Efficient tuning of automated machine learning pipelines

Nguyen, D.A.

CHAPTER 5

An Empirical Investigation Comparing CASH Optimization Approaches for Class Imbalance Problems

This imbalanced classification problem is relevant to both academic and industrial applications. The task of finding the best machine-learning model to use for a specific imbalanced dataset is complicated because of the large number of existing algorithms, each with its own hyperparameters. In this chapter, we study ML pipeline optimization in detail in the class imbalance domain, where the best combination of resampling techniques and classification algorithms is searched for, together with their optimized hyperparameters. The Combined Algorithm Selection and Hyperparameter Optimization (CASH) has been introduced to solve ML pipeline optimization problem by converting the problem into a hyperparameter optimization problem. We experimented with the first experiment (see Section 4.2), i.e., a search space of 5 classification algorithms, 21 resampling approaches and 64 relevant hyperparameters. Moreover, we investigated the performance of two well-known optimization approaches: random search and the Tree Parzen Estimator approach, which is a type of Bayesian optimization. For comparison, we also performed a grid search for all combinations of resampling techniques and classification algorithms with their default hyperparameters. The remainder of this chapter is organized as follows. First, Section 5.1 shows the motivation and provides a brief introduction to our work. In Section 5.2, the relevant background knowledge on imbalance classification and hyperparameter optimization are provided, and Section 5.3 outlines the experimental setup. Experimental results are discussed in Section 5.4. Finally, the chapter is concluded, and further work is outlined in Section 5.5.

5.1 Introduction

The imbalanced classification problem has garnered increasing attention from both academic and industrial fields. Technically, any dataset with an unequal class distribution is imbalanced. However, only datasets with a significantly skewed distribution are traditionally regarded as imbalanced in the learning domain [209]. Academic researchers aim to propose novel algorithms to handle imbalanced classification problems in different scenarios, for example, resampling techniques and algorithm-level approaches, whereas industrial researchers focus on improving imbalanced classification performances for specific real-life applications, such as fault diagnosis or anomaly detection [210], [211].

The combination of resampling techniques and classification algorithms is the most commonly used approach for handling imbalanced data [74], [212]. This leads to a challenge for an imbalanced classification problem on how to choose the best model (i.e., a combination of a resampling method) and a classifier (the so-called model selection problem or algorithm selection problem [213]) and their optimized hyperparameters [74] to achieve the best performance. This is a case of ML pipeline optimization where two tasks have to be considered in this chapter: model selection (MS) and hyperparameter optimization (HPO). Typically, these tasks are addressed separately and sequentially [14], [46], where the practitioner can choose to handle either task first. Generally, practitioners proceed by tuning the hyperparameters for each modeling algorithm separately and then choosing the best model. However, this approach is considerably more expensive due to a high number of possible combination operations.

Alternatively, the practitioner can select a suitable model by training all models with their default hyperparameters or based on experience, and then tune the hyperparameters only for the best model. This approach might get stuck in a local optimum of the model that was initially chosen based on the default hyperparameter setting. On the other hand, instead of sequentially solving these problems, they can be combined into a single problem and solved simultaneously. This approach is commonly referred to as the Combined Algorithm Selection and Hyperparameter Optimization (CASH) [40] or Full Model Selection (FMS) [44] approach.

Approaches for tackling the CASH problem have been widely proposed in the machine learning domain, particularly in the context of automated machine learning (AutoML), such as, Auto-Weka [40], [41] and Auto-Sklearn [39], [41], TPOT [43], HyperOpt-Sklearn [42]. In addition, [14] demonstrated that

the CASH approach is competitive with the sequential approach and requires less computational effort. However, the CASH approach has not yet been studied in detail in the context of learning from unbalanced data.

Hence, in this study, we introduce CASH in the context of optimizing the machine learning pipeline of combined classification algorithms and resampling techniques for the class imbalance problem. We are particularly interested in studying which optimization approach for handling the CASH problem yields the best classification performance.

In the first experiment (see Section 4.2), we use two well-known optimization approaches – Random search and Bayesian optimization. Furthermore, we experiment with dropping the hyperparameter tuning and carrying out only the model selection (MS) part, as sometimes done by practitioners. Our results suggest the inferiority of such an approach and demonstrate that applying CASH optimization yields better performance, for all test cases considered. Moreover, we observe that the Bayesian optimization approach produces better results than Random search. Hence, we recommend using this approach for handling the CASH problem for the class-imbalanced classification problem.

5.2 Related Works

In this section, we first provide a brief introduction to imbalanced classification (Section 5.2.1) and the CASH problem (Section 5.2.2) studied in this chapter.

5.2.1 Imbalanced Classification

The main problem in imbalanced classification is that the number of samples of one class is much lower than that of other classes [209]. Herein, the one or more classes being underrepresented are called minority class(es) and the other class(es) are called majority classes.

It has been shown that both the data-level (resampling) approaches and algorithm-level approaches are efficient in handling class-imbalance problems [214]. The data-level approaches focus on producing balanced datasets based on the unbalanced original data, whereas the algorithmic-level approaches concentrate on adjusting classification algorithms to make them appropriate for the imbalanced datasets. In the imbalanced learning domain, resampling techniques can be further divided into three groups: under-resampling, over-resampling, and combine-resampling. Under-resampling balances the class distribution by removing majority

samples, for example, the TomekLinks [194], while over-resampling balances the class distribution via producing synthetic minority samples, e.g., SMOTE [17]. The combine-resampling integrates both removing the majority samples and creating synthetic minority samples in order to balance the class distribution, e.g., SMOTETomek [18].

Owing to recent developments in data storage and management, it has become possible for industry and engineering practitioners to collect a large amount of data in order to extract knowledge and acquire hidden insights. An application example may be illustrated in the field of computational design optimization [215], where product parameters are modified to generate digital prototypes and the performance is usually evaluated through numerical simulations which often require minutes to hours of computation time. Here, some parameter variations (minority number of designs) would result in effective and producible geometric shapes, but the given constraints are violated in the final step of optimization. In this case, applying proper imbalanced classification algorithms to the design parameters may save computation time.

The family of evolutionary under-resampling techniques (EUS) has proven to be powerful in handling instance reduction [216]. An EUS algorithm attempts to optimize the selected samples in the majority class by performing a binary search guided by an evolutionary algorithm [108], [110]. Results of the EUS and the most recent research studies in this family consist of EUS-Windowing (EUSW) [217], clustering-based surrogate model for EUS (EUSC) [218] and hybrid surrogate model for EUS (EUSHC) [187] are also compared with our approach in the followed section.

In the class imbalance domain, it is widely known that *accuracy* is a deceptive estimate of performance [74], [219]. Instead of *accuracy*, other metrics such as the area under the receiver operating characteristic (ROC) curve, F-measure, or geometric mean (GM) are commonly used to measure performance [220]. For comparison with previous studies [187], [218], we use GM as the performance evaluation metric (see Section 2.1.2.2).

5.2.2 The Combined Algorithm Selection and Hyperparameter Optimization (CASH) Approach

The Combined Algorithm Selection and Hyperparameter Optimization (CASH) [40] is a commonly used approach for solving the ML pipeline optimization problem by converting it into a hyperparameter optimization (HPO) problem. As we delve

into the ongoing discussion, it is essential to reference Chapter 1 (Section 1.1.2), where the CASH is extensively discussed. Throughout the ongoing discussion, we consistently employ the notations and problem definition introduced in that section for a comprehensive understanding. In the context of optimization, HPO is generally viewed as a black-box optimization problem, which aims at finding the global optimum λ^* of the hyperparameters, with respect to a real-valued loss function f. As a reminder, the CASH approach incorporates an additional hyperparameter $\boldsymbol{\lambda}^0$ to model the choice of algorithms for each operator.

As mentioned in Section 5.1, we use a combination of resampling and classification algorithms to handle the class-imbalanced problem. Hence, the search space includes a set of resampling techniques, a set of classification algorithms, and their hyperparameters. Let $\boldsymbol{\lambda}^0_{res} = \{\mathcal{A}^1_{res}, \ldots, \mathcal{A}^{n_r}_{res}, \varnothing\}$ and $\boldsymbol{\lambda}^0_{cls} = \{\mathcal{A}^1_{cls}, \ldots, \mathcal{A}^{n_c}_{cls}\}$ denote sets of possible choice of resampling and classification algorithms, correspondingly. In practice, the use of a resampling technique is optional, we hence add a choice of not using any resampler, i.e., represented by $\varnothing$. Let $\Lambda_{res} = \boldsymbol{\lambda}^0_{res} \cup \Lambda^1_{res} \cup \ldots \cup \Lambda^{n_r}_{res}$ and $\Lambda_{cls} = \boldsymbol{\lambda}^0_{cls} \cup \Lambda^1_{cls} \cup \ldots \cup \Lambda^{n_c}_{cls}$ represent the hyperparameter spaces of resampling and classification operators. Hence, the entire search space for this particular problem is denoted by Λ, which includes Λ_{res} and Λ_{cls}. The ML pipeline optimization problem becomes the HPO maximizing problem:

$$\lambda^* = \arg\max_{\lambda \in \Lambda} f(\lambda) \ , \tag{5.1}$$

Note that in practice, most HPO methods can handle the CASH problem by modeling the choice of algorithms as a categorical hyperparameter. Each algorithm is mapped to its locally dependent hyperparameters by the so-called conditional parameter (see hierarchical hyperparameter in Table 7.1).

The HPO algorithms chosen in this study include Grid Search (see Section 3.1.1), Random Search (see Section 3.1.2) and a Bayesian optimization variant, namely Tree Parzen Estimators approach (TPE) (see Section 3.1.3).

5.3 Experimental Setup

This study reports and discusses the first experiment that has been introduced in detail in Section 4.2 including the search space, datasets and ML algorithms. Therefore, we only provide some additional information.

Random search and Bayesian optimization algorithms implemented in the Python package HyperOpt[1] are used as HPO algorithms. Based on the initial experiments, we set the number of iterations of HPO to 500, after which the algorithms have shown no significant improvements.

Moreover, to study the effectiveness of the HPO algorithms, we evaluated all possible combinations of classification and resampling algorithms with their default hyperparameter settings, i.e., dropping hyperparameter tuning and carrying out only the model selection part, as sometimes done by practitioners. For each dataset, we reported the combination with the highest GM value. The considered search space includes 5 classification algorithms and 21 resampling techniques, resulting in $5 \times 21 = 105$ combinations. Evaluating these combinations individually is referred to as "Grid-Def" here (grid search HPO algorithm).

The experiment scripts for the reproducibility of the reported results are provided in a git-repository[2].

5.4 Results and discussion

In this section, we report the results and discuss our insights. The experimental results are summarized in Table 5.1 to illustrate the performance differences between the three integrated optimization approaches used, i.e., TPE, Random search (RS) and Grid-Def (Grid), and to compare them with the state-of-the-art Evolutionary under-resampling (EUS) methods [218]. In this table, our results are presented in the corresponding columns on the left side (not shaded) and the results from [218] are presented on the right side (grey shaded) for EUS, EUSW, EUSC and EUSHC. In both groups, the highest performance for each dataset is highlighted in bold font. In our experimental results, the methods that perform significantly worse than the best according to the Wilcoxon signed-rank test with $\alpha = 0.05$ are underlined. A value labeled with * indicates that our results outperform those of [218] for the corresponding dataset. Additionally, an extra column to the right summarizes the method that achieved the highest GM for the corresponding dataset.

The results allow the following insights:

- HPO approaches exhibit better performance compared to the Grid-Def approach which uses static default hyperparameters. Moreover, according

[1] `https://github.com/hyperopt/hyperopt` (version 0.2.5).
[2] `https://github.com/ECOLE-ITN/NguyenDSAA2021`

Table 5.1: Average geometric mean (rounded to 4 decimals) over 10 repetitions for the 44 datasets, ordered by increasing IR value.

Dataset	#IR	Our experimental results			Evolutionary algorithms [218]				Overall Winner
		TPE	RS	Grid	EUS	EUSW	EUSC	EUSHC	
glass1	1.82	*0.7989	0.7763	0.7793	0.7773	0.7010	0.7941	0.7367	TPE
ecoli-0_vs_1	1.86	*0.9864	*0.9864	*0.9864	0.9583	0.9312	0.9581	0.9615	TPE | RS | Grid
wisconsin	1.86	*0.9814	*0.9807	*0.9788	0.9690	0.9652	0.9600	0.9590	TPE
pima	1.87	*0.7711	*0.7651	*0.7599	0.6943	0.6749	0.6957	0.7145	TPE
iris0	2.00	1	1	1	1	1	1	1	-
glass0	2.06	*0.8749	*0.8588	*0.8719	0.8009	0.6176	0.8047	0.6595	TPE
yeast1	2.46	*0.7324	*0.7304	*0.7183	0.6533	0.6501	0.6600	0.6600	TPE
haberman	2.78	*0.7025	*0.6926	*0.6678	0.5475	0.5635	0.5521	0.5497	TPE
vehicle2	2.88	*0.9915	*0.9874	*0.9895	0.9259	0.9175	0.9265	0.9173	TPE
vehicle1	2.90	*0.8658	*0.8429	*0.8333	0.6729	0.6624	0.6512	0.6926	TPE
vehicle3	2.99	*0.8482	*0.8231	*0.8108	0.7280	0.7142	0.7165	0.7204	TPE
glass-0-1-2-3 _vs_4-5-6	3.20	0.9559	0.9505	0.9483	0.9525	0.9385	0.9647	0.9546	EUSC
vehicle0	3.25	*0.9863	*0.9809	*0.9766	0.9164	0.9027	0.9103	0.9016	TPE
ecoli1	3.36	*0.9047	*0.8966	*0.8999	0.8634	0.8306	0.8554	0.8424	TPE
new-thyroid1	5.14	*0.9969	*0.9966	*0.9944	0.9882	0.9809	0.9859	0.9653	TPE
new-thyroid2	5.14	*0.9978	*0.9966	*0.9910	0.9865	0.9773	0.9831	0.9746	TPE
ecoli2	5.46	*0.9361	*0.9337	*0.9361	0.9000	0.8663	0.9034	0.8772	TPE | Grid
segment0	6.02	*0.9993	*0.9990	*0.9965	0.9881	0.9870	0.9876	0.9858	TPE
glass6	6.38	*0.9524	*0.9516	*0.9381	0.8889	0.9071	0.9156	0.9054	TPE
yeast3	8.10	*0.9428	*0.9395	*0.9290	0.8728	0.8740	0.8752	0.8550	TPE
ecoli3	8.60	*0.9061	*0.9023	*0.9044	0.8348	0.8153	0.8500	0.8097	TPE
page-blocks0	8.79	*0.9456	*0.9422	*0.9401	0.9117	0.9038	0.9096	0.9085	TPE
yeast-2_vs_4	9.08	*0.9559	*0.9474	*0.9401	0.9042	0.8774	0.9156	0.8930	TPE
yeast-0-5-6 -7-9_vs_4	9.35	*0.8212	*0.8063	*0.7938	0.7685	0.7663	0.7901	0.7535	TPE
vowel0	9.98	0.9581	0.9483	0.9427	0.9897	0.9719	0.9877	0.9831	EUS
glass-0-1-6_vs_2	10.29	*0.8498	*0.8216	*0.7904	0.6383	0.6164	0.6651	0.5815	TPE
glass2	11.59	*0.8516	*0.8271	*0.7903	0.7194	0.6525	0.7262	0.6173	TPE
shuttle-c0-vs-c4	13.87	*1	*1	*1	0.9960	0.9968	0.9960	0.9960	TPE | RS | Grid
yeast-1_vs_7	14.30	*0.8028	*0.7926	*0.7979	0.7176	0.7079	0.7068	0.6669	TPE
glass4	15.46	*0.9323	*0.9244	*0.9318	0.8700	0.8513	0.8613	0.8531	TPE
ecoli4	15.80	*0.9727	0.9551	0.9415	0.8984	0.9362	0.8857	0.9645	TPE
page-blocks -1-3_vs_4	15.86	*0.9925	*0.9877	*0.9884	0.9674	0.9399	0.9471	0.9294	TPE
abalone9-18	16.40	*0.8889	*0.8752	*0.8536	0.7269	0.6772	0.7224	0.6559	TPE
glass-0-1-6_vs_5	19.44	*0.9567	*0.9530	0.9304	0.9214	0.9151	0.9160	0.9501	TPE
shuttle-c2-vs-c4	20.50	*1	*1	*1	0.9577	0.6449	0.9414	0.7365	TPE | RS | Grid
yeast-1-4 -5-8_vs_7	22.10	*0.7035	*0.6874	*0.6650	0.6569	0.6088	0.6604	0.6149	TPE
glass5	22.78	*0.9637	0.9555	0.9438	0.8105	0.9076	0.9600	0.9103	TPE
yeast-2_vs_8	23.10	*0.8231	*0.8031	*0.7945	0.7931	0.7496	0.7656	0.7668	TPE
yeast4	28.10	*0.8803	*0.8664	*0.8585	0.8050	0.7799	0.8288	0.7970	TPE
yeast-1-2 -8-9_vs_7	30.57	*0.7459	*0.7402	*0.7289	0.6721	0.6078	0.6704	0.6500	TPE
yeast5	32.73	*0.9803	*0.9790	*0.9788	0.9634	0.9494	0.9455	0.9653	TPE
ecoli-0-1 -3-7_vs_2-6	39.14	*0.9095	*0.8770	*0.9091	0.6700	0.7048	0.6625	0.6865	TPE
yeast6	41.40	*0.8972	*0.8905	*0.8840	0.8357	0.8080	0.8034	0.8031	TPE
abalone19	129.44	*0.7967	*0.7942	*0.7579	0.6258	0.6061	0.7214	0.6556	TPE

to the results of the Wilcoxon signed-rank test, TPE was always the best method: it significantly outperforms the Grid-Def in 32/44 datasets, whereas it significantly outperforms RS in 26/44 tested cases

- Overall, TPE shows the highest GM for most of the datasets, 41/44. Other compared methods win on different datasets, for example, EUSC and EUS achieve the highest GM on "glass-0-1-2-3_vs_4-5-6" and "vowel0", respectively. All approaches obtained the maximum GM on dataset "iris0".

- Furthermore, based on our experimental results, we conclude that TPE wins over the methods from [218] on 41/44 datasets, RS – on 38/44 datasets and Grid-Def – on 37/44 datasets. This is surprising because the number of function evaluations used in our experiment is much smaller than in [218]: 500 function evaluations for TPE and RS, 105 function evaluations for Grid-Def vs 10.000 function evaluations for each method in [218]. A possible explanation for this might be that [218] employed a simple KNN rule with $k = 1$ as the mere classifier, whereas more complicated classification algorithms were used in our study. More precisely, according to our experimental results, the tuned KNN wins only in 11% (TPE), 13% (RS), and 9% (Grid-Def) of all cases.

To investigate the tuning behavior of the methods, we plot single runs of TPE and RS on the dataset "abalone9-18" in Figure 5.1. The scatter plots on the left show the observed GM values over 500 function evaluations. The six stacked histogram plots to the right describe the distribution of the observed values, in which the first plot shows all observed values, and the five last plots indicate the distributions for each of the classification algorithms, such as SVM, RF, KNN, LR and DT. We conclude that:

- Configurations generated by TPE can avoid infeasible parameters better than RS[3]. In this run, the number of errors occurring in the TPE and RS runs are 14 and 22, respectively. Based on all datasets and repetitions, the number of infeasible configurations encountered by TPE and RS are 4.4% and 5.9%, respectively.

- Apart from zero values, the GM values of TPE are mostly in the range from 0.8 to 0.9, while the GM of RS are distributed in the range from 0.6 to 0.7.

[3]Evaluations with infeasible combinations of parameters are marked in the figure as black dashes with $GM = 0$.

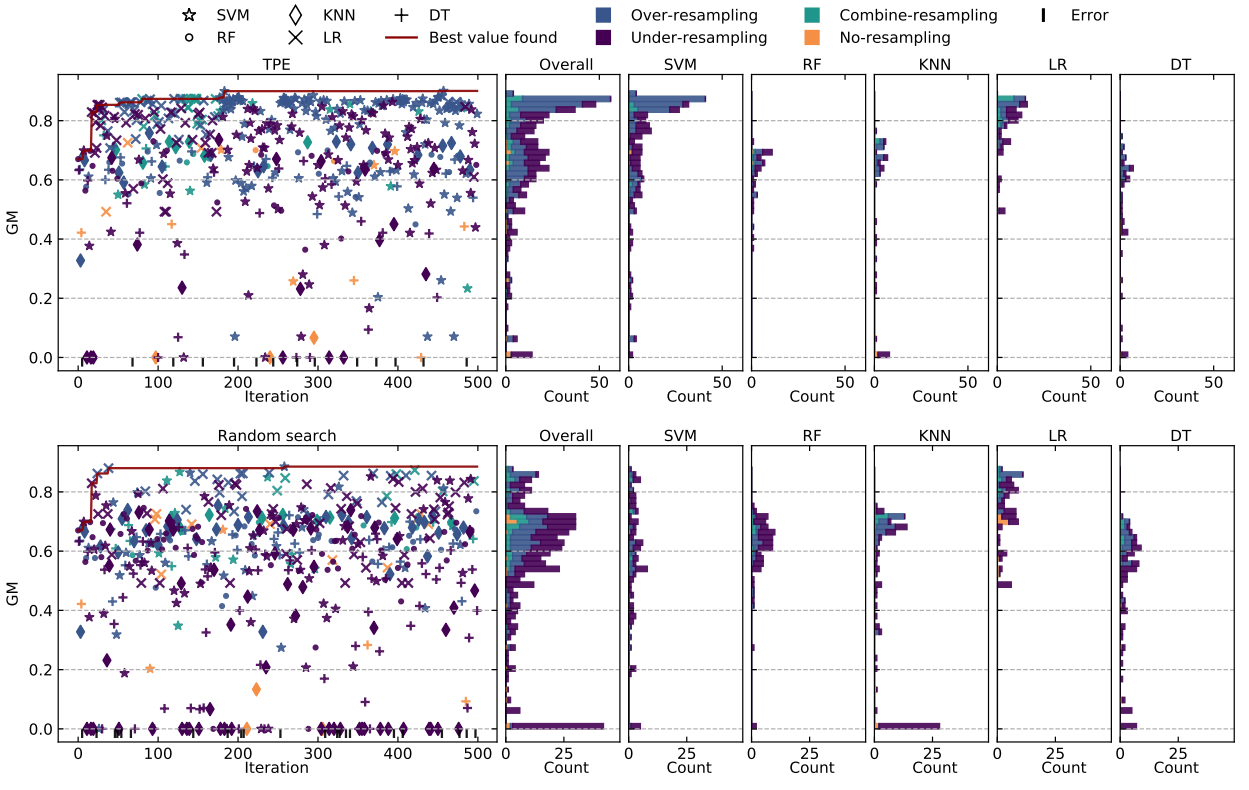

Figure 5.1: Illustration of the hyperparameter tuning process on dataset "abalone9-18" for TPE (top) and Random search (bottom). Scatter plots show the sequence of observed values of GM vs the number of function evaluations (iterations), the red line shows the current best value, and the black vertical bars on the horizontal axis indicate the infeasible configurations where GM returns to a zero if an invalid configuration is used. The stacked histogram plot next to the scatter plot shows the distribution of all observed values. The five last stacked histogram plots to the right indicate the distributions for each of the classification algorithms, namely SVM, RF, KNN, LR and DT.

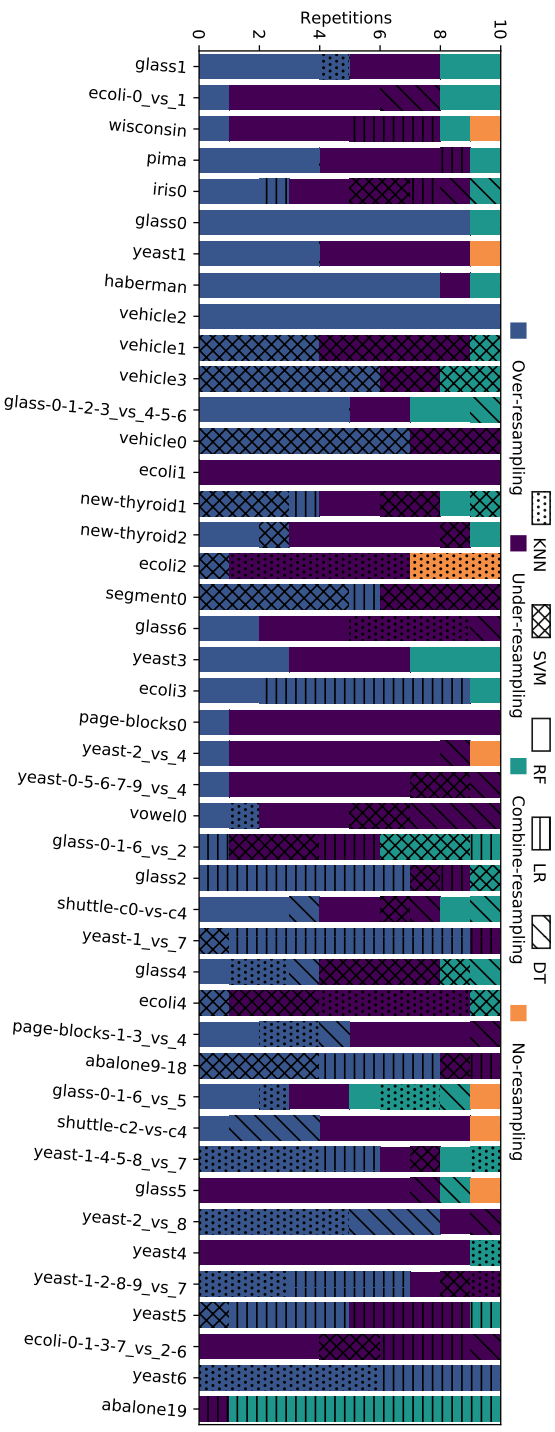

Figure 5.2: Resulting combination of classifier and resampler choices for an optimization process using TPE across 10 repetitions on 44 datasets.

Based on the highest results obtained by TPE, Figure 5.2 shows the final combination of choices of classification algorithms and resampling approaches once the optimization run is over. Clearly, no dominant algorithm exists over many datasets but different datasets benefit from different classification algorithms. For example, "glass0", "yeast1", "yeast3", "haberman", "vehicle2", "ecoli1" and "page-blocks0" achieve the best results with SVM, "vehicle0", "vehicle1", "vehicle3" with KNN, whereas "abalone19" always results in LR.

Besides, 98% of runs yield the best performance by using a resampling technique. Particularly, over-resampling, under-resampling and combine-resampling obtain 182, 199, 50 wins over $44 \times 10 = 440$ runs. Additionally, there is no classifier/resampler combination providing the best classification performance over all datasets. Specifically, RF and SVM obtain 206 and 84 wins, while other algorithms (LR, KNN, DT) find the best performance in 73, 48 and 29 runs.

5.5 Conclusions and Future Work

In this study, we applied a special type of Bayesian Optimization approach, the Tree Parzen Estimators to optimize the combined algorithm selection and hyperparameter optimization problem to improve the performance of classification algorithms for class imbalance problems. In other words, we propose an automated CASH optimization approach for imbalanced classification problems. Our approach automatically selects the best set of algorithms, i.e., the resampling technique and classification algorithm, together with their optimized hyperparameter settings for an arbitrary imbalanced dataset. The numeric results show significantly improved performance with respect to the state-of-the-art techniques in the imbalanced classification domain over 44 examined datasets.

Four main conclusions can be drawn from our experimental results:

1. Use of HPO clearly improves the classification performance compared to using static default parameters.

2. TPE outperforms the Random search on 91% of the tested datasets, while equal performance is found on the remaining cases.

3. Overall, the TPE approach produces the best results among other competitors in various scenarios. Hence, we recommend using TPE for handling CASH optimization in imbalanced classification problems.

4. Another finding was that 98% of runs yield the best performance with the help of resampling techniques. Thus we recommend to use resampling to handle class imbalanced problems.

There are several interesting research directions for extending this work. First, we intend to apply other Bayesian optimization variants such as SMAC, SPO, and MIPEGO, to study the performance of variants in this domain. Second, the scope of this study was limited in terms of classification problems; therefore, our future studies might extend the research for regression problems. Third, in addition to GM, other commonly used performance evaluation metrics in this domain will be investigated in our future work, including the Area Under the ROC Curve (AUC), F-measure, and recall. Fourth, the penalty-based methods, e.g., penalized-SVM, themselves can efficiently handle imbalanced datasets in several cases. Thus, we plan to study their effectiveness in the context of CASH optimization. Additionally, instead of applying hyperparameter tuning on the level of an individual dataset, we are interested in studying the behavior of HPO approaches when tuning for a set of datasets. Finally, besides Bayesian optimization, we shall extend our research with other state-of-the-art HPO approaches such as iRace [33] and Hyperband [35] for the class-imbalanced problem.