark datasets (given in Table 10) and a 2D imbalanced chess dataset, which is commonly used for visualizing the effectiveness of the selected techniques in the imbalanced learning domain **[1]**. For each dataset, we consider four scenarios, whether to perform resampling techniques on the original datasets and whether to perform resampling techniques on the datasets with additional attributes. For each scenario of each dataset, we repeat the experiments 30 times with different random seeds. After that, the paired t-tests were performed on each of the 30 performance metric values to test if there is significant difference between the results of each scenario on a 5% significance level. Each collected dataset is divided into 5 stratified folds (for cross-validation) and only the training set is oversampled, where the stratified fold is to ensure that the imbalance ratio in the training set is consistent with the original dataset and only oversampling the training set is to avoid over-optimism problem **[5]**.

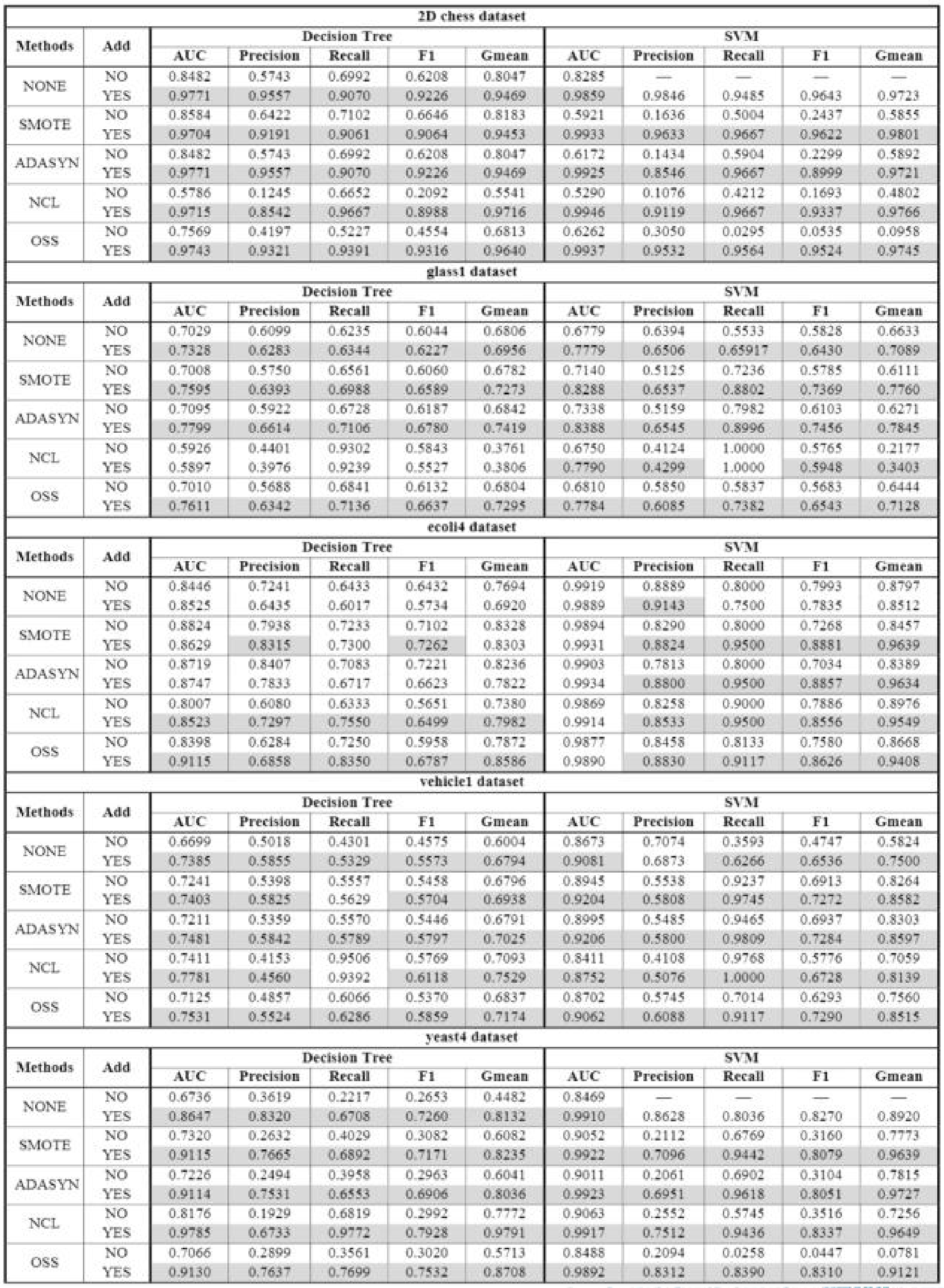
Table 10: Information on benchmark datasets [6].

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

6.3. Experimental Results and Discussion

Like other studies **[9]** **[4]**, we also use **SVM** and **Decision Tree** as the base classifiers in our experiments to compare the performance of the proposed method and the existing methods. Please note that we did not tune the hyperparameters for the classification algorithms and the resampling techniques **[22]**. The experimental results with the two additional attributes (four types of samples and LOF score) are presented in Table 11. We can observe that introducing outlier score and four types of samples as additional attributes can significantly improve the imbalanced classification performance in most cases. For 5 out of 7 datasets (*2D chess dataset, glass1, yeast4, wine quality and page block*), only introducing additional attributes (with no resampling) gives better results than performing resampling techniques.

*Table 11: Experimental results for two classification algorithms (with LOF score). "**Add** = YES" means we introduce additional attributes to the original datasets. "---" means TP+FN=0 or TP+FP=0 and performance metric cannot be computed.*



2D chess dataset

Methods	Add	Decision Tree					SVM				
		AUC	Precision	Recall	F1	Gmean	AUC	Precision	Recall	F1	Gmean
NONE	NO	0.8482	0.5743	0.6992	0.6208	0.8047	0.8285	—	—	—	—
	YES	0.9771	0.9557	0.9070	0.9226	0.9469	0.9859	0.9846	0.9485	0.9643	0.9723
SMOTE	NO	0.8584	0.6422	0.7102	0.6646	0.8183	0.5921	0.1636	0.5004	0.2437	0.5855
	YES	0.9704	0.9191	0.9061	0.9064	0.9453	0.9933	0.9633	0.9667	0.9622	0.9801
ADASYN	NO	0.8482	0.5743	0.6992	0.6208	0.8047	0.6172	0.1434	0.5904	0.2299	0.5892
	YES	0.9771	0.9557	0.9070	0.9226	0.9469	0.9925	0.8546	0.9667	0.8999	0.9721
NCL	NO	0.5786	0.1245	0.6652	0.2092	0.5541	0.5290	0.1076	0.4212	0.1693	0.4802
	YES	0.9715	0.8542	0.9667	0.8988	0.9716	0.9946	0.9119	0.9667	0.9337	0.9766
OSS	NO	0.7569	0.4197	0.5227	0.4554	0.6813	0.6262	0.3050	0.0295	0.0535	0.0958
	YES	0.9743	0.9321	0.9391	0.9316	0.9640	0.9937	0.9532	0.9564	0.9524	0.9745

glass1 dataset

Methods	Add	Decision Tree					SVM				
		AUC	Precision	Recall	F1	Gmean	AUC	Precision	Recall	F1	Gmean
NONE	NO	0.7029	0.6099	0.6235	0.6044	0.6806	0.6779	0.6394	0.5533	0.5828	0.6633
	YES	0.7328	0.6283	0.6344	0.6227	0.6956	0.7779	0.6506	0.65917	0.6430	0.7089
SMOTE	NO	0.7008	0.5750	0.6561	0.6060	0.6782	0.7140	0.5125	0.7236	0.5785	0.6111
	YES	0.7595	0.6393	0.6988	0.6589	0.7273	0.8288	0.6537	0.8802	0.7369	0.7760
ADASYN	NO	0.7095	0.5922	0.6728	0.6187	0.6842	0.7338	0.5159	0.7982	0.6103	0.6271
	YES	0.7799	0.6614	0.7106	0.6780	0.7419	0.8388	0.6545	0.8996	0.7456	0.7845
NCL	NO	0.5926	0.4401	0.9302	0.5843	0.3761	0.6750	0.4124	1.0000	0.5765	0.2177
	YES	0.5897	0.3976	0.9239	0.5527	0.3806	0.7790	0.4299	1.0000	0.5948	0.3403
OSS	NO	0.7010	0.5688	0.6841	0.6132	0.6804	0.6810	0.5850	0.5837	0.5683	0.6444
	YES	0.7611	0.6342	0.7136	0.6637	0.7295	0.7784	0.6085	0.7382	0.6543	0.7128

ecoli4 dataset

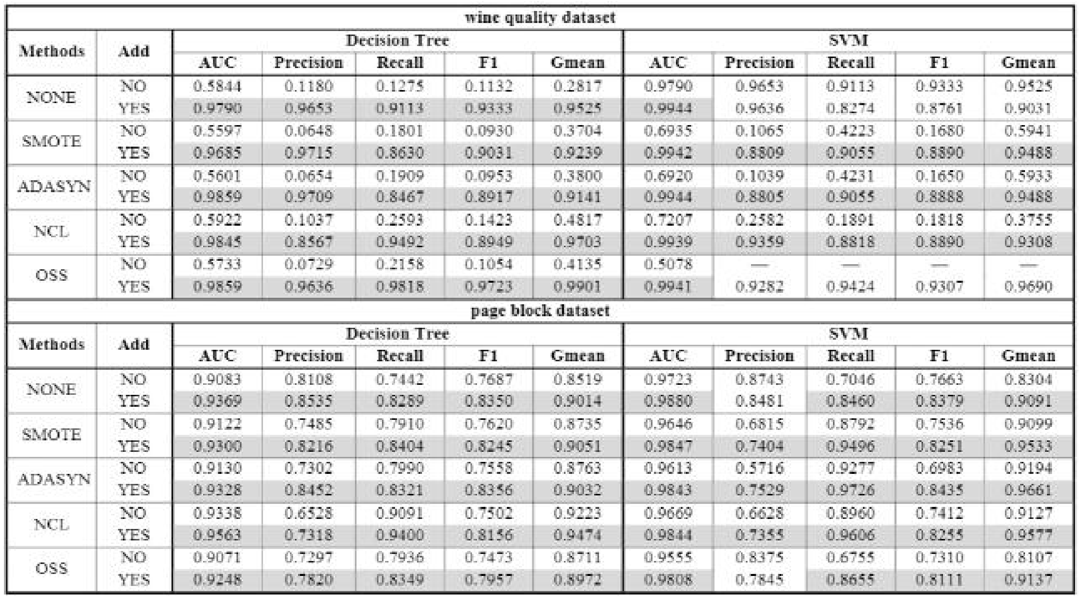
Methods	Add	Decision Tree					SVM				
		AUC	Precision	Recall	F1	Gmean	AUC	Precision	Recall	F1	Gmean
NONE	NO	0.8446	0.7241	0.6433	0.6432	0.7694	0.9919	0.8889	0.8000	0.7993	0.8797
	YES	0.8525	0.6435	0.6017	0.5734	0.6920	0.9889	0.9143	0.7500	0.7835	0.8512
SMOTE	NO	0.8824	0.7938	0.7233	0.7102	0.8328	0.9894	0.8290	0.8000	0.7268	0.8457
	YES	0.8629	0.8315	0.7300	0.7262	0.8303	0.9931	0.8824	0.9500	0.8881	0.9639
ADASYN	NO	0.8719	0.8407	0.7083	0.7221	0.8236	0.9903	0.7813	0.8000	0.7034	0.8389
	YES	0.8747	0.7833	0.6717	0.6623	0.7822	0.9934	0.8800	0.9500	0.8857	0.9634
NCL	NO	0.8007	0.6080	0.6333	0.5651	0.7380	0.9869	0.8258	0.9000	0.7886	0.8976
	YES	0.8523	0.7297	0.7550	0.6499	0.7982	0.9914	0.8533	0.9500	0.8556	0.9549
OSS	NO	0.8398	0.6284	0.7250	0.5958	0.7872	0.9877	0.8458	0.8133	0.7580	0.8668
	YES	0.9115	0.6858	0.8350	0.6787	0.8586	0.9890	0.8830	0.9117	0.8626	0.9408

vehicle1 dataset

Methods	Add	Decision Tree					SVM				
		AUC	Precision	Recall	F1	Gmean	AUC	Precision	Recall	F1	Gmean
NONE	NO	0.6699	0.5018	0.4301	0.4575	0.6004	0.8673	0.7074	0.3593	0.4747	0.5824
	YES	0.7385	0.5855	0.5329	0.5573	0.6794	0.9081	0.6873	0.6266	0.6536	0.7500
SMOTE	NO	0.7241	0.5398	0.5557	0.5458	0.6796	0.8945	0.5538	0.9237	0.6913	0.8264
	YES	0.7403	0.5825	0.5629	0.5704	0.6938	0.9204	0.5808	0.9745	0.7272	0.8582
ADASYN	NO	0.7211	0.5359	0.5570	0.5446	0.6791	0.8995	0.5485	0.9465	0.6937	0.8303
	YES	0.7481	0.5842	0.5789	0.5797	0.7025	0.9206	0.5800	0.9809	0.7284	0.8597
NCL	NO	0.7411	0.4153	0.9506	0.5769	0.7093	0.8411	0.4108	0.9768	0.5776	0.7059
	YES	0.7781	0.4560	0.9392	0.6118	0.7529	0.8752	0.5076	1.0000	0.6728	0.8139
OSS	NO	0.7125	0.4857	0.6066	0.5370	0.6837	0.8702	0.5745	0.7014	0.6293	0.7560
	YES	0.7531	0.5524	0.6286	0.5859	0.7174	0.9062	0.6088	0.9117	0.7290	0.8515

yeast4 dataset

Methods	Add	Decision Tree					SVM				
		AUC	Precision	Recall	F1	Gmean	AUC	Precision	Recall	F1	Gmean
NONE	NO	0.6736	0.3619	0.2217	0.2653	0.4482	0.8469	—	—	—	—
	YES	0.8647	0.8320	0.6708	0.7260	0.8132	0.9910	0.8628	0.8036	0.8270	0.8920
SMOTE	NO	0.7320	0.2632	0.4029	0.3082	0.6082	0.9052	0.2112	0.6769	0.3160	0.7773
	YES	0.9115	0.7665	0.6892	0.7171	0.8235	0.9922	0.7096	0.9442	0.8079	0.9639
ADASYN	NO	0.7226	0.2494	0.3958	0.2963	0.6041	0.9011	0.2061	0.6902	0.3104	0.7815
	YES	0.9114	0.7531	0.6553	0.6906	0.8036	0.9923	0.6951	0.9618	0.8051	0.9727
NCL	NO	0.8176	0.1929	0.6819	0.2992	0.7772	0.9063	0.2552	0.5745	0.3516	0.7256
	YES	0.9785	0.6733	0.9772	0.7928	0.9791	0.9917	0.7512	0.9436	0.8337	0.9649
OSS	NO	0.7066	0.2899	0.3561	0.3020	0.5713	0.8488	0.2094	0.0258	0.0447	0.0781
	YES	0.9130	0.7637	0.7699	0.7532	0.8708	0.9892	0.8312	0.8390	0.8310	0.9121



wine quality dataset											
Methods	Add	Decision Tree					SVM				
		AUC	Precision	Recall	F1	Gmean	AUC	Precision	Recall	F1	Gmean
NONE	NO	0.5844	0.1180	0.1275	0.1132	0.2817	0.9790	0.9653	0.9113	0.9333	0.9525
	YES	0.9790	0.9653	0.9113	0.9333	0.9525	0.9944	0.9636	0.8274	0.8761	0.9031
SMOTE	NO	0.5597	0.0648	0.1801	0.0930	0.3704	0.6935	0.1065	0.4223	0.1680	0.5941
	YES	0.9685	0.9715	0.8630	0.9031	0.9239	0.9942	0.8809	0.9055	0.8890	0.9488
ADASYN	NO	0.5601	0.0654	0.1909	0.0953	0.3800	0.6920	0.1039	0.4231	0.1650	0.5933
	YES	0.9859	0.9709	0.8467	0.8917	0.9141	0.9944	0.8805	0.9055	0.8888	0.9488
NCL	NO	0.5922	0.1037	0.2593	0.1423	0.4817	0.7207	0.2582	0.1891	0.1818	0.3755
	YES	0.9845	0.8567	0.9492	0.8949	0.9703	0.9939	0.9359	0.8818	0.8890	0.9308
OSS	NO	0.5733	0.0729	0.2158	0.1054	0.4135	0.5078	—	—	—	—
	YES	0.9859	0.9636	0.9818	0.9723	0.9901	0.9941	0.9282	0.9424	0.9307	0.9690

page block dataset											
Methods	Add	Decision Tree					SVM				
		AUC	Precision	Recall	F1	Gmean	AUC	Precision	Recall	F1	Gmean
NONE	NO	0.9083	0.8108	0.7442	0.7687	0.8519	0.9723	0.8743	0.7046	0.7663	0.8304
	YES	0.9369	0.8535	0.8289	0.8350	0.9014	0.9880	0.8481	0.8460	0.8379	0.9091
SMOTE	NO	0.9122	0.7485	0.7910	0.7620	0.8735	0.9646	0.6815	0.8792	0.7536	0.9099
	YES	0.9300	0.8216	0.8404	0.8245	0.9051	0.9847	0.7404	0.9496	0.8251	0.9533
ADASYN	NO	0.9130	0.7302	0.7990	0.7558	0.8763	0.9613	0.5716	0.9277	0.6983	0.9194
	YES	0.9328	0.8452	0.8321	0.8356	0.9032	0.9843	0.7529	0.9726	0.8435	0.9661
NCL	NO	0.9338	0.6528	0.9091	0.7502	0.9223	0.9669	0.6628	0.8960	0.7412	0.9127
	YES	0.9563	0.7318	0.9400	0.8156	0.9474	0.9844	0.7355	0.9606	0.8255	0.9577
OSS	NO	0.9071	0.7297	0.7936	0.7473	0.8711	0.9555	0.8375	0.6755	0.7310	0.8107
	YES	0.9248	0.7820	0.8349	0.7957	0.8972	0.9808	0.7845	0.8655	0.8111	0.9137

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

Figure **9** in order to further study the reason for the performance improvement.

3. From both the experimental results in Table 11 and the visualisation in

Figure **9**, 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.

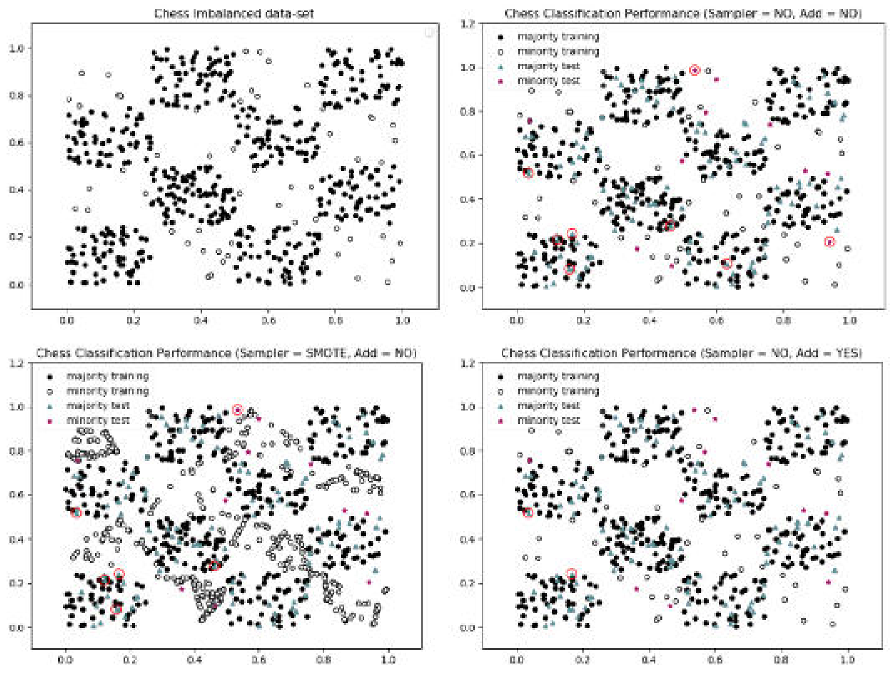


Figure 9: [top left]. Original imbalanced 2D chess dataset. [top right]. Classification performance for original chess dataset. The red-circled points indicate the misclassified points. [bottom left]. Classification performance for SMOTE-sampled chess dataset. [bottom right]. Classification performance for chess dataset with additional attributes.

7. Summary and Outlook

In this report, two aspects of class imbalance studies have been described: learning data complexity and improving imbalanced classification by introducing additional attributes. We provided details on hyperparameter optimisation on class imbalance classification and studied the empirical investigation on several state-of-the-art resampling techniques. The extended work on Section 3.2 "a potential application on the real-world inspired imbalanced vehicle mesh dataset" has already been introduced in the previous deliverable (D1.2). We reported the experimental results on the proposed idea of improving imbalanced classification by introducing additional attributes. The main conclusions that can be derived from our studies can be summarized as below:

- Oversampling techniques that consider the minority class distribution (new ones) perform better in most cases and the $\boldsymbol{F1v}$ value, a measure for evaluating the overlap which most researchers ignore, has a strong negative correlation with the potential AUC value.
- Applying hyperparameter optimisation for both classification algorithms and resampling approaches can produce the best results for classifying the imbalanced datasets. Furthermore, data complexity, especially the overlap between classes, has a big impact on the potential improvement that can be achieved through hyperparameter optimisation.
- Introducing 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)

Future work will cover the application of classifying the mesh prototypes in the car industry and extend our anomaly detection related work to online learning.

Bibliography

[1] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk and F. Herrera, Learning from imbalanced data sets, vol. 11, Springer, 2018.

[2] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence,* vol. 5, p. 221–232, 2016.

[3] G. E. A. P. A. Batista, R. C. Prati and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter,* vol. 6, p. 20–29, 2004.

[4] V. López, A. Fernández, S. García, V. Palade and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information sciences,* vol. 250, p. 113–141, 2013.

[5] M. S. Santos, J. P. Soares, P. H. Abreu, H. Araujo and J. Santos, "Cross-validation for imbalanced datasets: Avoiding overoptimistic and overfitting approaches [research frontier]," *ieee ComputatioNal iNtelligeNCe magaziNe,* vol. 13, p. 59–76, 2018.

[6] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez and F. Herrera, "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework.," *Journal of Multiple-Valued Logic & Soft Computing,* vol. 17, 2011.

[7] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research,* vol. 16, p. 321–357, 2002.

[8] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering,* vol. 21, p. 1263–1284, 2009.

[9] H. He, Y. Bai, E. A. Garcia and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, 2008.

[10] I. Tomek and others, "Two modifications of CNN.," 1976.

[11] A. C. Lorena, L. P. F. Garcia, J. Lehmann, M. C. P. Souto and T. K. Ho, "How complex is your classification problem? a survey on measuring classification complexity," *ACM Computing Surveys (CSUR),* vol. 52, p. 1–34, 2019.

[12] J. Bergstra, R. Bardenet, Y. Bengio and B. Kégl, "Algorithms for hyper-parameter optimization," in *25th annual conference on neural information processing systems (NIPS 2011)*, 2011.

[13] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Transactions on knowledge and Data Engineering,* vol. 17, p. 299–310, 2005.

[14] R. Barandela, J. S. Sánchez, V. Garca and E. Rangel, "Strategies for learning in class imbalance problems," *Pattern Recognition,* vol. 36, p. 849–851, 2003.

[15] J. Kong, T. Rios, W. Kowalczyk, S. Menzel and T. Bäck, "On the performance of oversampling techniques for class imbalance problems," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2020.

[16] S. Barua, M. M. Islam, X. Yao and K. Murase, "MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Transactions on knowledge and data engineering,* vol. 26, p. 405–425, 2012.

[17] B. Das, N. C. Krishnan and D. J. Cook, "RACOG and wRACOG: Two probabilistic oversampling techniques," *IEEE transactions on knowledge and data engineering,* vol. 27, p. 222–234, 2014.

[18] H. Zhang and M. Li, "RWO-Sampling: A random walk over-sampling approach to imbalanced data classification," *Information Fusion,* vol. 20, p. 99–116, 2014.

[19] T. K. Ho and M. Basu, "Complexity measures of supervised classification problems," *IEEE transactions on pattern analysis and machine intelligence,* vol. 24, p. 289–300, 2002.

[20] A. Orriols-Puig, N. Macia and T. K. Ho, "Documentation for the data complexity library in C++," *Universitat Ramon Llull, La Salle,* vol. 196, p. 1–40, 2010.

[21] I. Cordón, S. García, A. Fernández and F. Herrera, "Imbalance: Oversampling algorithms for imbalanced classification in R," *Knowledge-Based Systems,* vol. 161, p. 329–341, 2018.

[22] J. Kong, W. Kowalczyk, D. A. Nguyen, T. Bäck and S. Menzel, "Hyperparameter optimisation for improving classification under class imbalance," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019.

[23] M. Kubat, S. Matwin and others, "Addressing the curse of imbalanced training sets: one-sided selection," in *Icml*, 1997.

[24] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," in *Conference on Artificial Intelligence in Medicine in Europe*, 2001.

[25] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics,* p. 408–421, 1972.

[26] K. Napierala and J. Stefanowski, "Types of minority class examples and their influence on learning classifiers from imbalanced data," *Journal of Intelligent Information Systems,* vol. 46, p. 563–597, 2016.

[27] M. M. Breunig, H.-P. Kriegel, R. T. Ng and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000.

[28] Y. Zhao, Z. Nasrullah and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *arXiv preprint arXiv:1901.01588,* 2019.