computers

MDPI

Article

Determining Resampling Ratios Using BSMOTE and SVM-SMOTE for Identifying Rare Attacks in Imbalanced Cybersecurity Data

Sikha S. Bagui [1,*], Dustin Mink [1], Subhash C. Bagui [2] and Sakthivel Subramaniam [1]

[1] Department of Computer Science, University of West Florida, Pensacola, FL 32514, USA; dmink@uwf.edu (D.M.); covaissvel@gmail.com (S.S.)
[2] Department of Mathematics and Statistics, University of West Florida, Pensacola, FL 32514, USA; sbagui@uwf.edu
* Correspondence: bagui@uwf.edu

Abstract: Machine Learning is widely used in cybersecurity for detecting network intrusions. Though network attacks are increasing steadily, the percentage of such attacks to actual network traffic is significantly less. And here lies the problem in training Machine Learning models to enable them to detect and classify malicious attacks from routine traffic. The ratio of actual attacks to benign data is significantly high and as such forms highly imbalanced datasets. In this work, we address this issue using data resampling techniques. Though there are several oversampling and undersampling techniques available, how these oversampling and undersampling techniques are most effectively used is addressed in this paper. Two oversampling techniques, Borderline SMOTE and SVM-SMOTE, are used for oversampling minority data and random undersampling is used for undersampling majority data. Both the oversampling techniques use KNN after selecting a random minority sample point, hence the impact of varying KNN values on the performance of the oversampling technique is also analyzed. Random Forest is used for classification of the rare attacks. This work is done on a widely used cybersecurity dataset, UNSW-NB15, and the results show that 10% oversampling gives better results for both BMSOTE and SVM-SMOTE.

Keywords: imbalanced data; resampling; rare attacks; Network Intrusion Datasets; BSMOTE; SVM-SMOTE; KNN; Random oversampling; Random undersampling; Random Forest classifier

check for updates

Citation: Bagui, S.S.; Mink, D.; Bagui, S.C.; Subramaniam, S. Determining Resampling Ratios Using BSMOTE and SVM-SMOTE for Identifying Rare Attacks in Imbalanced Cybersecurity Data. *Computers* **2023**, *12*, 204. https://doi.org/10.3390/computers12100204

Academic Editor: Paolo Bellavista

Received: 14 July 2023
Revised: 19 September 2023
Accepted: 21 September 2023
Published: 11 October 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Cyberattacks have become a norm of the digital world. Ref. [1] states that there are only two types of companies in the world, the type that has been hacked and the type that doesn't yet know they have been hacked. Attacks, by the individuals or organizations, aim to either steal, block access to, or delete information. Cyber-attacks affect the ecosystem of industry and result in huge financial losses. In 2021, globally, data breaches costed USD 4 million, and in USA it was twice that amount [2]. Cyberattacks not only damage the reputation of an enterprise, but also lead to legal complications. Cybersecurity teams in every organization develop, maintain and enforce policies and systems to identify and prevent such attacks on the network. In the initial days, Intrusion Detection Systems (IDS) were comprised of either Signature Based Detection or Behavior Based Detection. The former relied on the signature of the attack to be similar to the known signatures while the later compared the profile of the attack and compared it with the normal behavior of the standard profiles [3]. But, though network attacks are increasing steadily, the ratio of attacks to regular network traffic is still significantly less, creating highly imbalanced datasets. And this leads to the problem of effectively training ML models to detect and classify malicious traffic, especially rare malicious (attacks) traffic. Hence, predicting rare attacks in imbalanced datasets has become a significant problem. In this work, we address this

issue by using data resampling techniques that oversample minority data and undersample majority data.

Though several oversampling and undersampling techniques are available [4], this paper uses two oversampling techniques, Borderline SMOTE (BSMOTE) and SVM-SMOTE, in varying percentages for oversampling minority data. Random Undersampling (RU) is used for undersampling majority data. The oversampling techniques use K-Nearest Neighbor (KNN) to identify neighbors after selecting a random minority sample point. The impact of different KNN values has also been studied on the performance of the oversampling technique. And finally, the Random Forest (RF) classifier, which has previously been successfully used for the classification of imbalanced data [5], is used for classification. This work is done on UNSW-NB15 [6], a well-researched cybersecurity dataset with many minority classes or rare attacks. This work looks at three of the rarest classes. Hence, the points that make this paper unique are:

- The paper attempts to determine the optimal oversampling ratio that is needed to classify an attack with high accuracy; oversampling percentages are varied from 10% to 100%, with undersampling kept at a constant 50%.
- The paper determines if the order of resampling, that is, oversampling before undersampling or undersampling before oversampling, has an impact on classification;
- The paper studies whether there is any difference between BSMOTE and SVM-SMOTE in this experimental setup. The paper examines the impact of various KNN values on oversampling techniques.

The rest of this paper is organized as follows. Section 2 presents the background; Section 3 offers work related to oversampling and undersampling; Section 4 details the data used in this paper; Section 5 contains the experimental design; Section 6 outlines the hardware and software configurations; Section 7 presents the metrics used to present the results; Section 8 presents the results and discussions; Section 9 collates the conclusions and presents the future works.

2. Background

2.1. Resampling

Resampling is primarily used in imbalanced datasets to modify the data distribution in the training dataset [7]. It is considered an effective way of obtaining a more balanced data distribution in imbalanced datasets [8]. But before starting resampling, it is essential to understand the need for resampling. ML classifiers get skewed towards classes that have more data. And, since the majority class(es) have more data, classifiers often give high accuracy, but this is because they are predicting only the majority class(es) and not really classifying the minority classes. Hence, to accurately identify network attacks, which are usually the minority classes, using an ML classifier, it is necessary to balance the classes, and therefore the need for resampling.

For a dataset to be considered highly imbalanced, it should contain most data from a few majority classes and very few data points from the minority classes. In the case of network attacks, benign network traffic forms the majority of the population, and malicious attacks form a minority segment; hence we term the minority segment as rare attacks. Such rare attacks are complicated to identify in the real world, and training a ML model is even harder.

2.1.1. Undersampling

Undersampling is a technique where the majority class instances are reduced to bring balance to the dataset. Depending on the resampling type, the majority class samples may be analyzed before removing them from the distribution. Brute force approaches to the resampling include Random Undersampling (RU), where the algorithm does not know whether the data points being removed (by RU) are critical.

2.1.2. Oversampling

Oversampling addresses the minority classes in the dataset. It either creates duplicates of the existing minority class samples, as in the case of Random Oversampling (RO) or generates new synthetic points in the feature space, as is the case of the SMOTE family of methods. Oversampling of the minority class is done to scale up to the count of the majority class. That is, 10% oversampling means the existing minority class count is increased to match ten percent of the majority class.

In this work, resampling methods were not applied to the whole dataset but were limited to the training dataset. For ML models, it is imperative to monitor the class distribution in the training set to observe the patterns of the real population. Stratified sampling was used to ensure that, after the split, each class was represented in the training as well as testing data.

2.2. K-Nearest Neighbor

K-Nearest Neighbor (KNN) works by finding the nearest instances surrounding the instance to be classified and predicting the output based on those K instances. It is a non-parametric machine learning algorithm that does not make any assumptions about the form of the mapping function [9]. To classify any given instance, it looks up its K neighbors and assigns the instance to the appropriate class. The nearest neighbors are selected based on a distance measure in the feature space. KNN is used by synthetic data generation techniques such as SMOTE by creating new synthetic samples on the lines connecting the existing minority sample to their k-nearest neighbors [10,11]. Since KNN plays a vital role in the identification of misclassified instances in the case of BSMOTE, it was decided to vary K and analyze the results.

2.3. BSMOTE and SVM-SMOTE

Borderline SMOTE (BSMOTE) is an improvement of the classic SMOTE, where instead of oversampling all the minority class instances, the ones nearest to the borderline are identified and synthetic samples are generated [10]. SVM-SMOTE differs from the BSMOTE in that first standard SVMs are trained on the training data set, and then the borderline area is approximated. In SVM-SMOTE also, new data generation happens on the line created by joining the minority class support vector with its nearest neighbors [11]. In SVM-SMOTE, however, KNN values are not used to generate the decision boundary. They are used to select the kth nearest neighbor to create the new instances. Because of the differing roles played by KNN in these two different oversampling techniques, they are chosen for analysis in this paper.

2.4. Random Forest

Random Forest (RF) is a highly used machine learning classifier that uses the decision of multiple decision trees to develop a final classification label [12]. In the RF algorithm, the decision tree is generated differently from a regular decision tree algorithm. Features are randomly selected when a decision tree node splits, and of the randomly selected features, the best feature is selected based on statistical measures like Information Gain and the Gini Index.

3. Related Works

In the present day, Machine Learning (ML) is widely used in cybersecurity to detect network intrusions [13–15]. Ref. [13] looked at the application of K-nearest neighbor (KNN) on artificial neural networks (ANNs) to develop an algorithm for IDSs. Ref. [14] looked at the concept of the one-class classifier on the problem of anomaly detection in communication networks. Several research works have been carried out to achieve better classification results using ML when the datasets are imbalanced. Many review papers have been identified on this topic [16–19].

Next, papers that present different techniques to handle ML classification in the imbalanced dataset are presented. Ref. [10] presented two new minority oversampling techniques, borderline-SMOTE1 and borderline-SMOTE2. In both these techniques, the minority data near the borderline were oversampled. Authors of [10] claim that their methods achieved better TP rates and F-value scores than regular SMOTE and other oversampling methods.

Using a hybrid approach by combining ADASYN oversampling and TomekLink, researchers improved the detection of network intrusions on the NSL-KDD dataset [20]. A heterogenous ensemble learning approach developed by authors in another research [21] improved the Worms attack type detection rate in the UNSW-NB15 dataset. In one work, the authors used a combination of preprocessing techniques such as data standardization, normalization, as well as selection of features, and class balancing to improve the efficiency of the Random Forest classifier [5].

Ref. [22] proposed Tomek-link (TLUSBoost). TLUSBoost finds outliers using the Tomek-link concept and eliminates the probable redundant instances, thereby conserving the dataset's characteristics. And AdaBoost is used for boosting. Their ensemble method had better experimental results than many other proposed methods.

Ref. [23] proposed a new algorithm that combined boosting with heuristic undersampling and distribution-based sampling (HUSDOS-Boost) to solve the extremely imbalanced and small minority data problem. This algorithm was tested on health care data and presented good results.

There are also a few works that worked on resampling ratios. Ref. [24]'s study showed that increasing the percentage of the minority class from 0.1% to 1.0% of the majority class with partial balancing of the majority class gives better performance than balancing the classes to a 50:50 ratio. Ref. [25] conducted a study on the impact of class distribution when the size of the training dataset is limited and found that naturally occurring distributions do not give better performance. This work also found that the optimal distribution of a dataset has minority class samples between 50% and 90%. Ref. [26] investigated the effectiveness of a class proportion threshold using different classifiers. Ref. [27] studied class imbalance in datasets using four different classifiers and six performance evaluation criteria to determine if oversampling of the minority class gave higher accuracy (96%) than undersampling of the majority class (77%). Oversampling the minority data rather than undersampling the majority data helped detect the minority classes [28]. In the same work, the authors also detected that resampling will not impact the detection of rare classes if the data is not highly imbalanced.

The uniqueness of our work in terms of resampling ratios is that we looked at resampling ratios from 0.1% to 1.0% at intervals of 0.1%. Keeping the undersampling constant at 50%, we looked at whether oversampling before undersampling is better or vice versa. Also, using B-SMOTE and SVM SMOTE, we looked at different K values.

4. The Data: UNSW-NB15

UNSW-NB15 [6], created in 2015, contains 2.5 million rows, of which 2.2 million rows are of regular or benign traffic, and the other 300,000 are attack traffic. The attack traffic has nine attack categories, of which the smallest categories are Worms, Shellcode, and Backdoors, making up 0.006%, 0.059%, and 0.091% of the total traffic, respectively. Hence, these categories can be considered rare attacks and are particularly interesting in this research. Figure 1 presents a distribution of the attack families in this dataset.

Worms are a classification of malware that is self-propagating within a single host and across networks [29]. Shellcode is a type of exploit payload that modifies a program's flow of execution to spawn a command interpreter [30]. Backdoor is a classification of malware that allows unauthorized access to a computer [29].

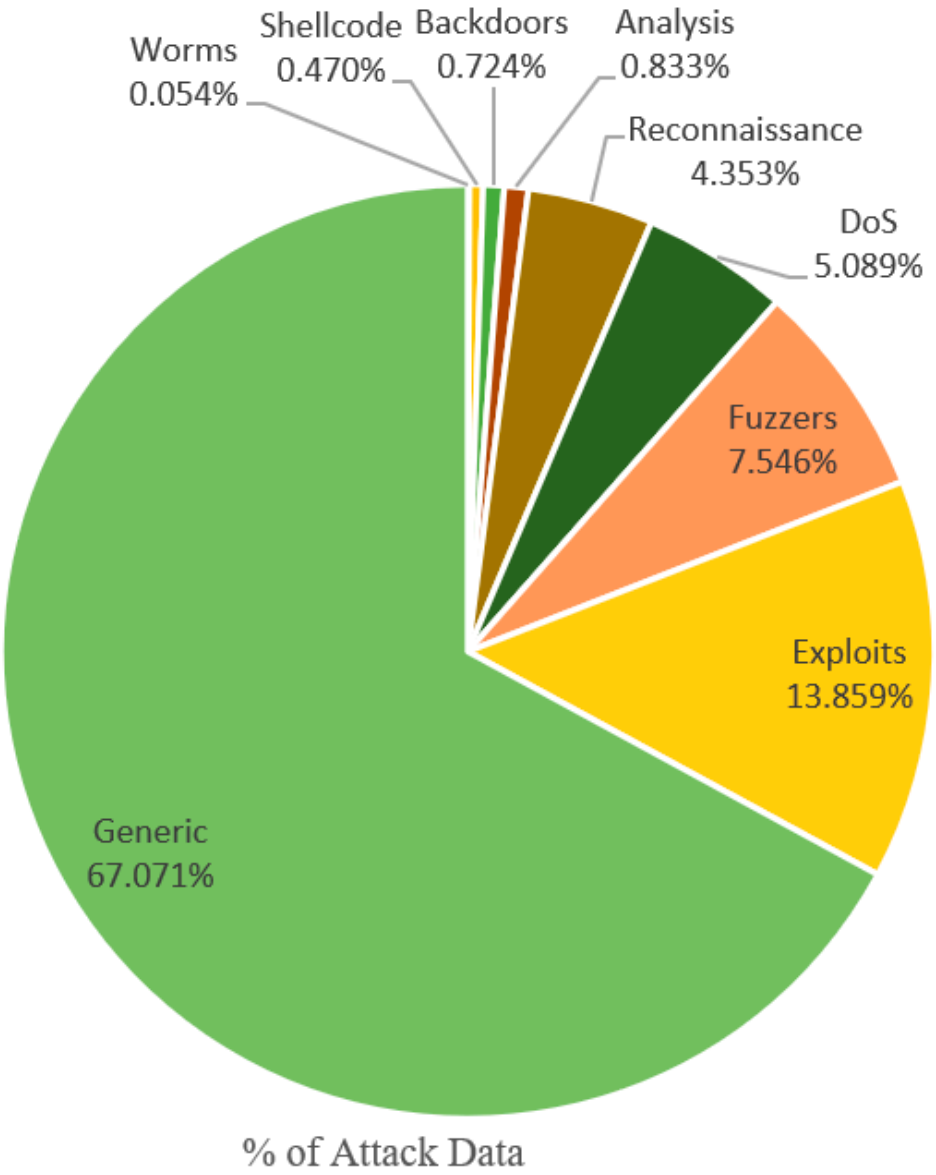


Figure 1. Distribution of Attack Families in UNSW-NB15.

5. Experimental Design

Two sets of experiments were performed to analyze the resampling effectiveness: (i) Oversampling minority data followed by undersampling, and (ii) Undersampling majority data followed by oversampling. Two oversampling techniques, BSMOTE and SVM-SMOTE, were used in both experiments, with RU held constant at 50%. A stratified split of the datasets into training and testing data was used for both sets of experiments. A stratified split is performed to ensure that the training and testing datasets follow the original distribution of the dataset and that minority class instances are present before the synthetic generation using resampling techniques. This experimental design is presented in Figure 2.

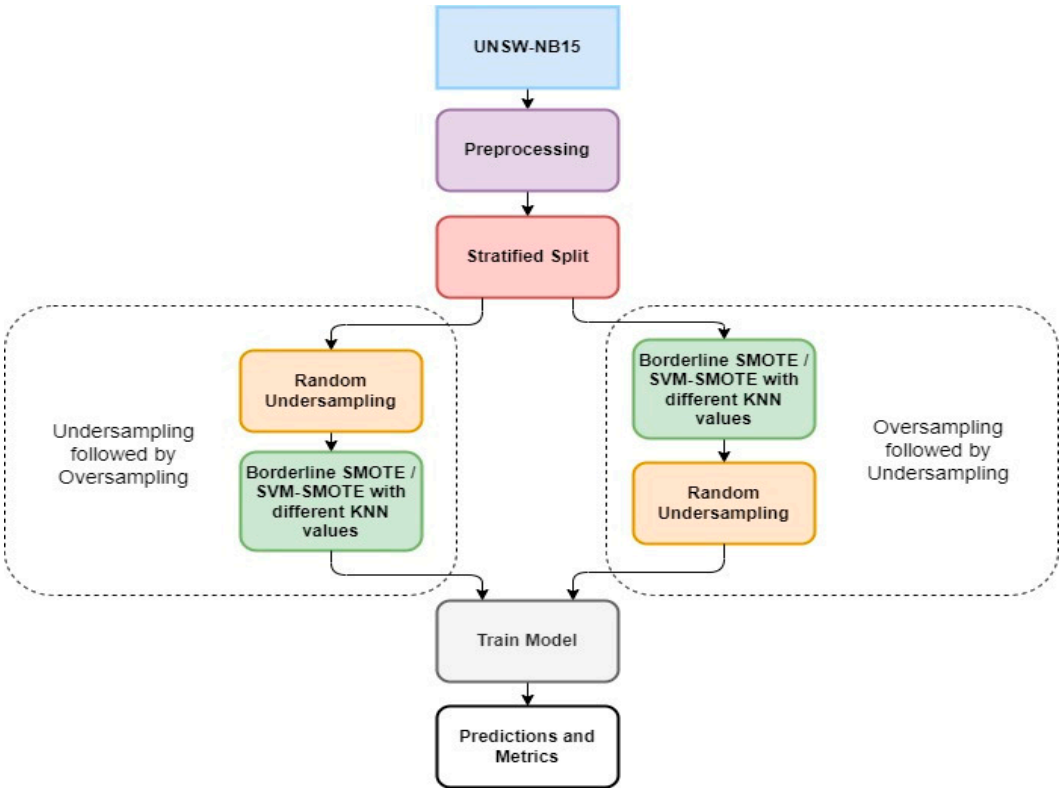


Figure 2. Experimental design.

Preprocessing

Preprocessing in this paper follows [31]. First, the time_stamps column was dropped since we were not doing any time related analysis. Categorical data, that is, protocol, state, and attack category, were converted into numeric values. Normalization was used on continuous data following [31]. Then, information gain was calculated, and features with low information gain were dropped. The dropped columns are shown with an asterisk (*) in Table 1. Information gain is calculated by removing the randomness in the dataset, which is measured by a class's entropy [32].

Table 1. Information Gain Analysis.

Feature	Information Gain	Columns Dropped
sttl	0.476	
dttl	0.422	
ct_state_ttl	0.354	
sbytes	0.345	
attack_cat	0.339	
state	0.318	
Sload	0.307	
smeansz	0.283	
proto	0.275	
dbytes	0.215	
dmeansz	0.204	
dur	0.193	
Dload	0.188	

Table 1. *Cont.*

Feature	Information Gain	Columns Dropped
Dintpkt	0.187	
Dpkts	0.178	
ct_dst_sport_lst	0.169	
swin	0.167	
dwin	0.165	
Ltime	0.139	
Stime	0.138	
Sintpkt	0.131	
tcprtt	0.127	
ackdat	0.126	
synack	0.125	
ct_src_dport_ltm	0.121	
ct_dst_src_ltm	0.108	
Spkts	0.107	
ct_dst_ltm	0.103	
Sjit	0.1	
Djit	0.097	
ct_src_ltm	0.097	*
ct_srv_dst	0.094	*
sloss	0.09	*
ct_srv_src	0.089	*
dloss	0.085	*
service	0.081	*
stcpd	0.056	*
dtcpb	0.054	*
res_bdy_len	0.016	*
trans_depth	0.009	*
is_sm_ips_ports	0.0004	*

6. Hardware and Software Configurations

The hardware and software configurations used for this analysis are listed in Table 2 and the Python libraries used in this paper are presented in Table 3.

Table 2. Hardware and Software Configurations.

Processor	M1 Max Pro
RAM	32 GB
OS	Mac OS Ventura
OS Version	13.1
OS Build	-
GPU	-

Table 3. Python Library Versions.

Python	3.9
Anaconda	2022.1
Pandas	1.5.2
Scikit-learn	1.9.3
Numpy	1.23.5
Imblearn	0.10.0

7. Metrics Used for Presentation of Results

7.1. Classification Metrics

Since accuracy gets biased towards classes with more data, which in this case would be the majority class or benign data, accuracy is not a good metric for classifying imbalanced data. To effectively evaluate the classification results for imbalanced data, the following matrices were used: Precision, Recall, F-score, and Macro Precision.

Precision: Precision is the proportion of Predicted Positive cases correctly labeled as Positive [33].

$$\text{Precision} = [\text{True Positives}]/[\text{True Positives} + \text{False Positives}] \quad (1)$$

Recall: This is the percentage of the correctly classified positive samples, also referred to as the True Positive Rate [33].

$$\text{All Real Positives} = [\text{True Positives} + \text{False Negatives}]$$

$$\text{Recall} = \text{True Positive Rate} = [\text{True Positives}]/[\text{All Real Positives}] \quad (2)$$

F-Score: The F-score is high when both the precision and recall are high [32].

$$\text{F-Score} = 2 \times [\text{Precision} \times \text{Recall}]/[\text{Precision} + \text{Recall}] \quad (3)$$

Macro Precision: This is the arithmetic mean of each individual class's precision.

7.2. Welch's t-Tests

Welch's *t*-tests were used to test the equality of two population means under unequal variances. Because of unequal variances, it's degrees of freedom (d.f.) is obtained using Satterthwaite approximation.

8. Results and Discussion

This section presents the results of the runs, that is, oversampling followed by undersampling and undersampling followed by oversampling using both BSMOTE and SVM-SMOTE, varying KNN. For all processing, undersampling was kept at 50% of the majority class, and oversampling was varied from 10% to 100%. In UNSW-NB15, non-attack(benign) data forms the majority class. All results are an average of ten runs, and the best results have been highlighted in green in the following tables.

8.1. Selection of KNN

Since the SMOTE algorithms use KNN before the selection of minority class instances for oversampling, the effect of varying KNN is studied. KNN = 3, 5, and 10 are used.

8.2. BSMOTE Oversampling Followed by Random Undersampling

In this set of runs, oversampling the minority data was followed by undersampling the majority data. Oversampling percentages were varied from 10% to 100% of the majority class count.

8.2.1. Worms: BSMOTE Oversampling Varying KNN followed by Random Undersampling

Table 4 presents the results for Worms for oversampling minority data using BSMOTE for KNN = 3, followed by RU of the majority data. Oversampling of 0.1 (10%) has the best overall results. This conclusion is arrived at by calculating the probabilistic value using Welch's *t*-test scores for the metrics, precision, recall, F-score and macro precision. The *p*-value significance was set to 1.0 to allow for more chances of rejecting the null hypothesis.

Table 4. Worms: BSMOTE Oversampling using KNN = 3 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.678	0.753	0.713	0.839
0.2	0.617	0.696	0.651	0.808
0.3	0.698	0.738	0.717	0.849
0.4	0.658	0.719	0.684	0.829
0.5	0.668	0.761	0.710	0.834
0.6	0.673	0.780	0.718	0.836
0.7	0.653	0.730	0.683	0.826
0.8	0.661	0.703	0.680	0.830
0.9	0.617	0.730	0.667	0.808
1.0	0.685	0.746	0.709	0.842

From Table 5, Welch's *t*-test values are first calculated between metrics of 0.1 and 0.2 oversampling ratios, followed by the *p*-value calculation. The idea here is to observe whether the metrics increase or decrease as the oversampling percentages vary. So, a one-tailed *t*-test is performed. Probability values for all four metrics are analyzed, and if any of them fall below the significance value, it is marked appropriately. There are three possible outcomes when doing this analysis based on three scenarios, as shown in Table 6.

Table 5. Welch's *t*-test: UNSW-NB15 Worms: BSMOTE oversampling followed by RU for KNN = 3.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	1.280	0.977	1.262	1.280	No statistical difference between 0.1 and 0.2
0.1 vs. 0.3	−0.470	0.388	−0.100	−0.470	No statistical difference between 0.1 and 0.3
0.1 vs. 0.4	0.537	0.716	0.760	0.537	No statistical difference between 0.1 and 0.4
0.1 vs. 0.5	0.221	−0.166	0.071	0.221	No statistical difference between 0.1 and 0.5
0.1 vs. 0.6	0.097	−0.604	−0.134	0.097	No statistical difference between 0.1 and 0.6
0.1 vs. 0.7	0.430	0.559	0.730	0.430	No statistical difference between 0.1 and 0.7
0.1 vs. 0.8	0.390	1.370	0.934	0.390	No statistical difference between 0.1 and 0.8
0.1 vs. 0.9	2.075	0.550	1.533	2.075	0.1 has better precision and macro precision than 0.9
0.1 vs. 1	−0.210	0.127	0.093	−0.210	No statistical difference between 0.1 and 1.0

For example, in Table 5, between 0.1 and 0.2, since no statistical differences are observed, 0.1 is preferred. Next, 0.1 is compared with 0.3, and so on. On the other hand, if we consider 0.1 vs. 0.9, all the *t*-test values are positive and significant differences are observed in precision and macro-precision. So, 0.1 oversampling is considered better than 0.9. For efficiency in execution time, having a lower oversampling percentage will always be better.

Table 6. Welch's *t*-Test Result Scenario vs. Outcome.

Scenario	Outcome
No statistical difference between the two oversampling percentages	Less oversampling percentage is preferred over the higher one due to less computational effort.
p-value indicates a significant difference in any of the metrics and the corresponding *t*-test score is positive.	First oversampling percentage is preferred as it gave better result than the second.
p-value indicates a significant difference in any of the metrics but the corresponding *t*-test score is negative.	Second oversampling percentage is preferred as it gave better result than the first.

Table 7 presents the results for Worms for oversampling using BSMOTE for KNN = 5, followed by RU, and Table 8 presents the results for KNN = 10. For both KNN = 5 and KNN = 10, 0.1 (10%) BSMOTE oversampling has the best results across all metrics. The Welch's *t*-test analysis results for these values are presented in Table 9 and Table 10, respectively.

Table 7. Worms: BSMOTE Oversampling using KNN = 5 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.608	0.736	0.664	0.804
0.2	0.600	0.711	0.646	0.800
0.3	0.566	0.773	0.651	0.783
0.4	0.565	0.780	0.653	0.782
0.5	0.581	0.738	0.649	0.790
0.6	0.586	0.759	0.656	0.793
0.7	0.619	0.753	0.678	0.809
0.8	0.539	0.719	0.614	0.769
0.9	0.573	0.711	0.628	0.786
1.0	0.600	0.750	0.665	0.800

Table 8. Worms: BSMOTE Oversampling using KNN = 10 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.511	0.750	0.606	0.755
0.2	0.504	0.757	0.604	0.752
0.3	0.480	0.742	0.580	0.740
0.4	0.494	0.746	0.593	0.747
0.5	0.490	0.757	0.593	0.745
0.6	0.471	0.703	0.563	0.735
0.7	0.492	0.723	0.582	0.746
0.8	0.480	0.734	0.578	0.740
0.9	0.482	0.726	0.578	0.741
1.0	0.495	0.734	0.589	0.747

Table 9. Welch's *t*-test: Worms: BSMOTE oversampling using KNN = 5 followed by RU.

Welch's *t*-Test Results ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.183	0.799	0.612	0.183	0.1 and 0.2 are statistically equal
0.1 vs. 0.3	1.413	−1.563	0.592	1.413	0.1 is better than 0.3 except F-Score.
0.1 vs. 0.4	1.773	−1.826	0.628	1.773	0.1 is better than 0.4 except F-Score
0.1 vs. 0.5	0.836	−0.065	0.506	0.836	0.1 and 0.5 are statistically equal

Table 9. *Cont.*

Welch's *t*-Test Results ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.6	0.451	−0.957	0.264	0.451	0.1 and 0.6 are statistically equal
0.1 vs. 0.7	−0.431	−0.859	−0.706	−0.431	0.1 and 0.7 are statistically equal
0.1 vs. 0.8	2.052	0.959	2.500	2.826	0.1 better than 0.8 except recall where both of them are statistically equal
0.1 vs. 0.9	0.821	0.745	1.131	0.821	0.1 and 0.9 are statistically equal
0.1 vs. 1	0.219	−0.746	−0.033	0.302	0.1 and 1 are statistically equal

Table 10. Welch's *t*-test: Worms: BSMOTE oversampling using KNN = 10 followed by RU.

Welch's *t*-Test Results ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.194	−0.264	0.077	0.194	No statistical difference between 0.1 and 0.2
0.1 vs. 0.3	1.282	0.230	1.400	1.282	No statistical difference between 0.1 and 0.3
0.1 vs. 0.4	0.696	0.108	0.566	0.696	No statistical difference between 0.1 and 0.4
0.1 vs. 0.5	1.130	−0.207	0.932	1.130	No statistical difference between 0.1 and 0.5
0.1 vs. 0.6	1.362	0.963	1.399	1.362	No statistical difference between 0.1 and 0.6
0.1 vs. 0.7	0.677	0.695	0.918	0.677	No statistical difference between 0.1 and 0.7
0.1 vs. 0.8	1.266	0.434	1.589	1.266	0.1 has better F-Score than 0.8
0.1 vs. 0.9	0.889	0.537	0.880	0.889	No statistical difference between 0.1 and 0.9
0.1 vs. 1	0.577	0.251	0.464	0.577	No statistical difference between 0.1 and 1.0

8.2.2. Shellcode: BSMOTE Oversampling Varying KNN Followed by Random Undersampling

Shellcode has the second least number of occurrences (1511 out of 321,283 attacks) in the UNSW-NB15 dataset. Tables 11–13 present the BSMOTE oversampling results followed by RU for Shellcode for KNN = 3, 5, and 10, respectively. When Welch's *t*-test scores were calculated to determine the best oversampling percentage, all three KNN values had 0.1 as the best oversampling percentage for better minority sample prediction.

Table 11. Shellcode: BSMOTE Oversampling using KNN = 3 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.719	0.906	0.802	0.859
0.2	0.719	0.913	0.805	0.859
0.3	0.713	0.908	0.799	0.856
0.4	0.700	0.903	0.789	0.850
0.5	0.720	0.905	0.802	0.859
0.6	0.713	0.899	0.795	0.856
0.7	0.705	0.898	0.790	0.852
0.8	0.703	0.895	0.788	0.851
0.9	0.706	0.910	0.795	0.853
1.0	0.692	0.905	0.784	0.846

Table 12. Shellcode: BSMOTE oversampling using KNN = 5 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.694	0.900	0.783	0.847
0.2	0.678	0.917	0.780	0.839
0.3	0.682	0.916	0.782	0.841
0.4	0.696	0.903	0.786	0.847
0.5	0.694	0.916	0.790	0.847
0.6	0.679	0.912	0.778	0.839
0.7	0.682	0.911	0.779	0.841
0.8	0.698	0.917	0.792	0.849
0.9	0.679	0.917	0.780	0.839
1.0	0.700	0.904	0.789	0.850

Table 13. Shellcode: BSMOTE oversampling using KNN = 10 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.660	0.929	0.772	0.830
0.2	0.657	0.929	0.770	0.828
0.3	0.653	0.925	0.765	0.826
0.4	0.666	0.913	0.770	0.832
0.5	0.658	0.917	0.766	0.829
0.6	0.652	0.921	0.764	0.826
0.7	0.643	0.924	0.758	0.821
0.8	0.649	0.920	0.761	0.824
0.9	0.656	0.923	0.767	0.828
1.0	0.660	0.907	0.764	0.830

Welch's *t*-test results for runs of BSMOTE oversampling with KNN = 3 for Shellcode are presented in Table 14. It can be noted that 0.1 had no statistical difference with 0.2, 0.3, 0.5 and 0.6 oversampling percentages. It also produced better results than 0.8, 0.9 and 1.0 for precision and macro-precision.

Table 14. Welch's *t*-test: Shellcode: BSMOTE oversampling using KNN = 3 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	−0.058	−0.958	−0.434	−0.058	No statistical difference between 0.1 and 0.2
0.1 vs. 0.3	1.011	−0.140	0.476	1.011	No statistical difference between 0.1 and 0.3
0.1 vs. 0.4	4.204	0.360	3.220	4.205	0.1 has better precision, F-Score and macro than 0.4
0.1 vs. 0.5	−0.091	0.095	0.000	−0.091	No statistical difference between 0.1 and 0.5
0.1 vs. 0.6	0.849	0.726	1.213	0.849	No statistical difference between 0.1 and 0.6
0.1 vs. 0.7	1.540	0.839	1.468	1.541	0.1 has better precision and macro precision than 0.7
0.1 vs. 0.8	2.186	1.217	2.743	2.188	0.1 has better precision, F-Score and macro precision than 0.8
0.1 vs. 0.9	2.636	−0.341	1.262	2.635	0.1 has better precision and macro-precision than 0.9
0.1 vs. 1	2.880	0.187	2.869	2.881	0.1 has better precision, F-Score and macro precision than 1

Welch's *t*-test analysis are presented in Tables 15 and 16 for BSMOTE oversampling of Shellcode data followed by RU using KNN = 5 and 10 respectively. In both of these sets of runs, 0.1 oversampling gave better results than higher oversampling percentages.

Table 15. Welch's *t*-test: Shellcode: BSMOTE oversampling using KNN = 5 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.566	−0.276	0.386	0.566	0.1 and 0.2 are statistically equal.
0.1 vs. 0.3	0.874	−0.706	0.442	0.874	0.1 and 0.3 are statistically equal.
0.1 vs. 0.4	1.805	0.070	1.702	1.806	0.1 better than 0.4 but the latter has better recall
0.1 vs. 0.5	3.349	−0.128	3.104	3.349	0.1 better than 0.5. but the latter has better recall.
0.1 vs. 0.6	−0.084	−0.489	−0.284	−0.084	0.1 and 0.6 are statistically equal
0.1 vs. 0.7	0.684	−0.730	0.293	0.683	0.1 and 0.7 are statistically equal
0.1 vs. 0.8	1.620	0.520	1.480	1.620	0.1 is better than 0.8 across precision, F-Score and macro precision.
0.1 vs. 0.9	1.202	2.814	2.187	1.204	0.1 is better than 0.9 across recall and F-Score metrics.
0.1 vs. 1	2.101	0.246	1.832	2.101	0.1 is better than 1

Table 16. Welch's *t*-test: Shellcode: BSMOTE oversampling using KNN = 10 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.423	0.109	0.550	0.423	No statistical difference between 0.1 and 0.2
0.1 vs. 0.3	1.122	0.627	1.278	1.123	No statistical difference between 0.1 and 0.3
0.1 vs. 0.4	−0.891	2.648	0.401	−0.890	0.1 has better recall than 0.4
0.1 vs. 0.5	0.205	1.923	0.817	0.206	0.1 has better recall than 0.5
0.1 vs. 0.6	1.091	1.326	1.317	1.092	0.1 and 0.6 are statistically equal
0.1 vs. 0.7	1.781	1.876	2.123	1.781	0.1 is better than 0.7 across all metrics
0.1 vs. 0.8	1.783	2.637	2.353	1.784	0.1 is better than 0.8 across all metrics
0.1 vs. 0.9	0.336	0.887	0.761	0.337	0.1 and 0.9 are statistically equal
0.1 vs. 1	−0.040	2.210	2.289	−0.038	0.1 has better recall and F-Score than 1.0

8.2.3. Backdoors: BSMOTE Oversampling Varying KNN Followed by Random Undersampling

Backdoors had more occurrences than Shellcode (2329 out of 321,283 attacks), but it is still a minority class in the distribution. Tables 17–19 present the results of varying oversampling percentages using BSMOTE followed by RU for KNN = 3, 5 and 10, respectively.

Table 17. Backdoors: BSMOTE oversampling using KNN = 3 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.958	0.939	0.948	0.978
0.2	0.953	0.942	0.947	0.976
0.3	0.938	0.950	0.944	0.969
0.4	0.938	0.950	0.944	0.969
0.5	0.947	0.947	0.947	0.973
0.6	0.943	0.949	0.946	0.971
0.7	0.948	0.945	0.947	0.974
0.8	0.949	0.942	0.946	0.974
0.9	0.945	0.941	0.943	0.972
1.0	0.952	0.945	0.948	0.976

Table 18. Backdoors: BSMOTE oversampling using KNN = 5 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.966	0.952	0.959	0.983
0.2	0.966	0.943	0.954	0.983
0.3	0.965	0.940	0.953	0.982
0.4	0.965	0.938	0.951	0.982
0.5	0.961	0.944	0.952	0.980
0.6	0.964	0.944	0.954	0.982
0.7	0.963	0.943	0.953	0.981
0.8	0.954	0.941	0.947	0.976
0.9	0.964	0.939	0.951	0.982
1.0	0.974	0.935	0.954	0.987

Table 19. Backdoors: BSMOTE oversampling using KNN = 10 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.968	0.944	0.956	0.984
0.2	0.965	0.937	0.950	0.982
0.3	0.955	0.945	0.950	0.977
0.4	0.964	0.953	0.958	0.982
0.5	0.964	0.949	0.956	0.982
0.6	0.955	0.944	0.949	0.977
0.7	0.949	0.948	0.948	0.974
0.8	0.959	0.943	0.951	0.979
0.9	0.957	0.944	0.950	0.978
1.0	0.957	0.947	0.952	0.978

Welch's *t*-test calculations result in 0.1 (10%) BSMOTE oversampling having better overall results when compared to other oversampling percentages. For KNN = 3, 5 and 10 in backdoors, 10% BSMOTE oversampling was sufficient for the ML model to predict the minority class samples in the testing dataset as presented in the Table 20, Table 21, and Table 22, respectively.

Table 20. Welch's *t*-test: Backdoors: BSMOTE oversampling using KNN = 3 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.885	−0.503	0.369	0.886	No statistical diff between 0.1 and 0.2
0.1 vs. 0.3	3.240	−2.923	1.179	3.238	0.1 has Precision and F-Score while 0.2 has better recall
0.1 vs. 0.4	3.013	−2.102	1.730	3.013	0.1 has Precision and F-Score while 0.4 has better recall
0.1 vs. 0.5	2.140	−1.339	0.485	2.140	0.1 has better precision and macro precision
0.1 vs. 0.6	3.929	−2.392	0.931	3.928	0.1 has better precision and macro precision while 0.6 has better recall
0.1 vs. 0.7	1.612	−1.610	0.301	1.611	0.1 has better precision and macro precision while 0.7 has better recall
0.1 vs. 0.8	2.281	−0.830	1.055	2.281	0.1 has better precision and macro precision than 0.8
0.1 vs. 0.9	1.833	−0.339	1.001	1.832	0.1 has better precision and macro precision than 0.8
0.1 vs. 1	1.208	−1.272	−0.052	1.207	No statistical diff between 0.1 and 1.0

Table 21. Welch's *t*-test: Backdoors: BSMOTE oversampling followed by RU for KNN = 5.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	−0.077	1.726	1.294	−0.075	0.1 has better recall than 0.2
0.1 vs. 0.3	0.101	1.787	1.735	0.104	0.1 has better recall and F-Score than 0.3
0.1 vs. 0.4	0.162	1.869	2.591	0.164	0.1 has better recall and F-Score than 0.4
0.1 vs. 0.5	0.904	1.401	2.207	0.906	0.1 has better F-Score than 0.5
0.1 vs. 0.6	0.331	1.732	1.611	0.333	0.1 has better recall and F-Score than 0.6
0.1 vs. 0.7	0.492	1.771	2.001	0.494	0.1 has better recall and F-Score than 0.7
0.1 vs. 0.8	1.747	2.140	2.807	1.749	0.1 is better than 0.8 across all metrics
0.1 vs. 0.9	0.303	2.023	1.681	0.306	0.1 has better recall and F-Score than 0.9
0.1 vs. 1	−1.319	2.715	1.436	−1.316	0.1 has better recall than 1.0

Table 22. Welch's *t*-test: Backdoors: BSMOTE oversampling using KNN = 10 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.785	1.484	1.507	0.787	No statistical diff between 0.1 and 0.2
0.1 vs. 0.3	4.463	−0.113	1.854	4.460	0.1 has better precision, F-Score and macro precision than 0.3
0.1 vs. 0.4	1.441	−1.525	−1.064	1.440	No statistical diff between 0.1 and 0.4
0.1 vs. 0.5	1.532	−0.902	−0.107	1.529	No statistical diff between 0.1 and 0.5
0.1 vs. 0.6	2.751	0.043	1.724	2.751	0.1 has better precision, F-Score and macro precision than 0.6
0.1 vs. 0.7	3.699	−1.024	2.715	3.699	0.1 has better precision, F-Score and macro precision than 0.7
0.1 vs. 0.8	2.271	0.066	1.327	2.274	0.1 has better precision and macro precision than 0.8
0.1 vs. 0.9	3.983	−0.039	1.464	3.985	0.1 has better precision and macro precision than 0.9
0.1 vs. 1	4.994	−0.632	1.922	4.996	0.1 has better precision, F-Score and macro precision than 1.0

8.3. SVM-SMOTE Oversampling Followed by Random Undersampling

SVM-SMOTE is an extension of the SMOTE technique where support vectors are used to identify the decision boundary between the majority and minority classes for classification. The minority samples close to the decision boundary are selected for resampling. As was done with BSMOTE oversampling, KNN was varied for 3, 5, and 10 for SVM-SMOTE.

8.3.1. Worms: SVM-SMOTE Oversampling Varying KNN Followed by Random Undersampling

Results for SVM-SMOTE oversampling using KNN 3, 5 and 10, on Worms, are presented in Table 23, Table 24 and Table 25 respectively. Probabilistic significance evaluation of the Welch's *t*-test values resulted in 10% oversampling performing better for KNN = 3 and 5 as shown in Tables 26 and 27.

When compared to KNN = 3 and 5, for KNN = 10, 30% oversampling performed the best. Though higher oversampling percentages (>30%) had higher precision and F-score values like 100% oversampling, the *p*-value calculations did not find the differences to be significant. The results are captured in Table 28.

Table 23. Worms: SVM-SMOTE oversampling using KNN = 3 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.674	0.776	0.721	0.837
0.2	0.673	0.753	0.710	0.836
0.3	0.649	0.734	0.686	0.824
0.4	0.649	0.761	0.698	0.824
0.5	0.602	0.723	0.656	0.801
0.6	0.708	0.815	0.752	0.854
0.7	0.652	0.734	0.689	0.826
0.8	0.682	0.796	0.733	0.841
0.9	0.618	0.800	0.696	0.809
1.0	0.614	0.703	0.655	0.807

Table 24. Worms: SVM-SMOTE oversampling using KNN = 5 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.605	0.692	0.643	0.802
0.2	0.562	0.773	0.650	0.781
0.3	0.605	0.788	0.683	0.802
0.4	0.534	0.726	0.612	0.767
0.5	0.572	0.738	0.642	0.786
0.6	0.552	0.776	0.644	0.776
0.7	0.591	0.769	0.668	0.795
0.8	0.512	0.780	0.617	0.756
0.9	0.578	0.780	0.660	0.789
1.0	0.492	0.753	0.594	0.746

Table 25. Worms: SVM-SMOTE oversampling using KNN = 10 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.444	0.765	0.562	0.722
0.2	0.420	0.750	0.538	0.710
0.3	0.492	0.780	0.600	0.746
0.4	0.479	0.819	0.603	0.739
0.5	0.461	0.769	0.572	0.730
0.6	0.453	0.780	0.573	0.726
0.7	0.454	0.773	0.571	0.727
0.8	0.465	0.792	0.585	0.732
0.9	0.481	0.765	0.590	0.740
1.0	0.495	0.769	0.602	0.747

Table 26. Welch's *t*-test: Worms: SVM-SMOTE oversampling using KNN = 3 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.048	0.493	0.313	0.048	No statistical difference between 0.1 and 0.2
0.1 vs. 0.3	0.863	0.898	1.127	0.863	0.1 has better F-Score and macro then 0.3
0.1 vs. 0.4	0.873	0.350	0.754	0.873	No statistical difference between 0.1 and 0.4
0.1 vs. 0.5	3.307	1.744	2.919	3.307	0.1 is significantly better than 0.5
0.1 vs. 0.6	−0.867	−1.075	−1.440	−0.867	No statistical difference between 0.1 and 0.6

Table 26. *Cont.*

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.7	1.320	1.333	1.793	1.320	0.1 has better F-Score than 0.7
0.1 vs. 0.8	−0.189	−0.514	−0.335	−0.189	No statistical difference between 0.1 and 0.8
0.1 vs. 0.9	3.252	−0.772	1.497	3.252	0.1 has better precision and macro
0.1 vs. 1	2.810	2.264	3.058	2.810	0.1 is better than 1.0 across all metrics

Table 27. Welch's *t*-test: Worms: SVM-SMOTE oversampling using KNN = 5 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	1.766	0.654	1.436	1.765	No significant difference
0.1 vs. 0.3	−2.615	−0.426	−1.909	−2.615	0.1 is better than 0.3 but the latter has better f-score.
0.1 vs. 0.4	−1.778	−1.887	−1.974	−1.778	0.1 is better than 0.4 but 0.4 has better f-score
0.1 vs. 0.5	−0.982	−0.095	−0.541	−0.982	0.1 and 0.5 are statistically equal
0.1 vs. 0.6	−0.439	−0.409	−0.444	−0.439	0.1 and 0.6 are statistically equal
0.1 vs. 0.7	−0.566	−0.309	−0.495	−0.566	0.1 and 0.7 are statistically equal
0.1 vs. 0.8	−1.206	−0.905	−1.159	−1.206	0.1 better than 0.8 except recall where both of them are statistically equal
0.1 vs. 0.9	−1.726	0.000	−1.216	−1.726	0.1 and 0.9 are statiscally equal
0.1 vs. 1	−2.922	−0.129	−1.976	−2.922	0.1 and 1 are statiscally equal

Table 28. Welch's *t*-test: Worms: SVM-SMOTE oversampling for KNN = 10 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	1.248	0.462	1.015	1.248	No statistical diff bet 0.1 and 0.2
0.1 vs. 0.3	−1.849	−0.301	−1.350	−1.849	0.3 has better precision and macro precision than 0.1
0.3 vs. 0.4	0.471	−0.820	−0.101	0.471	No statistical diff bet 0.3 and 0.4
0.3 vs. 0.5	1.230	0.187	1.106	1.230	No statistical diff bet 0.3 and 0.5
0.3 vs. 0.6	1.483	0.000	1.031	1.483	No statistical diff bet 0.3 and 0.5
0.3 vs. 0.7	1.401	0.181	1.153	1.402	No statistical diff bet 0.3 and 0.7
0.3 vs. 0.8	1.045	−0.238	0.566	1.045	No statistical diff bet 0.3 and 0.8
0.3 vs. 0.9	0.353	0.333	0.320	0.353	No statistical diff bet 0.3 and 0.9
0.3 vs. 1.0	−0.099	0.254	−0.068	−0.099	No statistical diff bet 0.3 and 1.0

8.3.2. Shellcode: SVM-SMOTE Oversampling Varying KNN Followed by Random Undersampling

Tables 29–31 present the results of SVM-SMOTE oversampling of the Shellcode attack. For KNN = 3, 5 and 10, *p*-value calculations of Welch's *t*-test scores reveal that there were no significant differences observed between 10% and most of the higher percentages. The analysis results are captured in Table 32, Table 33 and Table 34 respectively.

Table 29. Shellcode: SVM-SMOTE oversampling using KNN = 3 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.709	0.911	0.797	0.854
0.2	0.719	0.908	0.802	0.859
0.3	0.703	0.910	0.793	0.851
0.4	0.709	0.904	0.794	0.854
0.5	0.711	0.904	0.795	0.855
0.6	0.717	0.909	0.801	0.858
0.7	0.708	0.902	0.793	0.854
0.8	0.708	0.900	0.792	0.854
0.9	0.713	0.910	0.799	0.856
1.0	0.710	0.904	0.795	0.855

Table 30. Shellcode: SVM-SMOTE oversampling using KNN = 5 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.699	0.906	0.789	0.849
0.2	0.696	0.913	0.789	0.848
0.3	0.690	0.918	0.788	0.845
0.4	0.693	0.913	0.787	0.846
0.5	0.687	0.905	0.781	0.843
0.6	0.705	0.904	0.792	0.852
0.7	0.684	0.909	0.781	0.842
0.8	0.685	0.912	0.783	0.842
0.9	0.710	0.910	0.798	0.855
1.0	0.692	0.913	0.788	0.846

Table 31. Shellcode-SVM-SMOTE oversampling using KNN = 10 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.688	0.931	0.791	0.844
0.2	0.657	0.928	0.769	0.828
0.3	0.683	0.924	0.786	0.841
0.4	0.672	0.912	0.773	0.836
0.5	0.661	0.914	0.767	0.830
0.6	0.661	0.912	0.766	0.830
0.7	0.645	0.921	0.758	0.822
0.8	0.673	0.913	0.775	0.836
0.9	0.665	0.920	0.772	0.832
1.0	0.653	0.922	0.764	0.826

Table 32. Welch's *t*-test: Shellcode: SVM-SMOTE oversampling using KNN = 3 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	−0.536	0.391	−0.448	−0.536	No statistical diff between 0.1 and 0.2
0.1 vs. 0.3	0.360	0.177	0.367	0.360	No statistical diff between 0.1 and 0.3
0.1 vs. 0.4	−0.003	0.735	0.255	−0.003	No statistical diff between 0.1 and 0.4
0.1 vs. 0.5	−0.104	0.683	0.150	−0.104	No statistical diff between 0.1 and 0.5
0.1 vs. 0.6	−0.458	0.238	−0.411	−0.458	No statistical diff between 0.1 and 0.6

Table 32. *Cont.*

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.7	0.087	0.995	0.387	0.088	No statistical diff between 0.1 and 0.7
0.1 vs. 0.8	0.080	1.165	0.455	0.080	No statistical diff between 0.1 and 0.8
0.1 vs. 0.9	−0.251	0.169	−0.228	−0.251	No statistical diff between 0.1 and 0.9
0.1 vs. 1	−0.029	0.729	0.165	−0.029	No statistical diff between 0.1 and 1.0

Table 33. Welch's *t*-test: Shellcode: SVM-SMOTE oversampling using KNN = 5 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.261	−0.922	−0.047	0.261	No statistical diff between 0.1 and 0.2
0.1 vs. 0.3	0.551	−1.321	0.148	0.551	No statistical diff between 0.1 and 0.3
0.1 vs. 0.4	0.550	−0.896	0.281	0.549	No statistical diff between 0.1 and 0.4
0.1 vs. 0.5	1.127	0.154	1.194	1.128	No statistical diff between 0.1 and 0.5
0.1 vs. 0.6	−0.656	0.228	−0.502	−0.656	No statistical diff between 0.1 and 0.6
0.1 vs. 0.7	1.575	−0.348	1.192	1.575	0.1 is better than 0.7 in precision and macro precision
0.1 vs. 0.8	1.302	−1.370	0.996	1.302	No statistical diff between 0.1 and 0.8
0.1 vs. 0.9	−1.140	−0.527	−1.293	−1.141	No statistical diff between 0.1 and 0.9
0.1 vs. 1	0.601	−1.189	0.191	0.600	No statistical diff between 0.1 and 1.0

Table 34. Welch's *t*-test: Shellcode: SVM-SMOTE oversampling using KNN = 10 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	2.545	0.361	3.248	2.546	0.1 has better precision, f-Score and macro precision
0.1 vs. 0.3	0.443	0.883	0.721	0.443	No statistical difference between 0.1 and 0.3
0.1 vs. 0.4	1.095	2.170	2.072	1.096	0.1 has better recall and F-Score than 0.4
0.1 vs. 0.5	2.668	2.578	3.245	2.669	0.1 is better than 0.5 across all metrics
0.1 vs. 0.6	2.272	3.345	2.970	2.273	0.1 is better than 0.6 across all metrics
0.1 vs. 0.7	3.784	1.184	4.536	3.786	0.1 is better than 0.7 in precision, F-Score and macro precision
0.1 vs. 0.8	1.416	1.995	1.974	1.417	0.1 has better than 0.8 in precision, F-Score
0.1 vs. 0.9	2.069	2.263	2.546	2.070	0.1 is better than 0.9 across all metrics
0.1 vs. 1	2.347	2.071	2.613	2.348	0.1 is better than 1.0 across all metrics

8.3.3. Backdoors: SVM-SMOTE Oversampling Varying KNN Followed by Random Undersampling

In this section, SVM-SMOTE oversampling percentages are increased from 0.1 to 1.0, keeping the undersampling percent at a constant 0.5 for backdoors. Tables 35–37 present the results of the averages of the metrics for various SVM-SMOTE oversampling percentages for backdoors. As highlighted in the tables, 0.3 oversampling gave better results for KNN = 3 while 0.1 gave better results for the other two KNN variations, 5 and 10.

Table 38 captures the Welch's *t*-test analysis between metrics of varying oversampling percentages for backdoors. Initially 0.2 oversampling has better precision and macro-precision than 0.1, but it lags behind 0.3 in recall. The higher oversampling percentages do not have any statistical difference with 0.3.

Table 35. UNSW-NB15 Backdoors: SVM-SMOTE oversampling using KNN = 3 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.915	0.945	0.930	0.957
0.2	0.936	0.931	0.933	0.967
0.3	0.929	0.945	0.937	0.964
0.4	0.928	0.939	0.934	0.964
0.5	0.934	0.937	0.935	0.966
0.6	0.934	0.940	0.937	0.967
0.7	0.930	0.934	0.932	0.965
0.8	0.935	0.933	0.934	0.967
0.9	0.928	0.936	0.932	0.964
1.0	0.929	0.935	0.932	0.964

Table 36. Backdoors: SVM-SMOTE oversampling using KNN = 5 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.919	0.947	0.933	0.959
0.2	0.916	0.948	0.931	0.958
0.3	0.909	0.949	0.929	0.954
0.4	0.913	0.945	0.929	0.956
0.5	0.908	0.945	0.926	0.954
0.6	0.926	0.938	0.932	0.963
0.7	0.926	0.934	0.930	0.962
0.8	0.926	0.937	0.932	0.963
0.9	0.912	0.946	0.928	0.956
1.0	0.914	0.940	0.927	0.957

Table 37. Backdoors: SVM-SMOTE oversampling using KNN = 10 followed by RU.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.914	0.957	0.935	0.957
0.2	0.907	0.950	0.928	0.953
0.3	0.913	0.950	0.931	0.956
0.4	0.916	0.943	0.929	0.958
0.5	0.920	0.949	0.934	0.960
0.6	0.899	0.950	0.924	0.949
0.7	0.895	0.945	0.919	0.947
0.8	0.918	0.943	0.930	0.959
0.9	0.906	0.945	0.925	0.952
1.0	0.912	0.952	0.931	0.956

Table 38. Welch's *t*-test: Backdoors: SVM-SMOTE oversampling using KNN = 3 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	−2.599	2.404	−0.624	−2.596	0.2 has better precision and macro precision than 0.1 but the latter has better recall
0.2 vs. 0.3	0.874	−2.029	−0.582	0.872	0.3 has better recall than 0.2
0.3 vs. 0.4	0.143	0.829	0.607	0.144	No statistical difference between 0.3 and 0.4
0.3 vs. 0.5	−0.568	1.040	0.210	−0.567	No statistical difference between 0.3 and 0.5

Table 38. *Cont.*

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.3 vs. 0.6	−0.613	0.772	−0.018	−0.612	No statistical difference between 0.3 and 0.6
0.3 vs. 0.7	−0.111	1.919	0.992	−0.110	0.3 has better recall than 0.7
0.3 vs. 0.8	−0.898	1.698	0.459	−0.896	0.3 has better recall than 0.8
0.3 vs. 0.9	0.034	1.255	0.876	0.035	No statistical difference between 0.3 and 0.9
0.3 vs. 1.0	−0.021	1.198	0.604	−0.020	No statistical difference between 0.3 and 1.0

Welch's *t*-test analysis for KNN = 5 and 10 are presented in Table 39 and Table 40 respectively. For both KNN = 5 and 10, 0.1 oversampling either performed better at one of the metrics or had no statistical differences with the higher oversampling percentages.

Table 39. Welch's *t*-test: Backdoors: SVM-SMOTE oversampling using KNN = 5 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.235	−0.177	0.182	0.235	No statistical diff between 0.1 and 0.2
0.1 vs. 0.3	0.912	−0.390	0.757	0.912	No statistical diff between 0.1 and 0.3
0.1 vs. 0.4	0.471	0.508	0.662	0.471	No statistical diff between 0.1 and 0.4
0.1 vs. 0.5	0.713	0.345	0.952	0.714	No statistical diff between 0.1 and 0.5
0.1 vs. 0.6	−0.769	1.748	0.219	−0.768	0.1 has better recall than 0.6
0.1 vs. 0.7	−0.651	3.147	0.576	−0.650	0.1 has better recall than 0.7
0.1 vs. 0.8	−0.875	2.508	0.279	−0.874	0.1 has better recall than 0.8
0.1 vs. 0.9	0.751	0.264	0.892	0.752	No statistical diff between 0.1 and 0.9
0.1 vs. 1	0.364	1.699	0.899	0.364	0.1 has better recall than 1.0

Table 40. Welch's *t*-test: Backdoors: SVM-SMOTE oversampling using KNN = 10 followed by RU.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	1.162	1.345	1.747	1.163	0.1 has better F-Score than 0.2
0.1 vs. 0.3	0.138	1.515	0.649	0.138	No statistical diff between 0.1 and 0.3
0.1 vs. 0.4	−0.317	1.882	1.102	−0.315	0.1 has better recall than 0.4
0.1 vs. 0.5	−0.881	1.124	0.077	−0.880	No statistical diff between 0.1 and 0.4
0.1 vs. 0.6	1.932	1.063	1.656	1.932	0.1 has better precision, F-Score and macro precision than 0.6
0.1 vs. 0.7	3.183	2.461	3.993	3.185	0.1 is better than 0.7 across all metrics
0.1 vs. 0.8	−0.697	2.812	1.129	−0.694	0.1 has better F-Score than 0.8
0.1 vs. 0.9	1.062	1.681	2.055	1.064	0.1 has better recall and F-Score than 0.9
0.1 vs. 1	0.248	0.986	0.732	0.248	No statistical diff between 0.1 and 1.0

8.4. UNSW-NB15: Random Undersampling Followed by BMOTE Oversampling

In this section, results of the experiments carried out with RU of majority data followed by oversampling of minority data are presented. Since RU is performed first, the count of majority class is reduced, and therefore, the oversampling of minority data required to meet up to the majority data is reduced.

8.4.1. Worms: Random Undersampling Varying KNN Followed by BSMOTE Oversampling

In UNSW-NB15, Worms is the smallest of the minority classes. RU of the majority or benign data is performed first. RU is used to bring the majority class to 0.5 of the original count. This is followed by BSMOTE oversampling of the minority data, worms. Results are presented for KNN = 3, 5, and 10 in Table 41, Table 42 and Table 43, respectively. Welch's *t*-test scores are calculated for the averages of the metrics, and it is found that 0.2 oversampling gave better results for KNN = 3, as shown in Table 44. For KNN = 5 and 10, 0.1 BSMOTE oversampling gave better results as shown in Table 45 and Table 46 respectively.

Table 41. Worms: RU followed by BSMOTE oversampling using KNN = 3.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.693	0.738	0.712	0.846
0.2	0.705	0.838	0.766	0.852
0.3	0.756	0.765	0.757	0.878
0.4	0.666	0.753	0.702	0.833
0.5	0.643	0.715	0.676	0.821
0.6	0.694	0.753	0.717	0.847
0.7	0.712	0.776	0.741	0.856
0.8	0.716	0.757	0.735	0.858
0.9	0.711	0.753	0.729	0.855
1.0	0.664	0.738	0.694	0.832

Table 42. UNSW-NB15 Worms: RU followed by BSMOTE oversampling using KNN = 5.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.630	0.830	0.715	0.815
0.2	0.586	0.700	0.635	0.793
0.3	0.630	0.750	0.682	0.815
0.4	0.552	0.676	0.601	0.776
0.5	0.617	0.765	0.681	0.808
0.6	0.590	0.765	0.664	0.795
0.7	0.592	0.753	0.663	0.796
0.8	0.568	0.738	0.639	0.784
0.9	0.612	0.750	0.671	0.806
1.0	0.528	0.761	0.623	0.764

Table 43. Worms: RU followed by BSMOTE oversampling using KNN = 10.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.509	0.784	0.617	0.754
0.2	0.507	0.738	0.600	0.753
0.3	0.473	0.780	0.588	0.736
0.4	0.492	0.761	0.595	0.746
0.5	0.483	0.723	0.578	0.741
0.6	0.471	0.711	0.567	0.735
0.7	0.465	0.753	0.574	0.732
0.8	0.493	0.792	0.606	0.746
0.9	0.490	0.769	0.597	0.744
1.0	0.546	0.796	0.645	0.773

Table 44. Welch's *t*-test: Worms: RU followed by BSMOTE oversampling using KNN = 3.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	−0.318	−2.276	−1.529	−0.318	0.2 has better recall than 0.1
0.2 vs. 0.3	−1.223	1.811	0.252	−1.223	0.2 has better recall than 0.3
0.2 vs. 0.4	0.817	2.222	1.796	0.817	0.2 has better recall and F-Score than 0.4
0.2 vs. 0.5	2.018	2.578	2.600	2.018	0.2 is better than 0.5 across all metrics
0.2 vs. 0.6	0.257	2.178	1.468	0.258	0.2 has better recall than 0.6
0.2 vs. 0.7	−0.149	1.296	0.603	−0.149	0.2 and 0.7 are statistically equal
0.2 vs. 0.8	−0.256	2.016	0.787	−0.256	0.2 has better recall than 0.8
0.2 vs. 0.9	−0.115	2.136	0.928	−0.115	0.2 has better recall than 0.9
0.2 vs. 1.0	1.150	1.827	2.084	1.150	0.2 has better recall and F-Score than 1.0

Table 45. Welch's *t*-test: Worms: RU followed by BSMOTE oversampling using KNN = 5.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.719	3.506	1.645	0.719	0.1 is better than 0.2 in recall and F-Score
0.1 vs. 0.3	−0.001	2.156	0.841	−0.001	0.1 is better than 0.3 in recall
0.1 vs. 0.4	1.628	2.584	2.613	1.629	0.1 is better than 0.4 across all metrics
0.1 vs. 0.5	0.293	1.380	0.804	0.294	No statistical difference between 0.1 and 0.5
0.1 vs. 0.6	0.913	1.581	1.335	0.913	0.1 is better than 0.6 in recall
0.1 vs. 0.7	0.939	1.946	1.367	0.939	0.1 is better than 0.7 in recall
0.1 vs. 0.8	1.601	1.809	2.064	1.601	0.1 is better than 0.8 across all metrics
0.1 vs. 0.9	0.394	1.597	1.013	0.394	0.1 is better than 0.9 in recall
0.1 vs. 1	2.719	1.837	2.564	2.719	0.1 is better than 1.0 across all metrics

Table 46. Welch's *t*-test: Worms: RU followed by BSMOTE oversampling using KNN = 10.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.114	1.325	0.777	0.114	No statistical difference between 0.1 and 0.2
0.1 vs. 0.3	1.735	0.113	1.274	1.735	0.1 is better than 0.3 in precision and macro-precision
0.1 vs. 0.4	0.437	0.577	0.640	0.437	No statistical difference between 0.1 and 0.4
0.1 vs. 0.5	1.137	1.725	1.620	1.137	0.1 is better than 0.5 in recall and F-Score
0.1 vs. 0.6	1.591	1.714	1.781	1.592	0.1 is better than 0.6 across all metrics
0.1 vs. 0.7	1.539	0.771	1.373	1.539	0.1 is better than 0.7 in precision and macro-precision
0.1 vs. 0.8	0.812	−0.166	0.441	0.811	No statistical difference between 0.1 and 0.8
0.1 vs. 0.9	1.238	0.337	0.914	1.238	No statistical difference between 0.1 and 0.9
0.1 vs. 1	−0.900	−0.309	−0.799	−0.900	No statistical difference between 0.1 and 1.0

8.4.2. Shellcode: Random Undersampling Followed by BSMOTE Oversampling Varying KNN

For Shellcode, when RU was followed by BSMOTE oversampling, all three KNN values, 3, 5, and 10, gave better results at 0.1 oversampling, as shown in Table 47, Table 48 and Table 49 respectively. Welch's *t*-test analysis of the metrics from RU followed by BSMOTE oversampling of Shellcode with varying KNN values are presented in Table 50,

Table 51 and Table 52 respectively. It is clear from the analysis that 0.1 oversampling results are better than higher percentages for all three KNN values.

Table 47. Shellcode: RU followed by BSMOTE oversampling using KNN = 3.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.718	0.909	0.802	0.859
0.2	0.699	0.905	0.788	0.849
0.3	0.702	0.920	0.796	0.851
0.4	0.688	0.903	0.780	0.844
0.5	0.687	0.915	0.785	0.843
0.6	0.697	0.906	0.787	0.848
0.7	0.689	0.905	0.782	0.844
0.8	0.695	0.906	0.786	0.847
0.9	0.708	0.901	0.793	0.854
1.0	0.707	0.917	0.798	0.853

Table 48. Shellcode: RU followed by BSMOTE oversampling using KNN = 5.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.694	0.916	0.789	0.847
0.2	0.697	0.914	0.791	0.848
0.3	0.673	0.905	0.772	0.836
0.4	0.676	0.915	0.777	0.838
0.5	0.677	0.923	0.781	0.838
0.6	0.666	0.906	0.767	0.833
0.7	0.678	0.915	0.778	0.839
0.8	0.673	0.910	0.774	0.836
0.9	0.673	0.902	0.771	0.836
1.0	0.657	0.919	0.766	0.828

Table 49. Shellcode: RU followed by BSMOTE oversampling using KNN = 10.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.665	0.922	0.772	0.832
0.2	0.645	0.921	0.758	0.822
0.3	0.652	0.925	0.765	0.826
0.4	0.649	0.920	0.761	0.824
0.5	0.637	0.939	0.759	0.818
0.6	0.643	0.922	0.758	0.821
0.7	0.655	0.911	0.762	0.827
0.8	0.657	0.935	0.772	0.828
0.9	0.644	0.922	0.758	0.822
1.0	0.631	0.928	0.751	0.815

Table 50. Welch's *t*-test: Shellcode: RU followed by BSMOTE oversampling for KNN = 3.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	1.985	0.811	2.133	1.985	0.1 is statistically better than 0.2 in precision, F-Score and macro precision
0.1 vs. 0.3	2.223	−1.469	1.257	2.222	0.1 is better than 0.3 in precision and macro precision.

Table 50. *Cont.*

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.4	2.211	0.987	2.933	2.212	0.1 is better than 0.4 in precision, f-score and macro precision.
0.1 vs. 0.5	2.569	−1.023	2.167	2.569	0.1 is better than 0.5 in precision, f-score and macro precision.
0.1 vs. 0.6	2.714	0.398	3.320	2.715	0.1 is better than 0.6 in precision, f-score and macro precision.
0.1 vs. 0.7	3.947	0.775	5.141	3.948	0.1 is better than 0.7 in precision, f-score and macro precision.
0.1 vs. 0.8	2.189	0.488	2.401	2.190	0.1 is better than 0.8 in precision, f-score and macro precision.
0.1 vs. 0.9	2.501	0.654	1.628	2.501	0.1 is better than 0.9 in precision, f-score and macro precision.
0.1 vs. 1	1.260	−0.881	0.773	1.260	No statistical difference between 0.1 and 1

Table 51. Welch's *t*-test: Shellcode: RU followed by BSMOTE oversampling for KNN = 5.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	−0.313	0.312	−0.190	−0.313	No statistical difference between 0.1 and 0.2
0.1 vs. 0.3	2.232	1.188	2.809	2.233	0.1 is statistically better than 0.3 in precision, F-Score and macro-precision
0.1 vs. 0.4	2.129	0.100	1.744	2.129	0.1 is statistically better than 0.4 in precision, F-Score and macro-precision
0.1 vs. 0.5	1.925	−0.884	1.407	1.924	0.1 is statistically better than 0.5 in precision and macro-precision
0.1 vs. 0.6	2.937	1.144	3.344	2.938	0.1 is statistically better than 0.6 in precision, F-Score and macro-precision
0.1 vs. 0.7	1.496	0.181	1.459	1.496	No statistical difference between 0.1 and 0.7
0.1 vs. 0.8	2.306	0.731	2.373	2.306	0.1 is statistically better than 0.8 in precision, F-Score and macro-precision
0.1 vs. 0.9	2.133	2.963	0.000	2.134	0.1 is better than 0.9 across all metrics
0.1 vs. 1	3.678	−0.536	3.496	3.678	0.1 is statistically better than 1.0 in precision, F-Score and macro-precision

Table 52. Welch's *t*-test: Shellcode: RU followed by BSMOTE oversampling for KNN = 10.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	1.463	0.245	1.405	1.463	0.1 and 0.2 are statistically equal
0.1 vs. 0.3	1.088	−0.589	0.867	1.088	0.1 and 0.3 are statistically equal
0.1 vs. 0.4	1.275	0.429	1.270	1.275	0.1 and 0.4 are statistically equal
0.1 vs. 0.5	2.761	−2.422	1.815	2.760	0.1 is better than 0.5 in precision, F-Score and macro precision. 0.5 is better than 0.1 in recall.
0.1 vs. 0.6	1.991	0.067	1.771	1.991	0.1 is better than 0.6 in precision, F-Score and macro precision.
0.1 vs. 0.7	0.877	1.873	1.331	0.878	0.1 is better than 0.7 in recall.

Table 52. *Cont.*

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.8	0.709	−3.522	0.103	0.708	0.1 is better than 0.8 in recall.
0.1 vs. 0.9	2.043	0.056	1.921	2.043	0.1 is better than 0.9 in precision, F-Score and macro precision.
0.1 vs. 1	2.199	−0.713	2.296	2.199	0.1 is better than 1.0 in precision, F-Score and macro precision.

8.4.3. Backdoors: Random Undersampling Followed by BSMOTE Oversampling Varying KNN

Results for Backdoors are presented in Table 53, Table 54, and Table 55 for KNN = 3, 5 and 10 respectively. Probabilistic analysis of the metrics for KNN = 3 reveals that a 0.2 oversampling ratio has better precision, F-Score, and macro precision than 0.1. It did not have any statistical difference with 0.3 but had better recall than 0.4. No statistical differences were observed in the results for all the higher oversampling percentages, as presented in Table 56. The 10% oversampling ratio gave better KNN = 5 and 10 results in Welch's *t*-test analysis of the metrics, as presented in Table 57 and Table 58 respectively.

Table 53. Backdoors: RU followed by BSMOTE oversampling using KNN = 3.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.936	0.949	0.942	0.967
0.2	0.945	0.953	0.949	0.972
0.3	0.943	0.957	0.950	0.971
0.4	0.946	0.947	0.946	0.972
0.5	0.935	0.951	0.943	0.967
0.6	0.940	0.949	0.944	0.970
0.7	0.942	0.951	0.946	0.971
0.8	0.941	0.950	0.945	0.970
0.9	0.936	0.957	0.946	0.968
1.0	0.943	0.947	0.945	0.971

Table 54. UNSW-NB15 Backdoors: RU followed by BSMOTE oversampling using KNN = 5.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.934	0.959	0.946	0.967
0.2	0.938	0.953	0.945	0.969
0.3	0.941	0.951	0.946	0.970
0.4	0.929	0.954	0.941	0.964
0.5	0.936	0.954	0.945	0.968
0.6	0.934	0.948	0.941	0.967
0.7	0.931	0.955	0.943	0.965
0.8	0.940	0.951	0.945	0.970
0.9	0.930	0.953	0.941	0.965
1.0	0.939	0.955	0.947	0.969

Table 55. UNSW-NB15 Backdoors: RU followed by BSMOTE oversampling using KNN = 10.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.933	0.961	0.947	0.966
0.2	0.932	0.953	0.942	0.966
0.3	0.932	0.962	0.947	0.966

Table 55. *Cont.*

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.4	0.913	0.954	0.933	0.956
0.5	0.924	0.949	0.937	0.962
0.6	0.931	0.955	0.943	0.965
0.7	0.931	0.953	0.942	0.965
0.8	0.918	0.959	0.938	0.959
0.9	0.920	0.959	0.939	0.960
1.0	0.922	0.949	0.935	0.960

Table 56. Welch's *t*-test: Backdoors: RU followed by BSMOTE oversampling for KNN = 3.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	−1.685	−1.207	−2.040	−1.686	0.2 is better than 0.1 in precision, F-Score and macro-precision
0.2 vs. 0.3	0.300	−0.790	−0.437	0.300	No statistical difference between 0.2 and 0.3
0.2 vs. 0.4	−0.208	1.660	1.007	−0.207	0.2 is better than 0.4 in recall
0.2 vs. 0.5	1.429	0.564	1.487	1.429	No statistical difference between 0.2 and 0.5
0.2 vs. 0.6	0.654	0.750	1.289	0.655	No statistical difference between 0.2 and 0.6
0.2 vs. 0.7	0.366	0.959	0.570	0.366	No statistical difference between 0.2 and 0.7
0.2 vs. 0.8	0.483	0.447	0.731	0.484	No statistical difference between 0.2 and 0.8
0.2 vs. 0.9	1.487	−1.045	0.932	1.486	No statistical difference between 0.2 and 0.9
0.2 vs. 1	0.154	0.810	1.387	0.154	No statistical difference between 0.2 and 1.0

Table 57. Welch's *t*-test: UNSW-NB15 Backdoors: RU followed by BSMOTE oversampling for KNN = 5.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	−0.679	1.136	0.259	−0.678	No statistical difference between 0.1 and 0.2
0.1 vs. 0.3	−1.241	1.876	0.099	−1.240	0.1 has better recall than 0.3
0.1 vs. 0.4	0.636	1.051	1.309	0.637	No statistical difference between 0.1 and 0.4
0.1 vs. 0.5	−0.343	1.636	0.523	−0.342	0.1 has better recall than 0.5
0.1 vs. 0.6	−0.072	3.239	2.171	−0.070	0.1 has better recall and F-Score
0.1 vs. 0.7	0.777	0.646	0.986	0.778	No statistical difference between 0.1 and 0.7
0.1 vs. 0.8	−1.285	1.925	0.293	−1.283	0.1 has better recall than 0.8
0.1 vs. 0.9	0.454	1.394	1.262	0.455	No statistical difference between 0.1 and 0.9
0.1 vs. 1	−1.314	0.981	−0.238	−1.314	No statistical difference between 0.1 and 1.0

Table 58. Welch's *t*-test: UNSW-NB15 Backdoors: RU followed by BSMOTE oversampling for KNN = 10.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.139	1.792	1.178	0.140	0.1 has better than 0.2
0.1 vs. 0.3	0.175	−0.436	0.037	0.175	No statistical difference between 0.1 and 0.3
0.1 vs. 0.4	3.092	1.520	4.218	3.095	0.1 has better precision, F-Score and macro precision
0.1 vs. 0.5	1.185	3.461	2.171	1.186	0.1 has better recall and F-Score

Table 58. *Cont.*

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.6	0.297	1.331	0.982	0.298	No statistical difference between 0.1 and 0.6
0.1 vs. 0.7	0.271	2.853	1.090	0.272	0.1 has better recall than 0.7
0.1 vs. 0.8	2.282	0.532	2.166	2.283	0.1 has better precision, F-Score and macro precision
0.1 vs. 0.9	1.592	0.820	2.369	1.593	0.1 has better precision, F-Score and macro precision
0.1 vs. 1	1.407	2.129	3.498	1.409	0.1 has better recall and F-Score

8.5. Random Undersampling Followed by SVM-SMOTE Oversampling

In this section, results of the experiments carried out with RU of majority data followed by oversampling of minority data using SVM-SMOTE are presented.

8.5.1. Worms: Random Undersampling Followed by SVM-SMOTE Oversampling Varying KNN

Worms have better prediction when it is oversampled to 0.1 using SVM-SMOTE for KNN = 3 and 5 when the majority non-attack data is undersampled to 0.5, as shown in Table 59 and Table 60 respectively. For KNN = 10, however, it takes 0.4 oversampling using SVM-SMOTE to give overall better results, as shown in Table 61. It can be observed that precision and F-score decreased when KNN was increased from 3 to 10. This might be due to the imbalanced nature of the data. As the number of neighbors increases, more points from the majority dataset are included, and thus there is a higher chance of the actual point being classified incorrectly.

Table 59. Worms: RU followed by SVM-SMOTE oversampling using KNN = 3.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.645	0.796	0.709	0.822
0.2	0.564	0.715	0.626	0.782
0.3	0.618	0.773	0.685	0.809
0.4	0.609	0.803	0.692	0.804
0.5	0.615	0.723	0.660	0.807
0.6	0.647	0.753	0.695	0.823
0.7	0.612	0.807	0.696	0.806
0.8	0.601	0.757	0.668	0.800
0.9	0.635	0.811	0.710	0.817
1.0	0.645	0.746	0.690	0.822

Table 60. Worms: RU followed by SVM-SMOTE oversampling using KNN = 5.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.574	0.819	0.673	0.787
0.2	0.583	0.769	0.662	0.791
0.3	0.553	0.773	0.639	0.776
0.4	0.547	0.769	0.637	0.773
0.5	0.508	0.784	0.613	0.754
0.6	0.514	0.780	0.617	0.757
0.7	0.457	0.692	0.549	0.728
0.8	0.548	0.696	0.612	0.774
0.9	0.535	0.734	0.617	0.767
1.0	0.593	0.788	0.674	0.796

Table 61. Worms: RU followed by SVM-SMOTE oversampling using KNN = 10.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.491	0.753	0.593	0.745
0.2	0.465	0.788	0.585	0.732
0.3	0.484	0.792	0.599	0.742
0.4	0.460	0.834	0.592	0.730
0.5	0.470	0.788	0.587	0.735
0.6	0.451	0.807	0.578	0.725
0.7	0.449	0.769	0.566	0.724
0.8	0.482	0.723	0.577	0.741
0.9	0.469	0.753	0.577	0.734
1.0	0.428	0.753	0.544	0.714

Tables 62 and 63 capture the Welch's *t*-test analysis performed on the metrics from Worms where RU is performed first, followed by the SVM-SMOTE oversampling for KNN = 3 and 5. In both of these runs, 0.1 had no statistical difference or better results in at least one of the metrics than higher percentages. For KNN = 10, 0.4 oversampling gave better results. This is captured in Table 64.

Table 62. Welch's *t*-test: Worms: RU followed by SVM-SMOTE oversampling using KNN = 3.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	2.191	2.377	4.917	2.192	0.1 is better than 0.3 across all metrics
0.1 vs. 0.3	0.786	0.738	1.398	0.786	No statistical difference between 0.1 and 0.3
0.1 vs. 0.4	1.150	−0.267	1.062	1.150	No statistical difference between 0.1 and 0.4
0.1 vs. 0.5	0.753	2.068	2.561	0.754	0.1 has better recall and F-Score than 0.5
0.1 vs. 0.6	−0.044	1.261	0.529	−0.044	No statistical diff between 0.1 and 0.6
0.1 vs. 0.7	0.911	−0.287	0.457	0.911	No statistical diff between 0.1 and 0.7
0.1 vs. 0.8	1.302	1.018	1.850	1.302	0.1 has better F-Score than 0.7
0.1 vs. 0.9	0.252	−0.452	−0.043	0.252	No statistical diff between 0.1 and 0.9
0.1 vs. 1	0.015	1.600	0.610	0.015	0.1 has better recall than 1.0

Table 63. Welch's *t*-test: Worms: RU followed by SVM-SMOTE oversampling using KNN = 5.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	−0.233	2.076	0.331	−0.233	0.1 has better recall than 0.2
0.1 vs. 0.3	0.381	3.794	1.048	0.381	0.1 has better recall than 0.3
0.1 vs. 0.4	0.798	1.408	1.358	0.798	No statistical diff between 0.2 and 0.3
0.1 vs. 0.5	1.676	0.948	1.866	1.676	0.1 has better precision, F-Score and macro precision than 0.5
0.1 vs. 0.6	1.490	2.839	1.900	1.490	0.1 has better recall and F-Score than 0.6
0.1 vs. 0.7	3.023	4.288	3.745	3.024	0.1 is better than 0.7 across all metrics
0.1 vs. 0.8	0.681	3.938	1.989	0.682	0.1 has better recall and F-Score than 0.8
0.1 vs. 0.9	1.055	2.429	1.746	1.056	0.1 has better recall and F-Score than 0.9
0.1 vs. 1	−0.474	1.928	−0.047	−0.474	0.1 has better recall than 1.0

Table 64. Welch's *t*-test: UNSW-NB15 Worms: RU followed by SVM-SMOTE oversampling using KNN = 10.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.807	−0.977	0.271	0.806	No statistical diff between 0.1 and 0.2
0.1 vs. 0.3	0.175	−1.274	−0.171	0.175	No statistical diff between 0.1 and 0.3
0.1 vs. 0.4	0.920	−2.757	0.023	0.920	0.4 has better recall than 0.1
0.4 vs. 0.5	−0.276	1.533	0.160	−0.276	0.4 has better recall than 0.5
0.4 vs. 0.6	0.587	1.179	0.949	0.587	No statistical diff between 0.4 and 0.6
0.4 vs. 0.7	0.474	1.673	0.959	0.474	0.4 has better recall than 0.7
0.4 vs. 0.8	−0.744	5.094	0.674	−0.743	0.4 has better recall than 0.8
0.4 vs. 0.9	−0.401	2.002	0.664	−0.401	0.4 has better recall than 0.9
0.4 vs. 1.0	1.544	2.509	2.511	1.544	0.4 is better than 1.0 across all metrics

8.5.2. Shellcode: Random Undersampling Followed by SVM-SMOTE Oversampling Varying KNN

In this section, benign data is first undersampled (to 0.5), followed by SVM-SMOTE oversampling using varying KNN. Tables 65–67 present the results for KNN = 3, 5, and 10 respectively. From Table 65 it can be noted that, for KNN = 3, 0.4 oversampling gave better results overall. Welch's *t*-test calculations, Table 68, showed that 0.4 has better recall than oversampling percentages below 0.4, and there were no statistical differences with higher oversampling percentages. Also, 0.4 had a better F-score than 0.7 and no statistical difference with higher percentages. For KNN = 5 and 10, however, Welch's *t*-test analysis showed that the results were better at 0.1 oversampling, as shown in Table 69 and Table 70, respectively.

Table 65. Shellcode: RU followed by SVM-SMOTE oversampling using KNN = 3.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.704	0.901	0.790	0.852
0.2	0.704	0.905	0.792	0.851
0.3	0.696	0.911	0.789	0.848
0.4	0.693	0.910	0.787	0.846
0.5	0.694	0.912	0.788	0.847
0.6	0.693	0.922	0.791	0.846
0.7	0.681	0.905	0.777	0.840
0.8	0.694	0.910	0.787	0.847
0.9	0.695	0.915	0.790	0.847
1.0	0.688	0.910	0.784	0.844

Table 66. Shellcode: RU followed by SVM-SMOTE oversampling using KNN = 5.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.693	0.922	0.791	0.846
0.2	0.682	0.921	0.783	0.841
0.3	0.674	0.925	0.780	0.837
0.4	0.677	0.921	0.781	0.838
0.5	0.689	0.911	0.785	0.844
0.6	0.673	0.922	0.778	0.836
0.7	0.665	0.919	0.771	0.832

Table 66. *Cont.*

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.8	0.682	0.925	0.785	0.841
0.9	0.658	0.932	0.771	0.829
1.0	0.674	0.921	0.778	0.837

Table 67. Shellcode: RU followed by SVM-SMOTE oversampling using KNN = 10.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.679	0.920	0.781	0.839
0.2	0.653	0.924	0.765	0.826
0.3	0.647	0.923	0.761	0.823
0.4	0.647	0.932	0.764	0.823
0.5	0.653	0.916	0.762	0.826
0.6	0.645	0.917	0.757	0.822
0.7	0.655	0.922	0.766	0.827
0.8	0.662	0.927	0.772	0.831
0.9	0.635	0.924	0.753	0.817
1.0	0.660	0.928	0.771	0.830

Table 68. Welch's *t*-test: Shellcode: RU followed by SVM-SMOTE oversampling using KNN = 3.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.019	−0.964	−0.240	0.019	No statistical diff between 0.1 and 0.2
0.1 vs. 0.3	0.778	−1.324	0.119	0.777	No statistical diff between 0.1 and 0.3
0.1 vs. 0.4	1.150	−2.200	0.539	1.149	0.4 has better recall than 0.1
0.4 vs. 0.5	−0.064	−0.327	−0.083	−0.064	No statistical diff between 0.4 and 0.5
0.4 vs. 0.6	−0.050	−1.114	−0.730	−0.051	No statistical diff between 0.4 and 0.6
0.4 vs. 0.7	1.015	0.656	1.555	1.015	0.4 has better F-Score than 0.7
0.4 vs. 0.8	−0.116	0.000	−0.096	−0.116	No statistical diff between 0.4 and 0.8
0.4 vs. 0.9	−0.285	−0.943	−0.589	−0.286	No statistical diff between 0.4 and 0.9
0.4 vs. 1.0	0.573	0.000	0.660	0.573	No statistical diff between 0.4 and 1.0

Table 69. Welch's *t*-test: UNSW-NB15 Shellcode: RU followed by SVM-SMOTE oversampling using KNN = 5.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.657	0.396	0.672	0.657	No statistical diff between 0.1 and 0.2
0.1 vs. 0.3	1.238	−0.652	1.235	1.238	No statistical diff between 0.1 and 0.3
0.1 vs. 0.4	1.250	0.164	1.401	1.251	No statistical diff between 0.1 and 0.4
0.1 vs. 0.5	0.230	2.988	0.680	0.231	0.1 has better recall than 0.5
0.1 vs. 0.6	1.618	0.078	1.798	1.618	0.1 has better precision, F-Score and macro
0.1 vs. 0.7	1.993	1.006	2.203	1.993	0.1 has better precision, F-Score and macro
0.1 vs. 0.8	0.828	−0.676	0.689	0.828	No statistical diff between 0.1 and 0.8
0.1 vs. 0.9	2.968	−1.813	2.707	2.967	0.1 has better precision, F-Score and macro while 0.9 has better recall
0.1 vs. 1	1.404	0.180	1.533	1.404	0.1 has better F-Score

Table 70. Welch's *t*-test: UNSW-NB15 Shellcode: RU followed by SVM-SMOTE oversampling for KNN = 10.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	2.910	−0.429	3.681	2.911	0.1 has better precision, F-Score and macro precision than 0.2
0.1 vs. 0.3	3.491	−0.383	3.042	3.491	0.1 has better precision, F-Score and macro precision than 0.3
0.1 vs. 0.4	3.501	−2.305	2.932	3.501	0.1 is better than 0.4 across all metrics
0.1 vs. 0.5	3.056	0.424	2.845	3.056	0.1 has better precision, F-Score and macro precision than 0.5
0.1 vs. 0.6	3.576	0.443	3.452	3.576	0.1 has better precision, F-Score and macro precision than 0.6
0.1 vs. 0.7	2.259	−0.312	2.414	2.259	0.1 has better precision, F-Score and macro precision than 0.7
0.1 vs. 0.8	1.714	−1.200	1.350	1.713	0.1 has better precision and macro precision than 0.8
0.1 vs. 0.9	6.692	−0.572	6.036	6.692	0.1 has better precision, F-Score and macro precision than 0.9
0.1 vs. 1	1.519	−1.216	1.240	1.518	No statistical diff bet 0.1 and 1.0

8.5.3. Backdoors: Random Undersampling Followed by SVM-SMOTE Oversampling Varying KNN

Tables 71 and 72 show the average values of metrics for varying values of SVM-SMOTE oversampling with KNN = 3 and 5 respectively. Welch's *t*-test scores and the subsequent probabilistic calculations show that 0.1 oversampling gave overall better prediction results for both KNN = 3 and 5 as shown in Tables 73 and 74. For KNN = 3, there were no statistical differences between 0.1 and higher oversampling percentages until 0.6. Also, 0.1 has better precision, recall and macro precision than remaining percentages till 1.0 as shown in Table 71.

Table 71. UNSW-NB15 Backdoors: RU followed by SVM-SMOTE oversampling using KNN = 3.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.937	0.950	0.943	0.968
0.2	0.926	0.945	0.935	0.963
0.3	0.935	0.944	0.939	0.967
0.4	0.921	0.953	0.937	0.960
0.5	0.923	0.951	0.937	0.961
0.6	0.935	0.954	0.944	0.967
0.7	0.919	0.950	0.934	0.959
0.8	0.921	0.946	0.933	0.960
0.9	0.921	0.943	0.932	0.960
1.0	0.917	0.943	0.930	0.958

Table 74 captures the results for Welch's *t*-test analysis for KNN = 5. From Table 74 it can be noted that 0.1 oversampling had no statistical differences across all metrics until 1.0, where the former had a better recall.

Table 75 presents the results for KNN = 10. For KNN = 10, as shown in Table 76, RU followed by 0.2 SVM-SMOTE oversampling gave the best results. *p*-value calculations show that 0.2 had a better recall and F-score than 0.1 and also performed better across all metrics than the other oversampling percentages.

Table 72. UNSW-NB15 Backdoors: RU followed by SVM-SMOTE oversampling using KNN = 5.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.919	0.954	0.936	0.959
0.2	0.917	0.955	0.936	0.958
0.3	0.915	0.955	0.935	0.957
0.4	0.922	0.948	0.935	0.961
0.5	0.915	0.956	0.935	0.957
0.6	0.912	0.955	0.933	0.956
0.7	0.915	0.949	0.932	0.957
0.8	0.916	0.952	0.934	0.958
0.9	0.925	0.945	0.935	0.962
1.0	0.917	0.943	0.930	0.958

Table 73. Welch's *t*-test: UNSW-NB15 Backdoors: RU followed by SVM-SMOTE oversampling for KNN = 3.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.788	1.046	1.207	0.788	No statistical diff between 0.1 and 0.2
0.1 vs. 0.3	0.180	0.882	0.817	0.180	No statistical diff between 0.1 and 0.3
0.1 vs. 0.4	1.433	−0.489	0.997	1.432	No statistical diff between 0.1 and 0.4
0.1 vs. 0.5	1.456	−0.215	1.527	1.457	No statistical diff between 0.1 and 0.5
0.1 vs. 0.6	0.170	−0.674	−0.161	0.169	No statistical diff between 0.1 and 0.6
0.1 vs. 0.7	2.161	0.052	2.743	2.162	0.1 has better precision, recall and macro-precision than 0.7
0.1 vs. 0.8	1.582	0.674	2.439	1.582	0.1 has better precision, recall and macro-precision than 0.8
0.1 vs. 0.9	1.692	1.063	2.059	1.693	0.1 has better precision, recall and macro-precision than 0.9
0.1 vs. 1	2.375	1.149	4.135	2.377	0.1 has better precision, recall and macro-precision than 1.0

Table 74. Welch's *t*-test: UNSW-NB15 Backdoors: RU followed by SVM-SMOTE oversampling for KNN = 5.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	0.387	−0.291	0.214	0.387	No statistical diff between 0.1 and 0.2
0.1 vs. 0.3	0.601	−0.161	0.297	0.601	No statistical diff between 0.1 and 0.3
0.1 vs. 0.4	−0.357	1.140	0.496	−0.357	No statistical diff between 0.1 and 0.4
0.1 vs. 0.5	0.534	−0.320	0.272	0.534	No statistical diff between 0.1 and 0.5
0.1 vs. 0.6	0.825	−0.114	0.871	0.825	No statistical diff between 0.1 and 0.6
0.1 vs. 0.7	0.405	0.776	0.730	0.405	No statistical diff between 0.1 and 0.7
0.1 vs. 0.8	0.645	0.357	0.935	0.645	No statistical diff between 0.1 and 0.8
0.1 vs. 0.9	−0.842	1.354	0.654	−0.841	No statistical diff between 0.1 and 0.9
0.1 vs. 1	0.325	1.697	1.410	0.327	0.1 has better recall than 1.0

Table 75. UNSW-NB15 Backdoors: RU followed by SVM-SMOTE oversampling using KNN = 10.

Oversampling %	Precision	Recall	F-Score	Macro Precision
0.1	0.911	0.957	0.933	0.955
0.2	0.918	0.967	0.942	0.959
0.3	0.907	0.967	0.936	0.953
0.4	0.883	0.955	0.918	0.941
0.5	0.906	0.962	0.933	0.953
0.6	0.894	0.947	0.920	0.947
0.7	0.893	0.958	0.924	0.946
0.8	0.898	0.956	0.926	0.949
0.9	0.909	0.952	0.930	0.954
1.0	0.913	0.953	0.933	0.956

Table 76. Welch's *t*-test: UNSW-NB15 Backdoors: RU followed by SVM-SMOTE oversampling for KNN = 10.

Welch's *t*-Test ($p < 0.10$)	Precision *t* Value	Recall *t* Value	F-Score *t* Value	Macro Precision *t* Value	Analysis
0.1 vs. 0.2	−1.077	−1.716	−1.873	−1.079	0.2 has better recall and F-Score than 0.1
0.2 vs. 0.3	1.078	0.164	1.295	1.078	No statistical diff between 0.2 and 0.3
0.2 vs. 0.4	4.934	3.021	4.653	4.933	0.2 is better than 0.4 across all metrics
0.2 vs. 0.5	1.656	1.915	2.109	1.657	0.2 is better than 0.5 across all metrics
0.2 vs. 0.6	2.830	4.652	3.879	2.832	0.2 is better than 0.6 across all metrics
0.2 vs. 0.7	4.370	1.957	6.059	4.373	0.2 is better than 0.7 across all metrics
0.2 vs. 0.8	3.076	2.285	3.790	3.078	0.2 is better than 0.8 across all metrics
0.2 vs. 0.9	1.000	2.928	2.278	1.002	0.2 has better recall and F-Score than 0.9
0.2 vs. 1.0	0.729	3.945	2.667	0.732	0.2 has better recall and F-Score than 1.0

9. Conclusions and Future Work

This paper analyzed the impact of the following on the minority sample classification:

1. The order of resampling techniques, that is, whether oversampling followed by undersampling or undersampling followed by oversampling, is better;
2. The selection of the oversampling techniques, Borderline SMOTE vs. SVM-SMOTE;
3. The effect of the selection of the KNN value on the oversampling percentage;
4. The selection of the oversampling ratio while keeping the undersampling constant at 50%.

Table 77 presents the best oversampling percentages for the rare attacks in UNSW-NB15.

It is observed that any oversampling percentage containing the best result for any individual metric may not always give the best result overall and vice versa. Based on the requirement and application, oversampling and undersampling percentages, the order of resampling techniques and KNN values have to be considered. Following are the conclusions drawn through the analysis of results and the best oversampling percentages presented in Table 77:

1. 10% oversampling gave better results for both BSMOTE and SVM-SMOTE, irrespective of the order of resampling.
2. SVM-SMOTE gave better prediction results at higher oversampling percentages than BSMOTE.
3. For rarer classes such as Worms, higher KNN led to increase in SVM-SMOTE oversampling percentages in both the orders of resampling.

Table 77. Comparison of best oversampling percentages for UNSW-NB15 minority data.

UNSW-NB15		Oversampling then Undersampling		Undersampling then Oversampling	
	KNN	**BSMOTE**	**SVM-SMOTE**	**BSMOTE**	**SVM-SMOTE**
Worms	3	0.1	0.1	0.2	0.1
	5	0.1	0.1	0.1	0.1
	10	0.1	0.3	0.1	0.4
Shellcode	3	0.1	0.1	0.1	0.4
	5	0.1	0.1	0.1	0.1
	10	0.1	0.1	0.1	0.1
Backdoors	3	0.1	0.3	0.2	0.1
	5	0.1	0.1	0.1	0.1
	10	0.1	0.1	0.1	0.2

For future work, we plan to expand this work to other datasets and other types of datasets and see what impact that will have on the results.

Author Contributions: This work was conceptualized by S.S.B., D.M., S.C.B. and S.S.; methodology was mainly done by S.S.B., D.M., S.S.B. and S.S.; software was done by S.S.; validation was done by S.S.B., D.M., S.C.B. and S.S.; formal analysis was done by S.S.B., D.M., S.C.B. and S.S.; investigation was done by S.S.B., D.M., S.C.B. and S.S.; data curation S.S.; writing—original draft preparation was done by S.S. and S.S.B.; writing—review and editing was done by S.S.B., D.M., S.C.B. and S.S.; visualization was done S.S.; supervision was done by S.S.B., D.M. and S.C.B.; project administration was done by S.S.B. and D.M.; funding acquisition was done by S.S.B., D.M. and S.C.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data is available at datasets.uwf.edu (accessed on 12 September 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cisco. *What Is a Cyberattack?—Most Common Types*; Cisco: San Jose, CA, USA, 2023; Available online: https://www.cisco.com/c/en/us/products/security/common-cyberattacks.html#~how-cyber-attacks-work (accessed on 17 April 2023).
2. *What Is a Cyberattack?* IBM: Armonk, NY, USA; Available online: https://www.ibm.com/topics/cyber-attack (accessed on 17 April 2023).
3. Delplace, A.; Hermoso, S.; Anandita, K. Cyber Attack Detection thanks to Machine Learning Algorithms. *arXiv* **2020**, arXiv:2001.06309. Available online: https://arxiv.org/abs/2001.06309 (accessed on 12 April 2023).
4. Alencar, R. *Resampling Strategies for Imbalanced Datasets*; Kaggle: San Francisco, CA, USA, 2017; Available online: https://www.kaggle.com/code/rafjaa/resampling-strategies-for-imbalanced-datasets (accessed on 17 April 2023).
5. Ahmed, H.A.; Hameed, A.; Bawany, N.Z. Network intrusion detection using oversampling technique and machine learning algorithms. *PeerJ. Comput. Sci.* **2022**, *8*, e820. [CrossRef] [PubMed]
6. Moustafa, N.; Slay, J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6. [CrossRef]
7. Brownlee, J. Random Oversampling and Undersampling for Imbalanced Classification. 2021. Available online: https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/ (accessed on 17 April 2023).
8. Branco, P.S.; Torgo, L.; Ribeiro, R.A. A Survey of Predictive Modelling under Imbalanced Distributions. *arXiv* **2015**. Available online: http://export.arxiv.org/pdf/1505.01658 (accessed on 17 April 2023).
9. Patwardhan, S. Simple Understanding and Implementation of KNN Algorithm! Analytics Vidhya, Gurgaon, New Delhi, India. 2022. Available online: https://www.analyticsvidhya.com/blog/2021/04/simple-understanding-and-implementation-of-knn-algorithm/ (accessed on 25 April 2023).
10. Han, H.; Wang, W.-Y.; Mao, B.-H. Borderline-smote: A new over-sampling method in imbalanced data sets learning. *Lect. Notes Comput. Sci.* **2005**, *3644*, 878–887. [CrossRef]
11. Nguyen, H.M.; Cooper, E.W.; Kamei, K. Borderline over-sampling for Imbalanced Data Classification. *Int. J. Knowl. Eng. Soft Data Paradig.* **2011**, *3*, 4. [CrossRef]

12. Brownlee, J. *Bagging and Random Forest for Imbalanced Classification*; Machine Learning Mastery: Vermont, Australia, 2020; Available online: https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/ (accessed on 17 April 2023).
13. Dini, P.; Saponara, S. Analysis, design, and comparison of machine-learning techniques for networking intrusion detection. *Designs* **2021**, *5*, 9. [CrossRef]
14. Dini, P.; Begni, A.; Ciavarella, S.; De Paoli, E.; Fiorelli, G.; Silvestro, C.; Saponara, S. Design and testing novel one-class classifier based on polynomial interpolation with application to networking security. *IEEE Access* **2022**, *10*, 67910–67924. [CrossRef]
15. Elhanashi, A.; Gasmi, K.; Begni, A.; Dini, P.; Zheng, Q.; Saponara, S. Machine Learning Techniques for Anomaly-Based Detection System on CSE-CIC-IDS2018 Dataset. In *International Conference on Applications in Electronics Pervading Industry, Environment and Society*; Springer Nature: Cham, Switzerland, 2022.
16. Ramyachitra, D.; Manikandan, P. Imbalanced dataset classification and solutions: A review. *Int. J. Comput. Bus. Res. (IJCBR)* **2014**, *5*, 1–29.
17. Ganganwar, V. An overview of classification algorithms for imbalanced datasets. *Int. J. Emerg. Technol. Adv. Eng.* **2012**, *2*, 42–47.
18. Chawla, N.V. Data mining for imbalanced datasets: An overview. In *Data Mining and Knowledge Discovery Handbook*; Springer: Boston, MA, USA, 2010; pp. 875–886.
19. Nguyen, G.H.; Bouzerdoum, A.; Phung, S.L. Learning pattern classification tasks with imbalanced data sets. *Pattern Recognit.* **2009**, 193–208.
20. Abdelkhalek, A.; Mashaly, M. Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning. *J. Supercomput.* **2023**, *79*, 10611–10644. [CrossRef]
21. Eke, H.; Petrovski, A.; Ahriz, H. Handling minority class problem in threats detection based on heterogeneous ensemble learning approach. *Int. J. Syst. Softw. Secur. Prot.* **2020**, *11*, 13–37. [CrossRef]
22. Kumar, S.; Biswas, S.K.; Devi, D. TLUSBoost algorithm: A boosting solution for class imbalance problem. *Soft Comput.* **2019**, *23*, 10755–10767. [CrossRef]
23. Fujiwara, K.; Huang, Y.; Hori, K.; Nishioji, K.; Kobayashi, M.; Kamaguchi, M. Over- and Under-sampling Approach for Extremely Imbalanced and Small Minority Data Problem in Health 860 Record Analysis. *Front. Public Health* **2020**, *8*, 178. [CrossRef]
24. Hasanin, T.; Khoshgoftaar, T. The Effects of Random Undersampling with Simulated Class Imbalance for Big Data. In Proceedings of the 2018 IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, USA, 6–9 July 2018; pp. 70–79. [CrossRef]
25. Weiss, G.; Provost, F. *The Effect of Class Distribution on Classifier Learning: An Empirical Study*; Rutgers University: Camden, NJ, USA, 2001.
26. Silva, E.J.R.; Zanchettin, C. On the Existence of a Threshold in Class Imbalance Problems. In Proceedings of the 2015 IEEE International Conference on Systems, Man, and Cybernetics, Hong Kong, China, 9–12 October 2015; pp. 2714–2719. [CrossRef]
27. Joshi, A.; Kanwar, K.; Vaidya, P.; Sharma, S. A Principal Component Analysis, Sampling and Classifier strategies for dealing with concerns of class imbalance in datasets with a ratio greater than five. In Proceedings of the 2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA), Gunupur, India, 8 September 2022; pp. 1–6. [CrossRef]
28. Bagui, S.; Li, K. Resampling imbalanced data for network intrusion detection datasets. *J. Big Data* **2021**, *8*, 6. [CrossRef]
29. Sikorski, M.; Honig, A. *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*; No Starch Press: San Francisco, CA, USA, 2012.
30. Erickson, J. *Hacking: The Art of Exploitation*; No Starch Press: San Francisco, CA, USA, 2008.
31. Bagui, S.S.; Mink, D.; Bagui, S.C.; Subramaniam, S.; Wallace, D. Resampling Imbalanced Network Intrusion Datasets to Identify Rare Attacks. *Future Internet* **2023**, *15*, 130. [CrossRef]
32. Han, J.; Kamber, M.; Pei, J. *Data Mining: Concepts and Techniques*; Morgan Kaufmann: Burlington, MA, USA, 2022.
33. Powders, D.M. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *J. Mach. Learn. Technol.* **2011**, *2*, 37–63.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.