



International Journal of Marketing Management

ISSN 2454 - 5007



www.ijmm.net

Email ID: editor@ijmm.net , ijmm.editor9@gmail.com

Effectiveness of Data Preprocessing Techniques for Uneven Classification

¹ Sadia Kausar, ² Mohammed Ibrahim Ali, ³ Shaik Zayd Ahmed Farooqui, ⁴ Mohammed Misban Uddin

¹ Assistant Professor, Department of CSE-AIML, Lords Institute of Engineering & Technology.

^{2,3,4} Student Department of CSE-AIML, Lords Institute of Engineering & Technology.

Abstract—

Machine learning in cybersecurity is challenging due to many factors, one of which is the severe class imbalance. A variety of dataset preparation approaches have been introduced over the years. In order to enhance the prediction capabilities of classifiers trained on the training dataset, these techniques alter it by either oversampling or undersampling, or both. A thorough, objective benchmark evaluating the efficacy of various methodologies across a number of cybersecurity situations is lacking, despite their infrequent use in cybersecurity. This paper gives a comparison of 16 preprocessing approaches on six cybersecurity datasets coupled with 17 public imbalanced datasets from different fields. Using an AutoML system to train classifiers on the preprocessed datasets, we evaluate the approaches under numerous hyperparameter configurations. This lowers the possible bias from individual hyperparameter or classifier selections. The approaches are also evaluated with a focus on how well they mimic the actual functioning of cybersecurity systems in the real world by using suitable performance metrics. Our study's key takeaways are as follows: 1) There is usually a data preprocessing method that boosts classification performance. 2) Relative to most of the strategies in the benchmark, the baseline approach of doing nothing fared better. Thirdly, approaches that oversample tend to do better than those that undersample. 4) The typical SMOTE algorithm delivers the most noticeable performance advantages, whereas more complex approaches provide only incremental improvements, which are often accompanied by lower computing performance. Search Terms—AI, security, categorization, unbalanced categorization

I. INTRODUCTION

In cases when one class's class prior probability is much lower than another category's prior likelihood, we have a class-imbalanced classification problem. Class inequality is a problem in numerous machine learning-using industries, including astronomy [32], medicine [48], and finance [47] among 58 others. In the realm of cybersecurity in particular, almost all class-imbalanced scenarios, including intrusion, have been investigated. recognize phishing efforts [18] and harmful malware [13] as per [21]. The important class prior probability is that severe hostile conduct [13] creates a value of 10^{-5} or below, and there is typically a very substantial class imbalance. The good news is that attacks are rare. As an example, Regular, benign traffic accounts for the most majority of network telemetry recordings, with just a small fraction representing malicious activity. Curiously, there is still a disparity in class even within the limited set of data pertaining to harmful pursuits, as the frequency of low-risk pursuits such deceitful tracking and advertising significantly outweighs frequency and severity of the most intriguing dangers (e.g. malware, threats, and infections that infect remote access devices. The difficulties The first mention of cybersecurity occurred in the year 2000, according to Axelsson [7], which is concerning considering the severity of the problem of high social status disparity. After this, about twenty years later, we had a Raising the bar for applying machine learning to cybersecurity is a major factor in the widening income gap [5, 27]. Classifiers become unable to learn new information without appropriate protections and reliable outcomes when the social class difference reaches a certain threshold, even though a minor class gap is usually not a problem [31]. The reason for this is because classifiers often end up favoring the dominant group and ignoring the minority one. This might result in a situation where overall With a high rate of right predictions, the classifier achieves impressive accuracy. class all of the time. Other, more relevant outcomes measurements, however, do a poor job of reflecting students' overall performance. There has been an uneven amount of attention in the Classification problem from many individuals. There were a lot of alternative approaches proposed for use in building learning models across all machines. Here we are on the sixth stage: 1) data 1) steering, 2) acquiring knowledge, and 3) ensuring model accuracy. Phase one approaches are often called data-level strategies, whereas phase two tactics are called algorithmic-level procedures [34]. Several literature reviews that summarize the arguments made in one place For quite some time, the most popular approaches for each stage have been published in print.

II. RELATED WORK

There have been several publications of data preparation techniques that are appropriate for class-imbalanced learning; nevertheless, there is a dearth of benchmarks that include a wide variety of methodologies and datasets. Experimental assessment is always included in every publication that introduces a new approach, although the experiments are usually rather minor in scale. As an example, the method is only compared to SMOTE [16] and plain decision tree baseline in the paper that introduces ADASYN [30], which includes experiments on five datasets. However, there are papers that compare preprocessing approaches, although they typically only include oversampling methods. Even more so, the majority of these research only use a few of datasets [26], [3], and [10]. The exception to this rule is a research by Kov'acs [36], which compares a large number of approaches and uses a large number of datasets. On the other hand, it doesn't include any experiments related to cybersecurity and solely covers oversampling methods. Additionally, none of the research above performs as extensive a search in hyperparameters and successive classifier models as we do. The UNSW-NB15 [45] dataset was used by Wheelus et al. [59] to examine several preprocessing approaches in the cybersecurity sector. Using a feedforward neural network with a single hidden layer for classification, Bagui and Li [9] evaluated five different preprocessing approaches on six different network intrusion detection datasets. To our knowledge, there has not been a comprehensive comparison assessment of the most prevalent data preparation approaches outside of cybersecurity [1], [43], [2], [53], [8]. Finally, prior research also often uses a single number to summarize the outcomes of several strategies. This is the method's average rank or score across all datasets, as is typical. Instead of just numbers, we present density plots of rank distributions in this paper. Due of the considerable variation and overlap in the rankings among datasets, these charts provide a more comprehensive view.

III. BENCHMARKED METHODS

This section provides a synopsis of the benchmark's preparation techniques. We avoid detailed explanations in favor of citing the source papers because of space constraints.

Methods for Oversampling

One potential strategy for addressing the unbalanced classification issue is the use of oversampling approaches. Oversampling techniques primarily aim to alter the empirical distribution by augmenting the minority class's sample size. To achieve the target imbalance ratio, the empirical distribution is adjusted by either producing additional fake samples or replicating the current ones. Random Oversampling is the simplest method; it just does what it sounds like: it randomly repeats samples from the dataset.

The SMOTE approach was one of the earliest and is now one of the most used oversampling techniques for creating synthetic data samples [16]. It takes existing examples from the minority class and uses the line segments to produce new synthetic examples. On the other hand, SMOTE gives equal weight to all minority samples. It is unconcerned with the neighborhood of a minority sample and disregards past sample density. Many improvements to the original SMOTE algorithm have been suggested to address these issues. Our benchmark incorporates four of these improvements: BorderlineSMOTE [28], SVM SMOTE [46], KMeansSMOTE [38], and ADASYN [30]. In contrast to SMOTE, BorderlineSMOTE only chooses minority samples when half or more of the neighbors are members of the majority class. The underlying premise of this method is that categorization relies heavily on minority samples that are located near the so-called decision border and surrounded by larger majority samples. To identify minority samples close to the decision border, SVM SMOTE expands on the same concept but using the SVM algorithm rather than the kNN technique.



Fig. 1. High-level architecture of the benchmarking framework.

Tomek Links [56] is a data-cleaning strategy that removes noisy samples from the region around the decision border. A Tomek Link is formed when there is no sample x_k such that $d(x_i; x_k) \leq d(x_i; x_j)$ or $d(x_j; x_k) \leq d(x_i; x_j)$ and two points $x_i \in S_{min}$ and $x_j \in S_{maj}$ from different classes form a pair. The majority samples are divided into four categories using One Sided Selection (OSS) [37]: noise, borderline, redundant, and safe. OSS uses CNN to identify and eliminate samples that are duplicated. We then use Tomek Links to filter out any unwanted noise or samples that can be considered borderline, ensuring that only safe samples are retained. Since CNN has a tendency to keep noisy samples in the training set, Neighboring Cleaning Rule-NCL [39] aims to replace CNN with ENN to address one big issue with the OSS technique. On top of that, NCL removes minority samples from selected neighborhoods. A three-neighbor distance is calculated for every minority sample. Neighbors that incorrectly identify the minority sample will have their majority-class neighbors removed from the analysis. We conclude the benchmark with Cluster Centroids [49], the only technique that reflects undersampling strategies for prototype creation. The KMeans algorithm is employed to group samples from the majority class into clusters, and the center of each cluster is then used to generate new samples from the majority class.

IV. EXPERIMENT SETUP

We developed a benchmark framework to run trials with various preprocessing techniques on various datasets, presenting assessment metrics in an efficient and resilient manner. Figure 1 illustrates the fundamental concept of the framework. With each run, you'll find a new grid search instance of the hyperparameters, a dataset, and a preprocessing procedure. A fresh resampled training set is produced by applying a preprocessing procedure to the training portion of a dataset; this new set is then supplied to the AutoML component of the framework in each run. We employ a state-of-the-art AutoML framework Auto-Sklearn [23] for choosing, training and refining a classifier suitable for a given dataset. Detailed information regarding Auto-Sklearn is presented in Section IV-A1. Predictions are made using unseen examples from the test set after a classifier has been trained, and assessment scores are reported. Using five-fold cross-validation as the validation approach, we found, trained, and tuned the best-performing classifier on the training set using Auto-Sklearn IV-A1. Optimizing the ROC AUC score IV-B2 was the default setting for Auto-Sklearn. There was a 30-minute time limit for each run to train on public datasets, with 10 minutes allotted to a single machine learning model. There was no repetition of unsuccessful runs. The sizes of the private datasets meant that only five minutes of Auto-Sklearn time was enough, and no repeats were required. In order to get information about how well preprocessing approaches worked on datasets of varying sizes, we did not impose any time constraints on the preprocessing stage. First, there's AutoSklearn, a package that can automatically choose models and tune their hyperparameters [23]. Through the use of Auto-Sklearn, we are able to test out many models independently of one another. Auto-Sklearn outperformed its rival AutoML systems, so we went with it [23]. The second version of Auto-Sklearn, which included significant improvements [22], has been available since 2020, but we choose not to use it since it was still in the experimental phase when the trials were conducted. To further improve the system's efficiency, Auto-Sklearn incorporates meta-learning and ensemble construction, expanding upon previous AutoML systems that use the Bayesian optimiser. We provide a quick rundown of how everything fits together and offer feedback when we had to tweak Auto-Sklearn's behavior to have full command of the experiment.

Method	Hyperparameter Configurations
Baseline	1
Random Oversampling	2
SMOTE	4
Borderline SMOTE	16
SVM SMOTE	8
KMeans SMOTE	4
ADASYN	4
Random Undersampling	2
CNN	2
ENN	4
Repeated ENN	4
All KNN	4
Near Miss	12
Tomek Links	1
One-Sided Selection	2
NCL	8
Cluster Centroids	4
Σ	82

TABLE I
HYPERPARAMETER CONFIGURATIONS FOR PREPROCESSING METHODS. THE TABLE SHOWS THE NUMBER OF AVAILABLE HYPERPARAMETER CONFIGURATIONS IN THE BENCHMARK.

Name	Maj. Size	Min. Size	Imbalance
Asteroid	125,975	156	807.532
Credit Card Subset [17]	14,217	23	618.130
Credit Card [17]	284,315	492	577.876
PC2 [50]	5,566	23	242.000
MC1 [50]	9,398	68	138.206
Employee Turnover	33,958	494	68.741
Satellite [25]	5,025	75	67.000
BNG - Solar Flare	648,320	15,232	42.563
Mammography	10,923	260	42.012
Letter [24]	19,187	813	23.600
Relevant Images	129,149	5,582	23.137
Click Prediction V1	1,429,610	66,781	21.407
Click Prediction V2	142,949	6,690	21.368
Amazon Employee	30,872	1,897	16.274
BNG - Sick	938,761	61,239	15.329
Sylva Prior	13,509	886	15.247
BNG - Spect	915,437	84,563	10.826
CIC-IDS-2017 [51]	227,132	5,565	40.814
UNSW-NB15 [45]	164,673	9,300	17.707
CIC-Evasive-PDF [33]	4,468	555	8.050
Ember [4]	200,000	26,666	7.500
Graph - Embedding [20]	394	154	2.558
Graph - Raw [20]	394	154	2.558

TABLE II
DATASETS. THE TABLE SHOWS BASIC INFORMATION ABOUT THE DATASETS USED IN THE BENCHMARK. THE UPPER PART OF THE TABLE SHOWS PUBLICLY AVAILABLE NON-CYBERSECURITY DATASETS; THE LOWER PART SHOWS CYBERSECURITY DATASETS AND TWO PROPRIETARY DATASETS CONCERNING THE CLASSIFICATION OF NODES IN NETWORK GRAPHS.

C. Restrictions The execution time of AutoSklearn and Imbalanced Learn's preprocessing methods in Python has been our biggest challenge. Despite our earlier claims that we did not restrict the preprocessing step's duration, we had to cancel several runs since their calculation times were too lengthy. We terminated several preparation method/dataset combinations after they had run for almost two days. Consequently, not all 23 datasets were able to effectively complete all preprocessing techniques. If we wanted to do the calculation quickly and cheaply, we had to subsample the Ember dataset and use feature selection. A Random Forest classifier was used in conjunction with the Gini impurity measure to determine the 300 most essential characteristics in order of relevance. Another hurdle we

had to jump was the fact that the majority of the publicly accessible cybersecurity datasets either don't have any imbalance at all or have an imbalance that is much lower than the actual imbalance in the field. If we want to come closer to the true distribution, we should randomly select cybersecurity datasets. When it came to bigger cybersecurity datasets like CIC-IDS-2017 and Ember, this choice also greatly assisted us with the first issue. Even with subsampling, the imbalance ratios are still far lower than what is seen in real-world cybersecurity systems. The datasets, however, do not have enough data to subsample further to produce more realistic imbalances.

	PR AUC	ROC AUC	P-ROC AUC
Baseline	6.196 // 23	6.761 // 23	10.891 // 23
Random Oversampling	9.238 // 21	9.024 // 21	6.929 // 21
SMOTE	6.283 // 23	6.174 // 23	4.087 // 23
Borderline SMOTE	5.935 // 23	6.239 // 23	4.500 // 23
SVM SMOTE	4.841 // 22	4.909 // 22	3.545 // 22
KMeans SMOTE	4.500 // 04	2.625 // 04	4.000 // 04
ADASYN	7.955 // 22	7.977 // 22	5.386 // 22
Random Undersampling	10.318 // 22	9.818 // 22	5.773 // 22
CNN	11.964 // 14	11.821 // 14	9.036 // 14
ENN	6.310 // 21	6.452 // 21	9.524 // 21
Repeated ENN	7.222 // 18	6.583 // 18	11.056 // 18
All KNN	8.273 // 22	8.273 // 22	11.159 // 22
Near Miss	11.023 // 22	11.341 // 22	7.068 // 22
Tomek Links	7.667 // 21	7.857 // 21	11.810 // 21
One-Sided Selection	8.455 // 22	9.091 // 22	12.000 // 22
NCL	9.023 // 22	8.886 // 22	11.068 // 22
Cluster Centroids	10.706 // 17	10.235 // 17	6.529 // 17

TABLE III
MEAN RANK ACROSS ALL DATASETS. THE TABLE CONTAINS AVERAGE RANKS FOR EACH COMBINATION OF PREPROCESSING METHOD AND EVALUATION METRIC COMPUTED ACROSS ALL DATASETS. THE SECOND NUMBER AFTER // INDICATES THE NUMBER OF DATASETS USED TO COMPUTE THE AVERAGE.

V. RESULTS

On all datasets in the benchmark, the distribution of rankings generated for each preprocessing technique is shown in Figures 2, 3, and 4. We calculated ranks for each preprocessing technique independently using their respective measures. The lowest, highest, and mean ranks of each technique are shown by dark markings, while the 25th, 50th, and 75th percentiles of each method are denoted by three blue marks. Additionally, you may see the mean rankings in Table III. The cybersecurity datasets were the only ones used to calculate the mean rankings shown in Table IV. Because of space constraints and a limited sample size, we will not be displaying distributions for cybersecurity datasets. The SMOTE algorithm and its variations are examined in Table V, which also examines the techniques' relative differences. We left KMeansSMOTE off of Table V since there weren't enough successful runs. Figure 5 concludes the analysis by displaying the methods' runtime performance.

VI. DISCUSSION

Here we go further into the data and highlight the most crucial conclusions and suggestions. We begin by looking at the aggregated findings from every dataset. The second step is to check whether the results and suggestions are different when applied to cybersecurity datasets. Finally, we go over how well the strategies under study performed computationally. Consider first the results of the baseline technique, which uses the training dataset without any preprocessing at all. When compared to other approaches and across several metrics, the baseline method performed reasonably. The baseline approach consistently rated in the top half of the methods that were tested when looking at

the PR AUC and ROC AUC metrics (Figures 2 and 3). Figure 4 shows the P-ROC AUC measure; while the baseline method is never the poorest, it frequently ends up in the second half of the techniques. The effectiveness of the baseline

	PR AUC	ROC AUC	P-ROC AUC
Baseline	5.167 // 06	6.500 // 06	11.000 // 06
Random Oversampling	8.667 // 06	8.667 // 06	7.333 // 06
SMOTE	5.500 // 06	5.833 // 06	4.667 // 06
Borderline SMOTE	7.000 // 06	8.167 // 06	5.667 // 06
SVM SMOTE	5.000 // 06	5.000 // 06	4.500 // 06
KMeans SMOTE	3.500 // 02	1.250 // 02	1.000 // 02
ADASYN	7.500 // 06	8.583 // 06	5.667 // 06
Random Undersampling	11.667 // 06	11.167 // 06	6.167 // 06
CNN	14.667 // 03	12.000 // 03	10.667 // 03
ENN	8.000 // 06	6.750 // 06	10.667 // 06
Repeated ENN	5.800 // 05	5.800 // 05	9.800 // 05
All KNN	9.833 // 06	9.500 // 06	11.500 // 06
Near Miss	12.667 // 06	10.333 // 06	6.500 // 06
Tomek Links	7.333 // 06	8.833 // 06	12.167 // 06
One-Sided Selection	8.167 // 06	9.833 // 06	12.000 // 06
NCL	10.333 // 06	10.333 // 06	10.333 // 06
Cluster Centroids	9.000 // 04	8.125 // 04	6.250 // 04

TABLE IV
MEAN RANK ACROSS CYBERSECURITY DATASETS. THE TABLE CONTAINS AVERAGE RANKS FOR EACH COMBINATION OF PREPROCESSING METHOD AND EVALUATION METRIC COMPUTED ACROSS CYBERSECURITY DATASETS. THE SECOND NUMBER AFTER // INDICATES THE NUMBER OF DATASETS USED TO COMPUTE THE AVERAGE.

Method	Metric	Border. SMOTE	SVM SMOTE	ADASYN
SMOTE	PR AUC	0.007	-0.012	-0.018
	ROC AUC	-0.012	-0.018	-0.012
	P-ROC AUC	-0.003	-0.015	-0.006
Border. SMOTE	PR AUC	-	-0.020	-0.026
	ROC AUC	-	-0.006	-0.000
	P-ROC AUC	-	-0.013	-0.003
SVM SMOTE	PR AUC	-	-	-0.006
	ROC AUC	-	-	0.006
	P-ROC AUC	-	-	0.009

TABLE V
INCREMENTAL DIFFERENCES OF SMOTE VARIANTS COMPARED TO EACH OTHER. THE TABLE SHOWS THE GAINS/LOSSES OF DIFFERENT SMOTE VARIANTS COMPARED TO EACH OTHER. THE RESULTS ARE DIFFERENCES OF MEANS COMPUTED ACROSS MAXIMAL SCORES ATTAINED ON EACH DATASET. THE DIFFERENCES WERE COMPUTED BY SUBTRACTING THE MEAN SCORE OF A METHOD IN A COLUMN FROM THE MEAN SCORE OF A METHOD IN A ROW.

seems to be rather unexpected given that most approaches claim to improve performance for such issues. We put out a number of theories in an effort to clarify this finding. A number of datasets' summary statistics will be our starting point. Some techniques are designed for particular datasets with specific attributes and are not intended to be utilized in every case. One such method is Near Miss [42], which seeks to exclude samples located on the periphery of the dominant class. If these samples are mostly noise, then this strategy could work; however, if they are legitimate data, removing them might greatly raise the classifier's false positive rate. Second, we use AutoML, a far more robust baseline, to hyperparameter tune the classification layer. Finally, we are not a consulting firm

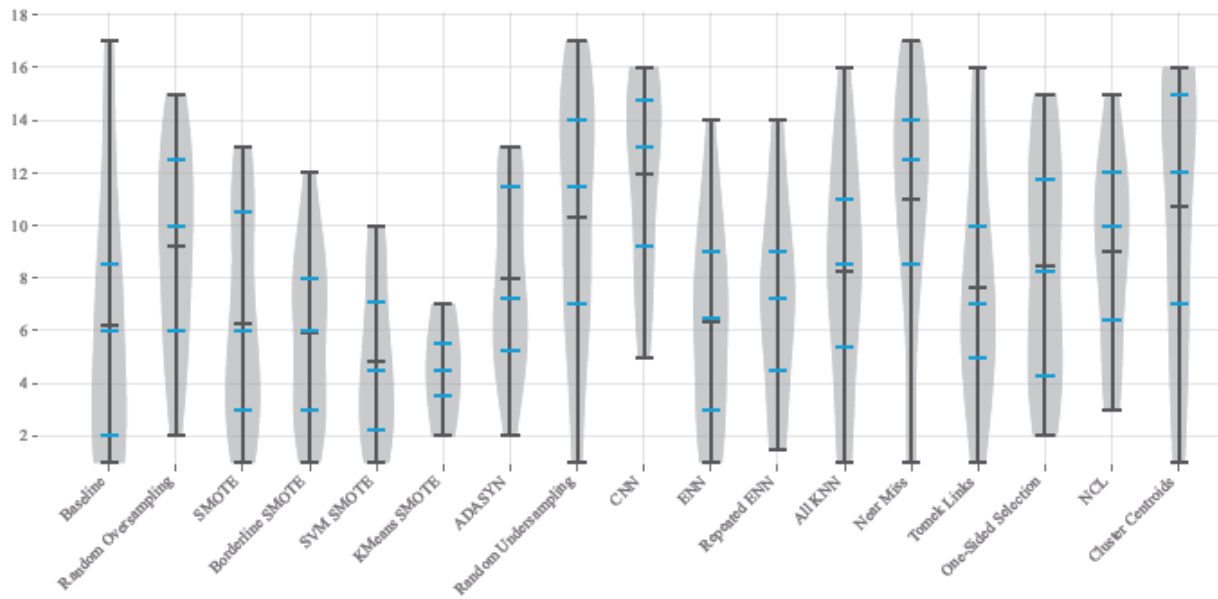


Fig. 2. Area under PR Curve (PR AUC). Ranks for each method were measured across all datasets in the benchmark.

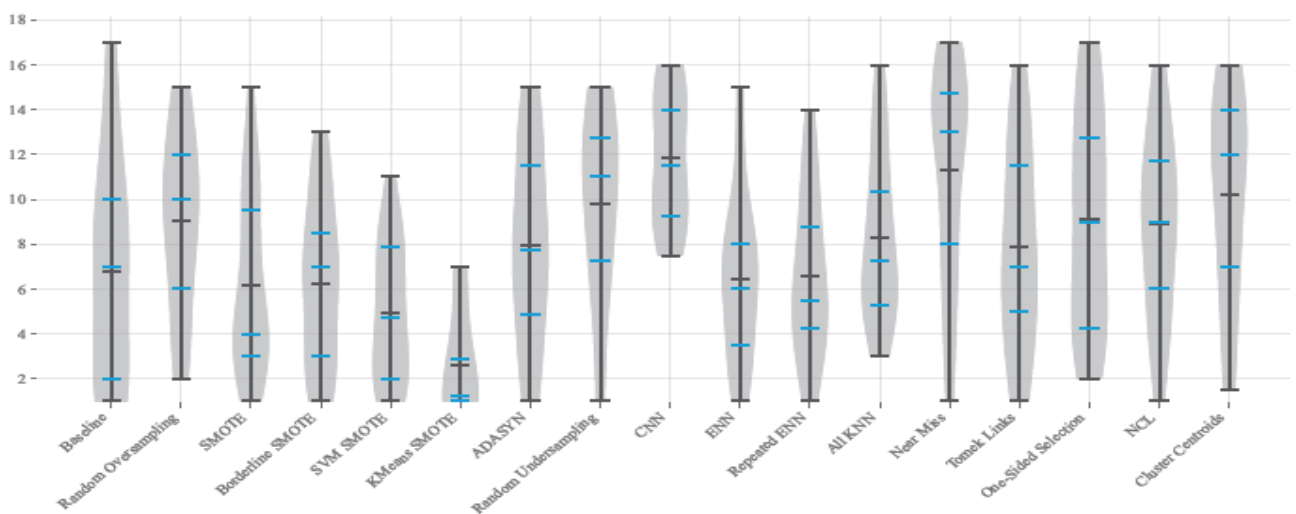


Fig. 3. Area under ROC Curve (ROC AUC). Ranks for each method were measured across all datasets in the benchmark.

The hyperparameter search is done not by using accuracy, as is typical, but by ROC AUC, which is a more appropriate metric for imbalanced problems. As a last point, our performance metrics take into account various operating points of the final model; PR AUC and P-ROC AUC specifically examine the relevant operating points. Occasionally, methods are compared using measures like F-score, which only take into account one operating point, and the operating point is selected at random without proper tuning. Adjusting the operating point can lead to oversampling or undersampling.

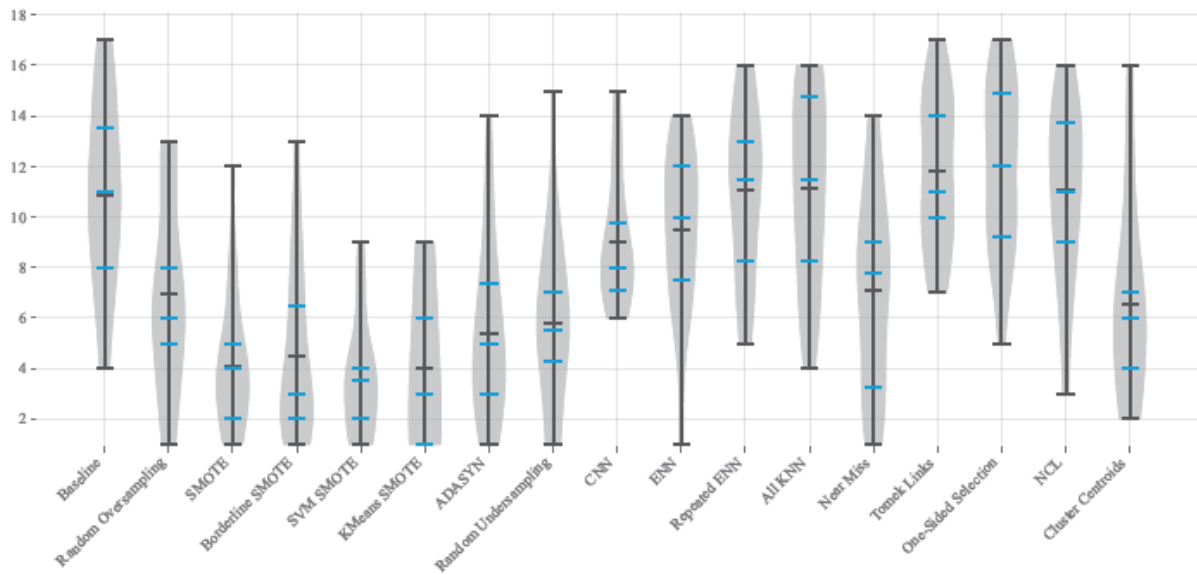


Fig. 4. Area under Partial ROC Curve (P-ROC AUC). Ranks for each method were measured across all datasets in the benchmark.

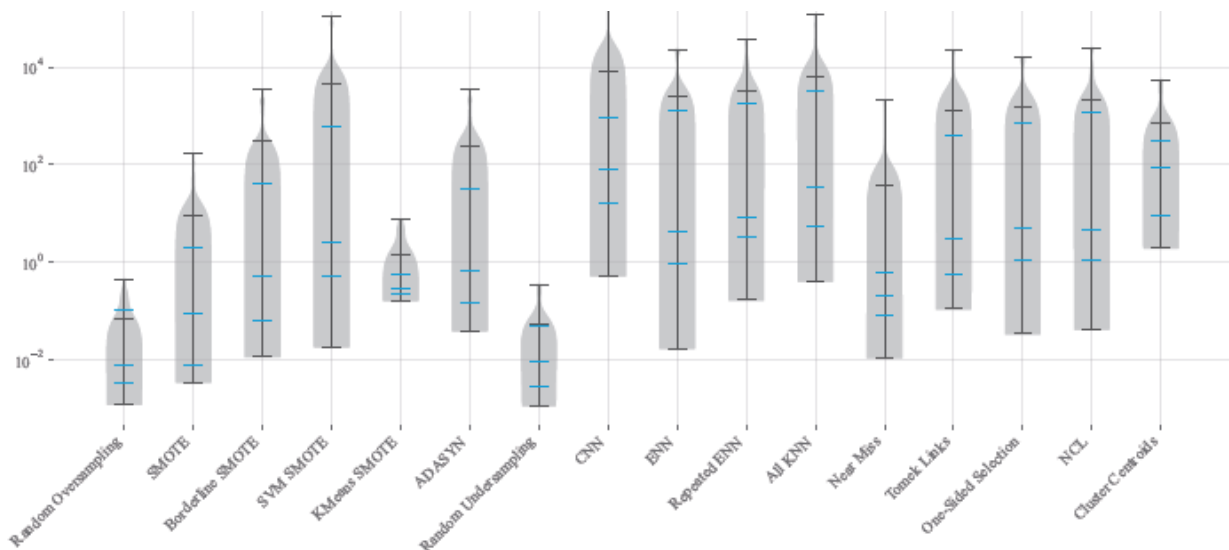


Fig. 5. Runtimes (in seconds) for each preprocessing method were measured across all datasets in the benchmark.

Overall, the best rank is achieved by dom Undersampling. It is clear from this that specialized undersampling approaches are not meant to be used for general-purpose tasks, but rather, they work best with particular datasets that possess the features that these methods aim to achieve. Concerning the oversampling approaches, one may make a number of interesting observations. When compared to the baseline, oversampling strategies often provide higher results. The P-ROC AUC metric is where they really shine compared to the baseline. Random Oversampling often underperforms baseline on other metrics, whereas more advanced oversampling techniques provide superior outcomes. There is a noticeable pattern that more complex oversampling algorithms perform better than basic Random Oversampling, in contrast to the undersampling approaches. The greatest noticeable improvement in performance is provided by SMOTE. Even if they're smaller, BorderlineSMOTE and SVM SMOTE might nonetheless outperform regular SMOTE. To determine whether SMOTE, BorderlineSMOTE, SVM SMOTE, and ADASYN vary significantly in rankings, we used the Friedman test [19]. At the generally accepted significance thresholds, the test findings ($p = 0.09$) do not show statistical significance.

VII. CONCLUSION

Six datasets are from the cybersecurity area, and we have performed an innovative research of sixteen preprocessing approaches on twenty-three datasets. We looked at computational and predictive performance. With that goal in mind, we conducted a large-scale experiment that makes use of AutoML to evaluate several classifiers and incorporates a hyperparameter search to eliminate any bias that may have been introduced by previous benchmarks. When dealing with class-imbalanced classification, our key result is that dataset preparation is typically useful. On the other hand, several of the approaches never manage to beat the control group's default of doing nothing. There are cases when undersampling approaches perform worse than oversampling methods, although this is not always the case. Traditional SMOTE algorithm oversampling produces the most performance increases, however more advanced versions of the algorithm are likely to provide only modest improvements. We got the same results when we limited our study to cybersecurity datasets that cover different cybersecurity topics. Last but not least, remember that the performance metric you choose affects the method ranking. When dealing with class imbalance, we include many performance metrics that are thorough and applicable to real-world classification situations. The key principles highlighted above remain constant, even if the rankings differ in detail between measures.

REFERENCES

- [1] Mostofa Ahsan, Rahul Gomes, and Anne Denton. Smote implementation on phishing data to enhance cybersecurity. In 2018 IEEE International Conference on Electro/Information Technology (EIT), pages 0531–0536. IEEE, 2018.
- [2] Bathini Sai Akash, Pavan Kumar Reddy Yannam, Bokkasam Venkata Sai Ruthvik, Lov Kumar, Lalita Bhanu Murthy, and Aneesh Krishna. Predicting cyber-attacks on iot networks using deep-learning and different variants of smote. In International Conference on Advanced Information Networking and Applications, pages 243–255. Springer, 2022.
- [3] Adnan Amin, Sajid Anwar, Awais Adnan, Muhammad Nawaz, Newton Howard, Junaid Qadir, Ahmad Hawalah, and Amir Hussain. Comparing oversampling techniques to handle the class imbalance problem: A customer churn prediction case study. *IEEE Access*, 4:7940–7957, 2016.
- [4] Hyrum S Anderson and Phil Roth. Ember: an open dataset for training static pe malware machine learning models. arXiv preprint arXiv:1804.04637, 2018.
- [5] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. Dos and don'ts of machine learning in computer security. In 31st USENIX Security Symposium (USENIX Security 22), pages 3971–3988, Boston, MA, August 2022. USENIX Association.
- [6] Rob Ashmore, Radu Calinescu, and Colin Paterson. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. 54(5), May 2021.
- [7] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):186–205, 2000.

- [8] Salahuddin Azad, Syeda Salma Naqvi, Fariza Sabrina, Shaleeza Sohail, and Sweta Thakur. Iot cybersecurity: On the use of machine learning approaches for unbalanced datasets. In 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), pages 1–6. IEEE, 2021.
- [9] Sikha Bagui and Kunqi Li. Resampling imbalanced data for network intrusion detection datasets. *Journal of Big Data*, 8(1):1–41, 2021.
- [10] Ricardo Barandela, Rosa M Valdovinos, J Salvador S´anchez, and Francesc J Ferri. The imbalanced training sample problem: Under or over sampling? In Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR), pages 806–814. Springer, 2004.
- [11] Jan Brabec, Tom´aš Kom´arek, Vojtěch Franc, and Luk´aš Machlica. On model evaluation under non-constant class imbalance. In *International Conference on Computational Science*, pages 74–87. Springer, 2020.
- [12] Jan Brabec and Lukas Machlica. Bad practices in evaluation methodology relevant to class-imbalanced problems. *arXiv preprint arXiv:1812.01388*, 2018.
- [13] Jan Brabec and Lukas Machlica. Decision-forest voting scheme for classification of rare classes in network intrusion detection. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3325–3330, Oct 2018.
- [14] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010.
- [15] Nitesh V Chawla. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*, pages 875–886. Springer, 2009.
- [16] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.
- [17] Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 159–166. IEEE, 2015.
- [18] Anusha Damodaran, Fabio Di Troia, Corrado Aaron Visaggio, Thomas H Austin, and Mark Stamp. A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13(1):1–12, 2017.
- [19] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.
- [20] Stepan Dvorak, Pavel Prochazka, and Lukas Bajer. Gnn-based malicious network entities identification in large-scale network data. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–4, 2022.
- [21] Ayman El Aassal, Shahryar Baki, Avisha Das, and Rakesh M. Verma. An in-depth benchmarking and evaluation of phishing detection research for security needs. *IEEE Access*, 8:22170–22192, 2020.
- [22] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: Hands-free automl via metalearning. *arXiv:2007.04074 [cs.LG]*, 2020.
- [23] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28 (2015)*, pages 2962–2970, 2015.
- [24] Peter W Frey and David J Slate. Letter recognition using holland-style adaptive classifiers. *Machine learning*, 6(2):161–182, 1991.
- [25] Markus Goldstein. Unsupervised anomaly detection benchmark, 2015.