future internet

MDPI

Article

Resampling Imbalanced Network Intrusion Datasets to Identify Rare Attacks

Sikha Bagui [1,*], Dustin Mink [1], Subhash Bagui [2], Sakthivel Subramaniam [1] and Daniel Wallace [1]

[1] Department of Computer Science, University of West Florida, Pensacola, FL 32514, USA; dmink@uwf.edu (D.M.); ss346@students.uwf.edu (S.S.); dw85@students.uwf.edu (D.W.)
[2] Department of Mathematics and Statistics, University of West Florida, Pensacola, FL 32514, USA; sbagui@uwf.edu
* Correspondence: bagui@uwf.edu

Abstract: This study, focusing on identifying rare attacks in imbalanced network intrusion datasets, explored the effect of using different ratios of oversampled to undersampled data for binary classification. Two designs were compared: random undersampling before splitting the training and testing data and random undersampling after splitting the training and testing data. This study also examines how oversampling/undersampling ratios affect random forest classification rates in datasets with minority dataor rare attacks. The results suggest that random undersampling before splitting gives better classification rates; however, random undersampling after oversampling with BSMOTE allows for the use of lower ratios of oversampled data.

Keywords: imbalanced data; resampling; rare attacks; network intrusion datasets; minority data; oversampling; BSMOTE; random undersampling; random forest

check for updates

Citation: Bagui, S.; Mink, D.; Bagui, S.; Subramaniam, S.; Wallace, D. Resampling Imbalanced Network Intrusion Datasets to Identify Rare Attacks. *Future Internet* **2023**, *15*, 130. https://doi.org/10.3390/fi15040130

Academic Editor: Cheng-Chi Lee

Received: 12 March 2023
Revised: 22 March 2023
Accepted: 27 March 2023
Published: 29 March 2023

1. Introduction

The internet generates traffic at a rate of 6.59 billion GB per second [1]. Approximately 1–3% of this traffic is malicious [2]. Advances in machine learning (ML) have allowed us to detect some of these attacks but not all. The challenge that cyber security experts face is not one of having enough data but one of having enough of the right type of data. Some organizations never see anomalous data on their network; most network traffic is generated by routine workplace business. Given this scenario, it is hard to know what to look for since we have such a small sample size of the attacks. Some types of attacks are more frequent than others, and it is only in the infrequent or rare attacks, or minority data, that this detection problem lies. Machine learning models, by their very nature, are good at detecting patterns where there is more data (in majority data); hence, detection of rare attacks (minority data) is challenging for machine learning models.

Minority data is a tiny percentage of network intrusion datasets, and for the purposes of this work, we are defining minority data as less than 0.1%. Various oversampling techniques, including different smote methods, random oversampling, ADASYN, and others, have been used by researchers to try to model minority data better. Oftentimes a combination of oversampling (of minority data) and undersampling (of majority data) is used [3].

When oversampling and undersampling are used, researchers have to determine how much undersampling should occur, how much oversampling should occur, and when the resampling should occur in the process. This research presents two different design methodologies, one performing random undersampling before splitting the training and testing data and the second performing random undersampling after splitting the training and testing data. In addition, for each of these two designs, the paper compares different ratios of random undersampling to oversampling using BSMOTE. Finally, binary classification using random forest is used to classify the various combinations of the data.

For this research, two datasets have been used—the first being a widely used network intrusion dataset, UNSW-NB15 [4], and the second being a newly created network intrusion dataset based on the MITRE ATT&CK framework, UWF-ZeekData22 [5].

The rest of this paper is organized as follows. Section 2 presents the background; Section 3 presents works related to oversampling and undersampling; Section 4 explains the datasets used in this study; Section 5 presents the experimental designs; Section 6 presents the hardware and software configurations; Section 7 presents the metrics used for assessment of the classification results; Section 8 presents the results and discussion; Section 9 presents the conclusions; and Section 10 presents the future works.

2. Background

To adequately address resampling, the resampling techniques used in this paper are briefly explained next. Though oversampling and undersampling are aimed at changing the ratios between the minority and majority classes, respectively, this is particularly important when the minority classes are minimal, that is, in the case of rare attacks.

A minority class refers to very low-occurring data categories. In this work, since a binary classification was performed, the dataset for each experimentation contained only two data types—one attack category and benign data. For example, for the analysis of worms, the dataset contained only worms and benign data. Out of these two categories of data, worms would be considered the minority class. Table 1, which presents the UNSW-NB15 dataset, shows that worms account for 0.006% of the data and 0.007% of the benign data. The raw data count for worms is 174 compared to the benign data count of 2,218,761.

Oversampling is the process of generating samples to increase the size of the minority class. Random oversampling involves randomly selecting samples from the minority class, with replacement, and adding them to the dataset [6]. Random oversampling can also lead to overfitting the data [3].

Various oversampling techniques are available, including various types of Synthetic Minority Oversampling Techniques (SMOTE) and ADASYN. SMOTE generates synthetic samples rather than resampling with replacement. These samples are generated along the line segment adjoining k-minority class neighbors [7]. In Borderline SMOTE (BSMOTE), a variant of the original SMOTE, borderline minority samples are identified and then synthetic samples are generated [8].

ADASYN [9] is a pseudo-probabilistic oversampling technique that uses a weighted distribution for different minority data points. This weighted distribution is based on the level of difficulty in learning, and more synthetic data are generated for minority class examples that are harder to learn than minority examples that are easier to learn [10].

Random undersampling samples the majority class by randomly picking samples with or without replacement from the majority class [11]. After random undersampling, the number of cases of the majority class decreases, significantly reducing the model's training time. However, the removed data points may include significant information leading to a decrease in classification results [3].

3. Related Works

For cybersecurity or network intrusion detection analysis, it is difficult to obtain a good workable dataset; among the few available, most are highly imbalanced due to the nature of the attacks. Certain attacks are rare in the real world, but measures must be taken to safeguard against them. The rarer the attack, the lesser the chance of it getting detected by a machine learning model.

Oversampling and undersampling are standard methods used by researchers to tackle class imbalance problems [3,7,12]. Oversampling generates synthetic samples from the existing minority class to balance the data or to have more instances for the classifier. Undersampling, on the other hand, reduces the majority class instances to bring the imbalance scale to normalcy. Though both these methods have inherent advantages and disadvantages, they can be used separately or combined with balancing ratios. It is possible

to selectively downsize only negative examples and keep all positive examples in the training set [13]. A major disadvantage of undersampling is the loss of data, and hence possibly critical information.

Oversampling increases the training time due to an increase in the training set [12], and may overfit the model [14]. Ref. [14] found that oversampling minority data before partitioning resulted in 40% to 50% AUC score improvement. When the minority oversampling is applied after the split, the actual AUC improvement is 4% to 10%. This behavior is due to what is termed as data leakage, caused by generating training samples correlated with the original data points that end up in the testing set [14]. Ref. [15] found that the synthetic instance creation approach plays a more significant role than the minority instance selection approach. The critical difference between these two papers is that the first process creates instances along the line connecting the two minority instances, while the second approach creates synthetic minority samples in the bounding rectangle created by joining the two minority instances.

Hence, both oversampling and undersampling can have potential overfitting or underfitting, respectively. Studies have been carried out with various combinations of oversampling and undersampling [12,16,17]. Random undersampling was also combined with each oversampling technique—SMOTE, ADASYN, Borderline, SVM-SMOTE, and random oversampling in [18]. Various percentages were used from each of these techniques to study the effect on minority class predictions. The results of each combination are highly dependent on the domain and distribution of the dataset [19]. In one domain, undersampling might help more than oversampling, but in another domain, it may be vice versa.

Some researchers have combined synthetic oversampling with other techniques, such as Grid Search (GS), to improve the prediction of attack traffic [20].

While the above papers present novel ways to examine the class imbalance classification problem, none directly addresses the problem of identifying rare attacks using different ratios of the sampling methods. The novelty of this research is that it considers different ratios of majority data to minority data when identifying rare attacks. Additionally, two different designs are compared—random undersampling before splitting the data and random undersampling after splitting the data, with different oversampling ratios.

4. The Datasets

Two datasets were used in this study: (i) a well-known network intrusion dataset, UNSW-NB15 [4] and (ii) a new network intrusion dataset created based on the MITRE ATT&CK framework, UWF-ZeekData22 [21].

4.1. UNSW-NB15

UNSW-NB15 [4], published in 2015, is a hybrid of real-world network data and simulated network attacks, comprising 49 features and 2.5 million rows. There are 2.2 million rows of normal or benign traffic, and the other 300,000 rows comprise nine different modern attack categories: Fuzzers, Reconnaissance, Shellcode, Analysis, Backdoors, DOS, Exploits, Worms, and Generic. Some attack categories, such as Worms, Shellcode, and Backdoors, that comprised only 0.006%, 0.059%, and 0.091%, of the total traffic, respectively, can be considered rare attacks. These are the three attack categories of particular interest in this research, and the rare attacks will be considered the minority classes. Table 1 presents the distribution of the attack families in this dataset, ordered from the smallest category to the largest category.

Table 1. UNSW-NB15: distribution of attack families [4].

Type of Attack	Count	% of Attack Data	% of Benign Data	% of Total Data
Worms	174	0.054	0.007	0.006
Shellcode	1511	0.47	0.068	0.059
Backdoors	2329	0.724	0.104	0.091
Analysis	2677	0.833	0.12	0.105
Reconnaissance	13,987	4.353	0.63	0.55
DoS	16,353	5.089	0.737	0.643
Fuzzers	24,246	7.546	1.092	0.954
Exploits	44,525	13.858	2.006	1.752
Generic	215,481	67.068	9.711	8.483
Total attack data	321,283	-	-	-
Benign data	2,218,761	-	-	87.351
Total	2,540,044	-	-	-

4.2. UWF-ZeekData22

UWF-ZeekData22 [5,21], published in 2022, is developed from data collected from Zeek, an open-source network-monitoring tool, and labeled based on the MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework. This dataset has approximately 9.280 million attack records and 9.281 benign records [22]. The breakdown of the data, that is, the percent of attack data for each attack type, is presented in Table 2, ordered from the smallest category to the largest category. For this analysis, two rare tactics were used: Privilege_escalation and Credential_access. Privilege_escalation and Credential_access form 0.00007% and 0.00016% of the total network traffic, respectively, and hence can be classified as rare attacks. These rare attacks will be treated as the minority classes.

Table 2. UWF-ZeekData22: distribution of MITRE ATT&CK tactics [5,21].

Label_Tactic	Count	% of Attack Data	% of Benign Data	% of Total Data
Persistence	1	0.00001	0.00001	0.000005
Initial_access	1	0.00001	0.00001	0.000005
Defense_evasion	1	0.00001	0.00001	0.000005
Resource_development	3	0.00003	0.00003	0.00001
Lateral_movement	4	0.00004	0.00004	0.00002
Exfiltration	7	0.00007	0.00007	0.00003
Privilege_escalation	13	0.00014	0.00014	0.00007
Credential_access	31	0.00033	0.00033	0.00016
Discovery	2086	0.02247	0.02247	0.01123
Reconnaissance	9,278,722	99.97686	99.969	49.98646
Total attack data	9,280,869	-	-	-
Benign_data	9,281,599	-	-	50.00196
Total	18,562,468	-	-	-

5. Experimental Design

The experimental design compares two approaches to solve the problem of having significantly small samples of minority data: (a) resampling or random undersampling before a stratified split (Figure 1a) and (b) resampling or random undersampling after a stratified split (Figure 1b). Since the objective of this work is to identify minority data or rare attacks, stratified sampling was used to guarantee that the training and testing data included minority data. However, the question is, when is it best to perform stratified sampling? In stratified sampling, the data are divided into groups and a certain percentage of samples are taken randomly from each group. This guarantees that there will be samples from the minority class in the data. For the purposes of this study, significantly small minority data are defined as <0.1 % of the total sample.

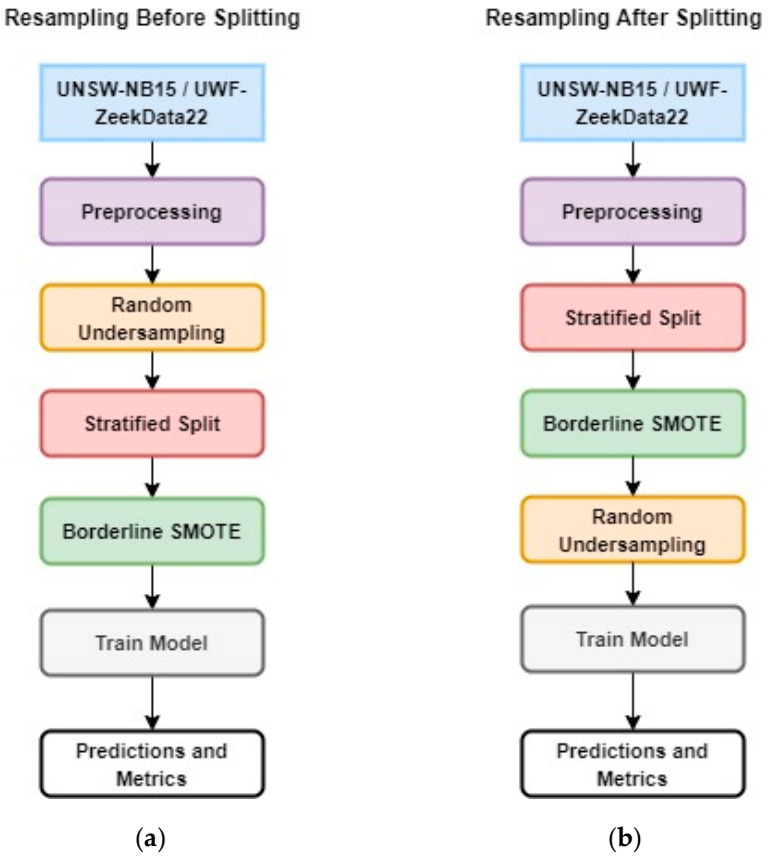

Figure 1. Experimental design: (**a**) Resampling Before Splitting; (**b**) Resampling After Splitting.

Basically, the two approaches differ in the preprocessing sequence prior to the machine learning model execution and have different approaches to creating training and testing data. In the first design, resampling before splitting, the data are preprocessed, and then random undersampling is performed. This is followed by a stratified split and oversampling using borderline SMOTE (B-SMOTE). Finally, these data were used for training and testing the machine learning model using random forest.

In the second design, resampling after splitting, after the initial preprocessing steps, the data are split (using stratified sampling). This method retains the stratified ratio of the majority to minority class since the stratified split was performed on the whole data. This is followed by oversampling using BSMOTE and random undersampling. Finally, these data were used to train and test the machine learning model using random forest.

5.1. The Classifier Used: Random Forest

Random forest (RF) is a highly used machine learning classifier that is basically an ensemble way of classifying records. In the RF algorithm, the decision of multiple decision trees taken together is used to come up with the final classification label [23].

The RF algorithm works by first generating each decision tree by randomly selecting a bootstrap sample. A bootstrap sample is a randomly selected sample from a dataset with replacement [24]. Each decision tree is trained with a separate bootstrap sample. Hence, if a random forest has N trees, N bootstrap samples will be required. Randomization is introduced during tree training. 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, such as information gain and Gini index [24]. Once forest training is complete, N-trained decision trees are created. A classification is made on one or more samples. RF classifies samples by querying each decision tree with the sample. A tally is kept to aggregate all classifications made by the decision trees [24]. Once all trees have voted with

a classification, the label with the most votes is chosen. Hence, RF is basically an ensemble technique based on decision trees.

5.2. Preprocessing

With regard to preprocessing, each dataset was handled differently, but information gain was used on both datasets to identify the relevant features. First, the information gain algorithm is explained, and then the preprocessing performed in each dataset is presented.

5.2.1. Information Gain

Information gain (IG) is used to assess the relative relevance of features in a dataset and is useful for classification. Information gain is calculated by removing the randomness in the dataset, which is measured by a class's entropy [23].

The following calculations were performed on each feature to produce information gain values for ranking purposes [23].

$$Gain(A) = Info(D) - Info_A(D) \tag{1}$$

where

$$Info(D) = -\sum_{i=1}^{m} p_i log_2(pi) \tag{2}$$

$$Info_A(D) = \sum_{j=1}^{V} \frac{|D_j|}{|D|} \times Info(D_j) \tag{3}$$

where:

- *Info(D)* is the average amount of information needed to identify the class level of a tuple in the data, *D*;
- $Info_A(D)$ is the expected information required to classify a tuple from *D* based on the partitioning by attribute *A*;
- p_i is the nonzero probability that an arbitrary tuple belongs to a class;
- $|Dj|/|D|$ is the weight of the *j*th partition;
- *V* is the number of distinct values in attribute A.

5.2.2. Preprocessing UNSW-NB15

For preprocessing UNSW-NB15, first, the following columns were dropped:

- ct_flw_http_mthd and is_ftp_login;
- Unique identifiers and time stamps;
- IP addresses.

Other preprocessing that was performed:

- The attack categories, NaN, were filled with zeros;
- Categorical data were turned into numeric representation: protocol, state, and attack category;
- A normalization technique was used on continuous data for all numeric variables:

$$X = \frac{x_i - \mu}{s} \tag{4}$$

where μ is the column mean and s is the column standard deviation.

Finally, IG was calculated on the remaining columns of this dataset, and columns with low information gain were not used. Columns dropped due to low information gain were service, dloss, stepd, dtcpb, res_bdy_len, trans_depth, and is_sm_ips_ports.

5.2.3. Preprocessing UWF-ZeekData22

For UWF-ZeekData22, the following preprocessing was performed following [22]:

- Continuous features, duration, orig_bytes, orig_pkts, orig_ip_bytes, resp_bytes, resp_pkts, resp_ip_bytes, and missed_bytes were binned using a moving mean;
- Nominal features, that is, features that contain non-numeric data, were converted to numbers using the *StringIndexer* method from MLib [25], Apache Spark's scalable machine learning library. The nominal features in this dataset were proto, conn_state, local_orig, history, and service;
- The IP address columns were categorized using the commonly recognized network classifications [26];
- Port numbers were binned as per the Internet Assigned Numbers Authority (IANA) [27].

Following the binning, IG was calculated on the binned dataset. Attributes with low IG were removed and not used for classification.

6. Hardware and Software Configurations

Tables 3 and 4 present the hardware and software configurations and python libraries, respectively, used in this research.

Table 3. Hardware and software configurations.

	Random Undersampling before Stratified Splitting	**Random Undersampling after Stratified Splitting**
Processor	AMD Ryzon 7 5700	Intel Core i7 1165G7
RAM	32 GB	16 GB
OS	Windows 11 Home	Windows 11 Home
OS Version	22 H2	21 H2
OS Build	22621.819	22000.1219
GPU	RTX 3060	NA

Table 4. Python library versions.

	Random Undersampling before Stratified Splitting	**Random Undersampling after Stratified Splitting**
Python	3.9	3.10.4
Anaconda	2022.1	2021.5
Pandas	1.5.2	1.5.0
Scikit-learn	1.9.3	1.0.2
Numpy	1.23.5	1.23.4
Imblearn	0.10.0	0

6.1. Hardware and Software Used in Random Undersampling before Stratified Splitting

Random undersampling before stratified splitting simulations was run on an x64-based processor with a 64-bit operating system. This Windows Home machine was running version 22H2 build 22621.819 and had an AMD Ryzen 7 5700 with 32 GB of RAM and an RTX 3060 Video card. CUDA for Python was installed; however, none of the trials used the parallel processing abilities of the machine.

6.2. Python Libraries Used in Random Undersampling before Stratified Splitting

Python version 3.9 was used with Jupyter Notebooks on Anaconda version 2022.10. Packages included pandas 1.5.2, scikit-learn 1.9.3, NumPy 1.23.5, and imblearn 0.10.0. The random forest machine learning algorithm was implemented using the scikit-learn RandomForestRegressor module. Borderline SMOTE was implemented using the BorderlineSMOTE module of the imblearn.over_sampling package.

6.3. Hardware and Software Used in Random Undersampling after Stratified Splitting

Random undersampling after stratified splitting simulations was run on a machine with Windows Home version 21H2 with OS build 22000.1219 run by an Intel Core i7 1165G7 processor and 16 GB RAM.

6.4. Python Libraries Used in Random Undersampling after Stratified Splitting

Python version 3.10.4 was used with Jupyter Notebooks on Anaconda version 2021.05. Packages included pandas 1.5.0, scikit-learn 1.0.2, NumPy 1.23.4, and imblearn 0.0.

6.5. Stratified Sampling

Scikit-learn in Python was used to generate the training and testing stratified splits.

7. Metrics Used for the Assessment of Results

The objective of this research is to show how a combination of undersampling and oversampling techniques helps build a model which classifies minority classes more accurately. Two approaches were taken for preparing the data: random undersampling before splitting and random undersampling after splitting, and different proportions of oversampling were used (from 0.1 to 1.0, incremented at intervals of 0.1), while the undersampling was maintained at a constant of 0.5 (50%).

7.1. Classification Metrics Used

For evaluating the classification of the random forest runs, the following matrices were used: accuracy, precision, recall, F-Score, and macro precision.

Accuracy: *Accuracy* is the number of correctly classified instances (i.e., True Positives and True Negatives) divided by the total number of instances [28].

$$\textit{Accuracy} = [\text{True Positives (TP)} + \text{True Negatives (TN)}]/[\text{TP} + \text{False Positives} + \text{TN} + \text{False Negatives}] \quad (5)$$

Precision: *Precision* is the proportion of predicted positive cases that are correctly labeled as positive [29]. Precision by label considers only one class and measures the number of times a specific label was predicted correctly, normalized by the number of times that label appears in the output.

$$\textit{Precision} = \text{Positive Predictive Value} = [\text{True Positives}]/[\text{True Positives} + \text{False Positives}] \quad (6)$$

Recall: *Recall* is the ability of the classifier to find all the positive samples, or the True Positive Rate (TPR). Recall is also known as sensitivity, and is the proportion of Real Positive cases that are correctly predicted as Positive (TP) [29].

All Real Positives = [True Positives + False Negatives]
All Real Negatives = [True Negatives + False Positives]

$$\textit{Recall} = \text{True Positive Rate} = [\text{True Positives}]/[\text{All Real Positives}] \quad (7)$$

F-Score: *The F-score* is the harmonic mean of a prediction's precision and recall metrics. It is another overall measure of the test's accuracy [29].

$$\textit{F-Score} = 2 * [\text{Precision} * \text{Recall}]/[\text{Precision} + \text{Recall}] \quad (8)$$

Recall, sensitivity, and TPR connate the same measure.

Macro Precision: Macro precision finds the unweighted mean of the precision values. This does not take label imbalance into account [30].

7.2. Welch's t-Tests

Welch's *t*-tests were used to find the differences in the means of the different oversampling percentages. When comparing metrics across two successive percentage runs, the increase or decrease in the metric being evaluated has to be determined, and hence a one-tailed Welch's *t*-test was used. Welch's *t*-test is calculated using the formula:

$$(\bar{x}_1 - \bar{x}_2) / \left(\sqrt{s_1^2/n_1 + s_2^2/n_2} \right) \quad (9)$$

where $\bar{x}_1$ and $\bar{x}_2$ are sample means of the metrics, ${s_1}^2$ and ${s_2}^2$ are sample variances, n_1 and n_2 are sample sizes, and the df v is calculated using Satterwaite approximation.

The mean of the individual runs for each metric for one oversampling percentage is compared with another, for example, 0.1 vs. 0.2. If the *t*-score value is high, there is more difference between the two means. In order to determine whether this difference is significant, the *p*-value was calculated for each *t*-score. The significance level was kept at 0.1, making the test more sensitive to results and increasing the significance zone. If the *p*-value is less than the threshold, then the increase or decrease in the *t*-score value is significant between the two means.

8. Results and Discussion

This section presents the statistical results, followed by a discussion. Since several oversampling techniques are available, the first step was to perform a study to select an oversampling technique. Then, the results of the random undersampling before stratified splitting are presented, followed by the results of the random undersampling after stratified splitting.

8.1. Selection of an Oversampling Technique

An initial analysis was conducted to determine the best synthetic oversampling technique among a few commonly used oversampling techniques, SMOTE, Borderline SMOTE, and ADASYN.

Traditionally, a 70-30 training-testing split is performed on the data. However, in this case, since the occurrences of the minority class are minimal, there is no guarantee that samples from the minority class will be present in both the training data and the testing data without sample stratification by category. Hence, stratified sampling was used to split the data into a 70-30 training-testing ratio.

From the evaluation metrics shown in Figure 2, it is evident that BSMOTE performs better than the other resampling techniques in terms of precision, F-score, and macro precision. SMOTE and ADASYN had better recall than BSMOTE, but the latter performed better overall.

Hence, all future analyses in this work use BSMOTE as the primary oversampling technique, combined with random undersampling. Several researchers have confirmed that neither oversampling nor undersampling alone can successfully classify imbalanced datasets [12,19], let alone identify rare attacks. Hence, this study looks at the effects of varying oversampling percentages.

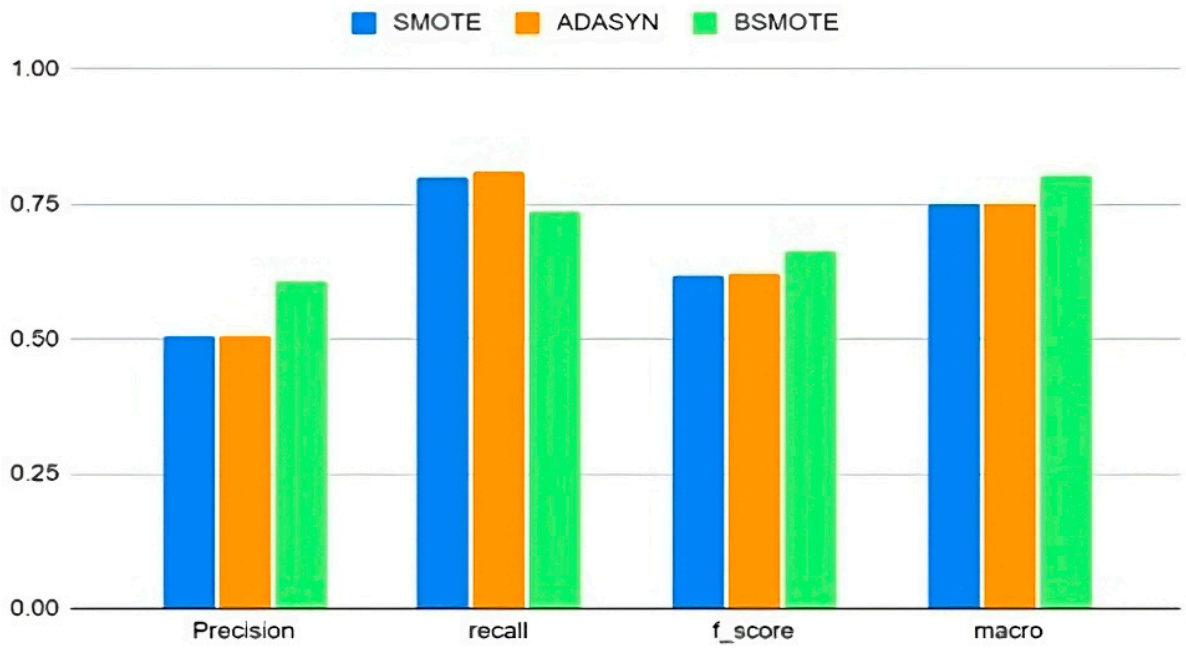

Figure 2. Comparison of oversampling techniques.

8.2. Resampling before and after Splitting

For random undersampling before stratified splitting, as well as random undersampling after stratified splitting, the percent of undersampling was kept at 0.5 (50%) and the percent of oversampling varied from 0.1 (10%) of the data to 1.0 (100%) of the data, at increments of 0.1 (10%). That is, 50% of the instances were selected at random from the majority class, and from the minority class, first 0.1 (10%) oversampling was used, then 0.2 (20%), then 0.3 (30%), and so on, until 1.0 (100%).

For UNSW-NB15, the three most minor attack categories, Worms, Shellcode, and Backdoors, were used. Worms, Shellcode, and Backdoors comprise 0.006%, 0.059%, and 0.091% of the total data, respectively. For UWF-ZeekData22, two tactics were used: credential access and privilege escalation. Privilege escalation and credential access were contained in 0.00007 and 0.00016% of the total data respectively.

For each of the attack categories, for random undersampling before stratified splitting, as well as random undersampling after stratified splitting, an average of ten runs were performed. Values are presented for accuracy, precision, recall, F-score, and macro precision for random undersampling of the majority data at 0.5 (50%), and oversampling the minority data using BSMOTE varied from 0.1 (10%) to 1.0 (100%), at increments of 0.1 (10%). The standard deviations (SDs) are also presented. Since only binary classification is performed in this study, the majority of the data were the benign or non-attack data and the minority of the data were the respective attack category, for example, worms or credential access.

8.2.1. Random Undersampling before Stratified Splitting

Table 5 presents the classification results for Random Undersampling Before Stratified Splitting for Worms (UNSW-NB15) for the various oversampling percentages (0.1 to 1.0, at intervals of 0.1). The best results are highlighted in green.

To determine the best results among the various oversampling percentages, Welch's *t*-tests were performed. Results of Welch's *t*-tests for UNSW-NB15, Worms Random Sampling before Stratified Splitting (Table 5), are presented in Table 6. The results presented in Table 6 show the *t*-test comparisons between the results of various oversampling runs. In the first row of comparison between 0.1 and 0.2, we observed that the *p*-values across all metrics are above 0.1, the significance level. Hence, there is no statistical difference between the compared results. However, in the second row of comparison between 0.1 and 0.3, the

recall and F-score had significant differences, and both the *t*-score values were positive. This implies that 10% oversampling performed better than 30%.

Table 5. UNSW-NB15: Worms—classification results for random undersampling before splitting.

Oversampling %		Accuracy	Precision	Recall	F-Score	Macro Precision
0.1	Avg	0.9999	0.769	0.794	0.778	0.884
	SD		0.056	0.063	0.034	0.028
0.2	Avg	0.99991	0.806	0.779	0.789	0.902
	SD		0.065	0.09	0.058	0.032
0.3	Avg	0.9999	0.799	0.717	0.752	0.899
	SD		0.059	0.075	0.047	0.029
0.4	Avg	0.99991	0.802	0.783	0.787	0.901
	SD		0.068	0.076	0.043	0.034
0.5	Avg	0.999	0.836	0.871	0.852	0.918
	SD		0.051	0.037	0.033	0.026
0.6	Avg	0.999	0.828	0.851	0.838	0.914
	SD		0.057	0.057	0.042	0.029
0.7	Avg	0.999	0.849	0.88	0.862	0.924
	SD		0.053	0.051	0.026	0.027
0.8	Avg	0.999	0.847	0.851	0.847	0.924
	SD		0.056	0.051	0.039	0.028
0.9	Avg	0.999	0.807	0.892	0.845	0.903
	SD		0.056	0.043	0.034	0.028
1.0	Avg	0.999	0.818	0.75	0.782	0.909
	SD		0.069	0.051	0.055	0.034
	Averages		0.8161	0.8168	0.8132	0.9078

Table 6. Welch's *t*-test results: UNSW-NB15: Worms—random undersampling before splitting.

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	−1.366	0.441	−0.451	−1.366	0.1 and 0.2 are statistically equal
0.1 vs. 0.3	−1.167	2.489	1.423	−1.167	0.1 is better
0.1 vs. 0.4	−1.179	0.368	−0.516	−1.180	0.1 and 0.4 are statistically equal
0.1 vs. 0.5	−2.793	−3.315	−4.955	−2.795	0.5 is better than 0.1
0.5 vs. 0.6	0.326	0.949	0.863	0.326	0.5 and 0.6 are statistically equal
0.5 vs. 0.7	−0.551	−0.423	−0.722	−0.551	0.5 and 0.7 are statistically equal
0.5 vs. 0.8	−0.467	1.022	0.291	−0.466	0.5 and 0.8 are statistically equal
0.5 vs. 0.9	1.213	−1.130	0.442	1.213	0.5 and 0.9 are statistically equal
1 vs. 0.5	0.649	6.075	3.451	0.651	0.5 is better than 1.0

Hence, based on the analysis in Table 6, the best results were obtained at 0.5 oversampling for Worms (highlighted in green in Table 5). There are, however, sampling ratios that are statistically equivalent to using 0.5, as shown in Table 6. However, an oversampling of 0.5 was chosen as the best since it has the smallest amount of oversampled data and thus would take the least computational time.

Table 7 presents the classification results for Random Undersampling Before Splitting for Shellcode (UNSW-NB15) for the various oversampling percentages (0.1 to 1.0, at intervals of 0.1). The best results are highlighted in green.

Table 7. UNSW-NB15: Shellcode—classification results for random undersampling before splitting.

Oversampling %		Accuracy	Precision	Recall	F-Score	Macro Precision
0.1	Avg	0.9996	0.849	0.941	0.892	0.924
	SD		0.013	0.014	0.012	0.006
0.2	Avg	0.9997	0.889	0.969	0.927	0.944
	SD		0.015	0.01	0.007	0.007
0.3	Avg	0.9997	0.896	0.962	0.927	0.948
	SD		0.013	0.012	0.007	0.006
0.4	Avg	0.9997	0.898	0.958	0.927	0.949
	SD		0.008	0.008	0.007	0.004
0.5	Avg	0.9996	0.887	0.964	0.924	0.944
	SD		0.014	0.007	0.007	0.007
0.6	Avg	0.9996	0.876	0.964	0.918	0.938
	SD		0.012	0.014	0.008	0.006
0.7	Avg	0.9995	0.844	0.925	0.883	0.922
	SD		0.012	0.013	0.009	0.006
0.8	Avg	0.9995	0.846	0.932	0.887	0.923
	SD		0.012	0.013	0.005	0.006
0.9	Avg	0.9995	0.849	0.929	0.887	0.924
	SD		0.013	0.01	0.007	0.006
1	Avg	0.9996	0.855	0.939	0.895	0.928
	SD		0.014	0.014	0.008	0.007
	Averages	0.9996	0.8689	0.9483	0.9067	0.9344

Results of Welch's *t*-tests for UNSW-NB15 Shellcode Random Undersampling before Splitting (Table 7) are presented in Table 8. Based on the analysis in Table 8, the best results were obtained at 0.4 oversampling for Shellcode (highlighted in green in Table 7).

Table 8. Welch's *t*-test results: UNSW-NB15: Shellcode—random undersampling before splitting.

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	−6.530	−5.250	−8.049	−6.537	0.2 is better than 0.1
0.2 vs. 0.3	−1.093	1.438	−0.107	−1.092	0.2 better than 0.3
0.2 vs. 0.4	−1.613	2.616	−0.005	−1.609	0.2 has better recall
0.4 vs. 0.5	2.015	−1.784	0.888	2.012	0.5 has better recall
0.4 vs. 0.6	4.557	−1.176	2.507	4.553	0.4 is better than 0.6
04. vs. 0.7	11.157	6.763	11.799	11.163	0.4 is better than 0.7
0.4 vs. 0.8	11.097	5.620	13.965	11.110	0.4 is better than 0.8
0.4 vs. 0.9	10.104	7.201	12.712	10.113	0.4 is better than 0.9
0.4 vs. 1	8.290	3.908	9.362	8.296	0.4 is better than 1.0

Table 9 presents the classification results for Random Undersampling Before Splitting for Backdoors (UNSW-NB15) for the various oversampling percentages (0.1 to 1.0, at intervals of 0.1). The best results are highlighted in green.

Table 9. UNSW-NB15: Backdoors—classification results for random undersampling before splitting.

Oversampling %		Accuracy	Precision	Recall	F-Score	Macro Precision
0.1	Avg	0.9998	0.962	0.96	0.961	0.981
	SD		0.006	0.006	0.003	0.003
0.2	Avg	0.9998	0.969	0.964	0.966	0.984
	SD		0.009	0.007	0.006	0.004
0.3	Avg	0.9997	0.962	0.95	0.956	0.981
	SD		0.005	0.01	0.005	0.003

Table 9. *Cont.*

Oversampling %		Accuracy	Precision	Recall	F-Score	Macro Precision
0.4	Avg	0.9998	0.97	0.956	0.963	0.985
	SD		0.007	0.009	0.005	0.003
0.5	Avg	0.9998	0.97	0.951	0.961	0.985
	SD		0.007	0.008	0.005	0.003
0.6	Avg	0.9998	0.965	0.954	0.96	0.982
	SD		0.007	0.009	0.005	0.003
0.7	Avg	0.9997	0.968	0.946	0.957	0.984
	SD		0.006	0.011	0.006	0.003
0.8	Avg	0.9998	0.966	0.95	0.958	0.983
	SD		0.005	0.01	0.006	0.002
0.9	Avg	0.9998	0.967	0.957	0.962	0.983
	SD		0.008	0.006	0.004	0.004
1	Avg	0.9998	0.968	0.952	0.96	0.984
	SD		0.009	0.01	0.008	0.004
	Averages	0.99978	0.9667	0.954	0.9604	0.9832

Results of Welch's *t*-tests for UNSW-NB15 Backdoors Random Undersampling before Splitting (Table 9) are presented in Table 10. Based on the analysis in Table 10, the best results were obtained at 0.2 oversampling for Backdoors (highlighted in green in Table 9).

Table 10. Welch's *t*-test results: UNSW-NB15: Backdoors—random undersampling before splitting.

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	−1.813	−1.602	−2.697	−1.818	0.2 is better than 0.1 across all metrics
0.2 vs. 0.3	1.955	3.627	4.492	1.970	0.2 is better than 0.1 across all metrics
0.2 vs. 0.4	−0.404	2.338	1.491	−0.397	0.2 is better than 0.4 in recall and F-score
0.2 vs. 0.5	−0.474	3.866	2.359	−0.464	0.2 is better than 0.5 in recall and F-score
0.2 vs. 0.6	1.070	2.621	2.861	1.079	0.2 is better than 0.6 in recall and F-score
0.2 vs. 0.7	0.189	4.415	3.563	0.206	0.2 is better than 0.7 in recall and F-score
0.2 vs. 0.8	0.829	3.661	3.254	0.842	0.2 is better than 0.8 in recall and F-score
0.2 vs. 0.9	0.524	2.378	2.136	0.531	0.2 is better than 0.9 in recall and F-score
0.2 vs. 1.0	0.180	3.043	2.149	0.189	0.2 is better than 1.0 in recall and F-score

Table 11 presents the classification results for Random Undersampling Before Splitting for Credential Access (UWF-ZeekData22) for the various oversampling percentages (0.1 to 1.0, at intervals of 0.1). The best results are highlighted in green.

Table 11. UWF-ZeekData22: credential access—classification results for random undersampling before splitting.

Oversampling %		Accuracy	Precision	Recall	F-Score	Macro Precision
0.1	Avg	0.999	0.742	0.889	0.801	0.871
	SD		0.134	0.165	0.126	0.067
0.2	Avg	0.999	0.885	0.911	0.891	0.942
	SD		0.05	0.139	0.075	0.024
0.3	Avg	0.999	0.847	0.867	0.849	0.923
	SD		0.101	0.147	0.107	0.05
0.4	Avg	0.999	0.836	0.856	0.83	0.918
	SD		0.106	0.165	0.107	0.053
0.5	Avg	0.999	0.936	0.911	0.919	0.968
	SD		0.069	0.109	0.072	0.034
0.6	Avg	0.999	0.906	0.867	0.87	0.953
	SD		0.103	0.163	0.094	0.052
0.7	Avg	0.999	0.882	0.922	0.894	0.941
	SD		0.067	0.122	0.06	0.033
0.8	Avg	0.999	0.826	0.911	0.853	0.913
	SD		0.143	0.097	0.084	0.071
0.9	Avg	0.999	0.829	0.967	0.889	0.915
	SD		0.079	0.071	0.054	0.04
1	Avg	0.999998	0.832	0.911	0.864	0.916
	SD		0.109	0.097	0.076	0.054
	Averages	0.9991	0.8521	0.9012	0.866	0.926

Results of Welch's *t*-tests for UWF-ZeekData22 Credential Access Random Undersampling before Splitting (Table 11) are presented in Table 12. Based on the analysis in Table 12, the best results were obtained at 0.5 oversampling for credential access (highlighted in green in Table 11).

Table 12. Welch's *t*-test results: UWF-ZeekData22: credential access—random undersampling before splitting.

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	−3.162	−0.322	−1.941	−3.155	0.2 is better than 0.1 in precision, F-score, and macro precision
0.2 vs. 0.3	1.066	0.688	1.016	1.083	0.2 and 0.3 are statistically the same
0.2 vs. 0.4	1.322	0.806	1.476	1.304	0.2 is better than 0.4 in F-score
0.2 vs. 0.5	−1.893	0.000	−0.852	−1.976	0.5 is better than 0.2 in precision and macro precision
0.5 vs. 0.6	0.765	0.710	1.309	0.763	0.5 and 0.6 are statistically equal
0.5 vs. 0.7	0.113	−0.188	−0.099	0.077	0.5 and 0.7 are statistically equal
0.5 vs. 0.8	2.191	0.000	1.886	2.209	0.5 is better than 0.8 in precision, F-score, and macro precision
0.5 vs. 0.9	3.226	−1.361	1.054	3.193	0.5 is better than 0.9 in precision and macro precision
0.5 vs. 1.0	1.398	0.000	0.800	1.391	0.5 is better than 1.0 in precision and macro precision

Table 13 presents the classification results for Random Undersampling Before Splitting for Privilege Escalation (UWF-ZeekData22) for the various oversampling percentages (0.1 to 1.0, at intervals of 0.1). The best results are highlighted in green.

Table 13. UWF-ZeekData22: privilege escalation—classification results for random undersampling before splitting.

Oversampling %		Accuracy	Precision	Recall	F-Score	Macro Precision
0.1	Avg	0.9999	0.902	0.775	0.81	0.951
	SD		0.125	0.236	0.156	0.063
0.2	Avg	0.999996	0.904	0.8	0.828	0.952
	SD		0.138	0.232	0.165	0.069
0.3	Avg	0.999996	0.895	0.825	0.841	0.947
	SD		0.15	0.243	0.185	0.075
0.4	Avg	0.999996	0.9	0.794	0.824	0.95
	SD		0.147	0.249	0.185	0.074
0.5	Avg	0.999996	0.908	0.815	0.839	0.954
	SD		0.139	0.244	0.177	0.069
0.6	Avg	0.999996	0.907	0.846	0.857	0.954
	SD		0.136	0.233	0.169	0.068
0.7	Avg	0.999996	0.905	0.843	0.853	0.952
	SD		0.139	0.24	0.175	0.07
0.8	Avg	0.999996	0.906	0.834	0.848	0.953
	SD		0.139	0.247	0.182	0.069
0.9	Avg	0.999996	0.899	0.844	0.851	0.95
	SD		0.14	0.24	0.178	0.07
1	Avg	0.999996	0.899	0.848	0.852	0.949
	SD		0.14	0.242	0.181	0.07
	Averages	0.9999959	0.9025	0.8224	0.8403	0.9512

Results of Welch's *t*-tests for UWF-ZeekData22 Privilege Escalation Random Undersampling before Splitting (Table 13) are presented in Table 14. Based on the analysis in Table 14, the best results were obtained at 0.1 oversampling for privilege escalation (highlighted in green in Table 13). Though there are sampling ratios that are statistically equivalent to 0.1 (as shown in Table 14), 0.1 was chosen as the best since 0.1 has the smallest amount of oversampled data, thus taking the least computational time.

Table 14. Welch's *t*-test results: UWF-ZeekData22: privilege escalation—random undersampling before splitting.

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	−0.042	−0.239	−0.252	−0.042	0.1 and 0.2 are statistically equal
0.1 vs. 0.3	0.108	−0.467	−0.411	0.108	0.1 and 0.3 are statistically equal
0.1 vs. 0.4	0.034	−0.173	−0.178	0.034	0.1 and 0.4 are statistically equal
0.1 vs. 0.5	−0.102	−0.373	−0.387	−0.102	0.1 and 0.5 are statistically equal
0.1 vs. 0.6	−0.100	−0.675	−0.644	−0.100	0.1 and 0.5 are statistically equal
0.1 vs. 0.7	−0.056	−0.638	−0.587	−0.056	0.1 and 0.7 are statistically equal
0.1 vs. 0.8	−0.074	−0.550	−0.502	−0.074	0.1 and 0.8 are statistically equal

Table 14. *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. 0.9	0.037	−0.652	−0.555	0.037	0.1 and 0.9 are statistically equal
0.2 vs. 1.0	0.048	−0.678	−0.556	0.048	0.1 and 1.0 are statistically equal

After analyzing the classification results using Welch's *t*-tests, for Random Undersampling Before Stratified Splitting, it appears that a random undersampling of 0.5 of the majority data before stratified splitting gives the best results when the BSMOTE oversampling is also at 0.5 for two datasets: Worms and credential access. The best results for Shellcode were achieved at a random undersampling at 0.5 and a BSMOTE oversampling of 0.4, and at a BSMOTE oversampling of 0.2 and 0.1 for Backdoors and privilege escalation, respectively.

8.2.2. Random Undersampling after Stratified Splitting

Table 15 presents the classification results for Random Undersampling After Splitting for Worms (UNSW-NB15) for the various oversampling percentages (0.1 to 1.0, at intervals of 0.1). The best results are highlighted in green.

Table 15. UNSW-NB15: Worms—classification results for random undersampling after splitting.

Oversampling %		Accuracy	Precision	Recall	F-Score	Macro Precision
0.1	Avg	0.999	0.608	0.737	0.665	0.804
	SD	NA	0.067	0.044	0.0495	0.034
0.2	Avg	0.999	0.601	0.712	0.646	0.8
	SD	NA	0.108	0.089	0.083	0.054
0.3	Avg	0.999	0.566	0.773	0.651	0.783
	SD	NA	0.066	0.059	0.051	0.033
0.4	Avg	0.999	0.566	0.781	0.654	0.783
	SD	NA	0.035	0.063	0.026	0.018
0.5	Avg	0.999	0.581	0.738	0.65	0.791
	SD	NA	0.078	0.082	0.079	0.039
0.6	Avg	0.999	0.587	0.76	0.656	0.793
	SD	NA	0.097	0.044	0.061	0.049
0.7	Avg	0.999	0.62	0.753	0.679	0.81
	SD	NA	0.053	0.046	0.041	0.026
0.8	Avg	0.999	0.54	0.719	0.614	0.77
	SD	NA	0.081	0.036	0.041	0.018
0.9	Avg	0.999	0.573	0.711	0.629	0.787
	SD	NA	0.117	0.097	0.089	0.058
1	Avg	0.999	0.601	0.75	0.666	0.801
	SD	NA	0.062	0.081	0.06	0.031
	Averages	0.999	0.5843	0.7434	0.651	0.7922

Results of Welch's *t*-tests for UNSW-NB15, Worms for Random Undersampling after Splitting (Table 15), are presented in Table 16. Consider the *t*-test comparison between 0.1 vs. 0.3 in Table 16. It was found that the *t*-values of precision, recall, and macro precision have statistical significance. Since precision and macro precision have positive *t*-values, it implies that 0.1 oversampling performed better in these two metrics. However, the recall value was negative, which means that 0.3 had better true positive predictions than 0.1. If the *p*-value is higher than 0.1, then there is statistically no significant difference between the two means. This is the case in the comparison of 0.1 vs. 0.2 oversampling in Table 16.

Table 16. Welch's *t*-test results: UNSW-NB15: Worms—random undersampling after splitting.

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	No significant difference
0.1 vs. 0.3	1.413	−1.563	0.592	1.413	0.1 is better than 0.3 in precision and macro precision, but 0.3 is better in recall
0.1 vs. 0.4	1.773	−1.826	0.628	1.773	0.1 is better than 0.4 in precision and macro precision, but 0.3 is better in recall
0.1 vs. 0.5	0.836	−0.065	0.506	0.836	0.1 and 0.5 are statistically equal
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	2.5	2.826	0.1 better than 0.8 except for 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 statically equal
0.1 vs. 1	0.219	−0.746	−0.033	0.302	0.1 and 1 are statically equal

Based on the analysis in Table 16, the best results were obtained at 0.1 oversampling for Worms (highlighted in green in Table 15). Though there are sampling ratios that are statistically equivalent to 0.1 (as shown in Table 16), 0.1 was chosen as the best since it has the smallest amount of oversampled data, thus taking the least computational time.

Table 17 presents the classification results for Random Undersampling After Splitting for Shellcode (UNSW-NB15) for the various oversampling percentages (0.1 to 1.0, at intervals of 0.1). The best results are highlighted in green.

Table 17. UNSW-NB15: Shellcode—classification results for random undersampling after splitting.

Oversampling %		Accuracy	Precision	Recall	F-Score	Macro Precision
0.1	Avg	0.999	0.698	0.906	0.788	0.849
	SD	NA	0.016	0.017	0.010	0.008
0.2	Avg	0.999	0.694	0.907	0.786	0.847
	SD	NA	0.016	0.015	0.011	0.008
0.3	Avg	0.999	0.691	0.911	0.786	0.846
	SD	NA	0.017	0.015	0.011	0.008
0.4	Avg	0.999	0.686	0.905	0.780	0.843
	SD	NA	0.014	0.011	0.010	0.007
0.5	Avg	0.999	0.678	0.906	0.776	0.839
	SD	NA	0.011	0.014	0.008	0.005
0.6	Avg	0.999	0.699	0.908	0.790	0.849
	SD	NA	0.020	0.003	0.013	0.010
0.7	Avg	0.999	0.699	0.908	0.790	0.849
	SD	NA	0.021	0.016	0.016	0.011
0.8	Avg	0.999	0.692	0.911	0.787	0.846
	SD	NA	0.011	0.021	0.013	0.006

Table 17. *Cont.*

Oversampling %		Accuracy	Precision	Recall	F-Score	Macro Precision
0.9	Avg	0.999	0.688	0.901	0.780	0.844
	SD	NA	0.021	0.010	0.015	0.011
1.0	Avg	0.999	0.688	0.888	0.775	0.844
	SD	NA	0.018	0.015	0.016	0.009
	Averages	0.999	0.6913	0.9051	0.7838	0.8456

Results of Welch's *t*-tests for Random Undersampling after Splitting for Shellcode (UNSW-NB15) (Table 17) are presented in Table 18. Based on the analysis in Table 18, the best results were obtained at 0.1 oversampling for Shellcode (highlighted in green in Table 17). Though there are sampling ratios that are statistically equivalent to 0.1 (as shown in Table 18), 0.1 was chosen as the best since it has the smallest amount of oversampled data, thus taking the least computational time.

Table 18. Welch's *t*-test results: UNSW-NB15: Shellcode—random undersampling after splitting.

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	0.566	−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	0.07	1.702	1.806	0.1 better than 0.4 in precision, F-score, and macro precision
0.1 vs. 0.5	3.349	−0.128	3.104	3.349	0.1 better than 0.5 in precision, F-score, and macro precision
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	−0.73	0.293	0.683	0.1 and 0.7 are statistically equal
0.1 vs. 0.8	1.62	0.52	1.48	1.62	0.1 better than 0.8 in 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 in recall and F-score
0.1 vs. 1	2.101	0.246	1.832	2.101	0.1 is better than 1 in all metrics except recall

Table 19 presents the classification results for Random Undersampling After Splitting for Backdoors (UNSW-NB15) for the various oversampling percentages (0.1 to 1.0, at intervals of 0.1). The best results are highlighted in green.

Results of Welch's *t*-tests for UNSW-NB15 Backdoors for Random Undersampling after Splitting (Table 19) are presented in Table 20. Based on the analysis in Table 20, the best results were obtained at 0.1 oversampling for Backdoors (highlighted in green in Table 19). Again, although there are sampling ratios that are statistically equivalent to 0.1 (as shown in Table 20), 0.1 was chosen as the best since it has the smallest amount of oversampled data, thus taking the least computational time.

Table 19. UNSW-NB15: Backdoors—classification results for random undersampling after splitting.

Oversampling %		Accuracy	Precision	Recall	F-Score	Macro Precision
0.1	Avg	0.999	0.939	0.951	0.945	0.969
	SD	NA	0.01	0.005	0.004	0.005
0.2	Avg	0.999	0.938	0.948	0.943	0.969
	SD	NA	0.013	0.008	0.006	0.006
0.3	Avg	0.999	0.936	0.945	0.941	0.968
	SD	NA	0.009	0.011	0.007	0.004
0.4	Avg	0.999	0.939	0.95	0.944	0.969
	SD	NA	0.01	0.008	0.007	0.005
0.5	Avg	0.999	0.938	0.948	0.943	0.969
	SD	NA	0.01	0.007	0.006	0.005
0.6	Avg	0.999	0.941	0.945	0.943	0.97
	SD	NA	0.006	0.011	0.005	0.003
0.7	Avg	0.999	0.946	0.948	0.947	0.973
	SD	NA	0.007	0.003	0.003	0.003
0.8	Avg	0.999	0.946	0.943	0.944	0.973
	SD	NA	0.012	0.009	0.01	0.006
0.9	Avg	0.999	0.94	0.949	0.944	0.97
	SD	NA	0.007	0.011	0.003	0.003
1	Avg	0.999	0.944	0.943	0.943	0.972
	SD	NA	0.012	0.007	0.005	0.006
	Averages	0.999	0.9407	0.947	0.9437	0.9702

Table 20. Welch's *t*-test results: UNSW-NB15: Backdoors—random undersampling after splitting.

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	0.013	0.812	0.555	0.013	0.1 and 0.2 are statistically equal
0.1 vs. 0.3	0.493	1.461	1.579	0.495	0.1 better than 0.3 in recall and F-score
0.1 vs. 0.4	−0.036	0.225	0.097	−0.036	0.1 and 0.4 statistically equal
0.1 vs. 0.5	0.102	0.978	0.68	0.103	0.1 and 0.5 statistically equal
0.1 vs. 0.6	−0.676	1.296	0.657	−0.675	0.1 and 0.6 statistically equal
0.1 vs. 0.7	−1.738	1.318	−1.316	−1.738	0.1 and 0.7 statistically equal
0.1 vs. 0.8	−1.389	2.264	0.063	−1.387	0.8 is better than 0.1 in precision and F-score while recall is better in 0.1
0.1 vs. 0.9	1.336	−1.37	−0.011	1.334	0.8 and 0.9 are statistically equal but 0.9 has better recall
0.1 vs. 1	0.353	0	0.282	0.353	0.8 and 1 are statistically equal

Table 21 presents the classification results for Random Undersampling After Splitting for Credential Access (UWF-ZeekData22) for the various oversampling percentages (0.1 to 1.0, at intervals of 0.1). The best results are highlighted in green.

Table 21. UWF-ZeekData22: credential access—classification results for random undersampling after splitting.

Oversampling %		Accuracy	Precision	Recall	F-Score	Macro Precision
0.1	Avg	0.999	0.822	0.878	0.833	0.911
	SD	NA	0.176	0.116	0.112	0.088
0.2	Avg	0.999	0.732	0.900	0.788	0.867
	SD	NA	0.141	0.168	0.123	0.071
0.3	Avg	0.999	0.799	0.911	0.847	0.899
	SD	NA	0.112	0.120	0.101	0.056
0.4	Avg	0.999	0.744	0.911	0.804	0.872
	SD	NA	0.162	0.097	0.100	0.081
0.5	Avg	0.999	0.770	0.944	0.835	0.885
	SD	NA	0.154	0.075	0.085	0.077
0.6	Avg	0.999	0.696	0.944	0.793	0.848
	SD	NA	0.116	0.102	0.092	0.056
0.7	Avg	0.999	0.713	0.922	0.793	0.857
	SD	NA	0.155	0.122	0.116	0.0777
0.8	Avg	0.999	0.639	0.933	0.749	0.820
	SD	NA	0.067	0.133	0.058	0.034
0.9	Avg	0.999	0.722	0.922	0.800	0.861
	SD	NA	0.110	0.100	0.052	0.055
1.0	Avg	0.999	0.742	0.889	0.789	0.871
	SD	NA	0.146	0.131	0.083	0.073
	Averages	0.999	0.7379	0.9154	0.8031	0.8691

Results of Welch's *t*-tests for UWF-ZeekData22 Credential Access for Random Undersampling after Splitting (Table 21) are presented in Table 22. Based on the analysis in Table 22, the best results were obtained at 0.5 oversampling for credential access (highlighted in green in Table 21). Again, although there are sampling ratios that are statistically equivalent to 0.5 (as shown in Table 22), 0.5 was chosen as the best since it has the smallest amount of oversampled data, thus taking the least computational time.

Table 22. Welch's *t*-test results: UWF-ZeekData22: credential access—random undersampling after splitting.

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	1.252	−0.344	0.858	1.252	0.1 and 0.2 are statistically equal
0.1 vs. 0.3	0.348	−0.632	−0.286	0.348	0.1 and 0.3 are statistically equal
0.1 vs. 0.4	1.023	−0.697	0.611	1.023	0.1 and 0.4 are statistically equal
0.1 vs. 0.5	0.704	−1.528	−0.047	0.704	0.1 and 0.5 are statistically equal, but 0.5 has better recall
0.1 vs. 0.6	1.204	0	1.06	1.204	0.5 and 0.6 are statistically equal
0.1 vs. 0.7	0.814	0.49	0.936	0.814	0.5 and 0.7 are statistically equal
0.1 vs. 0.8	2.461	0.23	2.653	2.461	0.5 is better than 0.8 except for recall
0.1 vs. 0.9	0.786	0.563	1.112	0.786	0.5 and 0.9 are statistically equal
0.1vs. 1	0.408	1.162	1.22	0.408	0.5 and 1.0 are statistically equal

Table 23 presents the classification results for Random Undersampling After Splitting for Privilege Escalation (UWF-ZeekData22) for the various oversampling percentages (0.1 to 1.0, at intervals of 0.1). The best results are highlighted in green.

Table 23. UWF-ZeekData22: privilege escalation—classification results for random undersampling after splitting.

Oversampling %		Accuracy	Precision	Recall	F-Score	Macro Precision
0.1	Avg	0.999	0.900	0.750	0.779	0.949
	SD	NA	0.213	0.273	0.226	0.106
0.2	Avg	0.999	0.813	0.725	0.712	0.906
	SD	NA	0.203	0.343	0.260	0.101
0.3	Avg	0.999	0.820	0.825	0.768	0.909
	SD	NA	0.130	0.275	0.147	0.065
0.4	Avg	0.999	0.800	0.800	0.788	0.899
	SD	NA	0.322	0.331	0.316	0.161
0.5	Avg	0.999	0.843	0.825	0.811	0.921
	SD	NA	0.253	0.275	0.238	0.126
0.6	Avg	0.999	0.745	0.899	0.804	0.872
	SD	NA	0.091	0.135	0.068	0.045
0.7	Avg	0.999	0.704	0.911	0.785	0.852
	SD	NA	0.121	0.147	0.107	0.060
0.8	Avg	0.999	0.707	0.888	0.781	0.853
	SD	NA	0.094	0.157	0.099	0.047
0.9	Avg	0.999	0.738	0.855	0.778	0.869
	SD	NA	0.138	0.131	0.087	0.069
1.0	Avg	0.999	0.759	0.966	0.843	0.879
	SD	NA	0.131	0.071	0.087	0.065
	Averages	0.999	0.7833	0.845	0.7853	0.8915

Results of Welch's *t*-tests for UNSW-NB15 Backdoors for Random Undersampling after Splitting (Table 23) are presented in Table 24. Based on the analysis in Table 24, the best results were obtained at 0.1 oversampling for privilege escalation (highlighted in green in Table 23). Again, although there are sampling ratios that are statistically equivalent to 0.1 (as shown in Table 24), 0.1 was chosen as the best since it has the smallest amount of oversampled data, thus taking the least computational time.

Table 24. Welch's *t*-test results: UWF-ZeekData22: privilege escalation—random undersampling after splitting.

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	0.929	0.179	0.611	0.929	0.1 and 0.2 are statistically equal
0.1 vs. 0.3	1.012	−0.611	0.118	1.012	0.1 and 0.3 are statistically equal
0.1 vs. 0.4	0.817	−0.367	−0.08	0.817	0.1 and 0.4 are statistically equal
0.1 vs. 0.5	0.541	−0.611	−0.307	0.541	0.1 and 0.4 are statistically equal
0.1 vs. 0.6	2.1	−1.552	−0.344	2.1	0.1 is better than 0.6 except for recall
0.1 vs. 0.7	2.52	−1.638	−0.086	2.52	0.1 is better than 0.7 except for recall
0.1 vs. 0.8	2.606	−1.391	−0.027	2.606	0.1 is better than 0.8 except for recall

Table 24. *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. 0.9	1.999	−1.098	0.011	1.999	0.1 is better than 0.9 in precision and macro precision
0.1 vs. 1	1.77	−2.421	−0.834	1.77	0.1 is better than 1 except for recall

After analyzing the classification results using Welch's *t*-tests, for Random Undersampling After Stratified Splitting, it can be seen that an undersampling of 0.5 for the majority of the data after stratified splitting gives the best results when the oversampling is at 0.1, for four of the five datasets. All the UNSW-NB15 datasets performed better at an oversampling of 0.1 and one of the UWF-ZeekData22 datasets, privilege escalation, also performed better at 0.1. Credential access, however, performed better at 0.5. This means that, for four out of the five datasets, generating more synthetic minority class samples beyond 0.1 will not result in a better prediction by the model.

9. Conclusions

In this paper, two different designs that address the issue of class imbalance in network intrusion or cybersecurity datasets were compared using resampling techniques. The objective was to see how combinations of undersampling and oversampling help to better predict the minority classes in highly imbalanced datasets. Comparing both design approaches, we found that the ratio of 0.5 random undersampling to 0.1–0.5 oversampling using BSMOTE works best (based on the dataset) for random undersampling before stratified splitting of the training and testing data. On the other hand, the ratio of 0.5 random undersampling to 0.1 oversampling using BSMOTE works best (in most cases) for random undersampling after stratified splitting of the training and testing data. Random undersampling after oversampling using BSMOTE allows for the use of lower ratios of oversampled data. However, although the average accuracy would appear comparable for both methods, the average precision, recall, and other measures were higher in the random undersampling before splitting. This can be attributed to stratified train/test splitting before random undersampling, ensuring that the train/test samples mimic the actual ratios of the majority to minority classes.

10. Future Work

For future work, we plan to look at the following. We fixed random undersampling to 0.50% of the original data and varied the percentages of oversampling. Future work would vary both undersampling and oversampling. Additionally, we would like to extend this to other data with small minority classes and compare them against other classifiers.

Author Contributions: This work was conceptualized by S.B. (Sikha Bagui), D.M., S.B. (Subhash Bagui), S.S. and D.W.; methodology was performed by S.B. (Sikha Bagui), D.M., S.B. (Subhash Bagui), S.S. and D.W.; validation was performed by S.B. (Sikha Bagui), S.B. (Subhash Bagui), S.S. and D.W.; formal analysis was performed by S.S. and D.W.; investigation was performed by S.S. and D.W.; resources were provided by S.B. (Sikha Bagui), D.M., S.S. and D.W.; data curation was performed by S.S. and D.W; original draft preparation was performed by S.B. (Sikha Bagui), S.S. and D.W., reviewing and editing was performed by S.B. (Sikha Bagui), D.M., S.B. (Subhash Bagui), S.S. and D.W.; visualizations were performed by S.B. (Sikha Bagui), S.S. and D.W., supervision was performed by S.B. (Sikha Bagui), D.M. and S.B. (Subhash Bagui); project administration was performed by S.B. (Sikha Bagui), D.M., S.B. (Subhash Bagui), S.S. and D.W.; funding acquisition was performed by S.B. (Sikha Bagui), D.M. and S.B. (Subhash Bagui). All authors have read and agreed to the published version of the manuscript.

Funding: The research was funded by 2021 NCAE-C-002: Cyber Research Innovation Grant Program, grant number H98230-21-1-0170.

Data Availability Statement: UWF-ZeekData22 is available at datasets.uwf.edu (accessed on 1 March 2023).

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

References

1. Zippia, How Many People Use the Internet? Available online: https://www.zippia.com/advice/how-many-people-use-the-internet/ (accessed on 1 March 2023).
2. CSO, Up to Three Percent of Internet Traffic is Malicious, Researcher Says. Available online: https://www.csoonline.com/article/2122506/up-to-three-percent-of-internet-traffic-is-malicious--researcher-says.html (accessed on 15 February 2023).
3. Bagui, S.; Li, K. Resampling Imbalanced Data for Network Intrusion Detection Datasets. *J. Big Data* **2021**, *8*, 6. [CrossRef]
4. 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, ACT, Australia, 10–12 November 2015; pp. 1–6. [CrossRef]
5. UWF-ZeekData22 Dataset. Available online: Datasets.uwf.edu (accessed on 1 February 2023).
6. Machine Learning Mastery Random Oversampling and Undersampling for Imbalanced Classification. Available online: https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.under_sampling.RandomUnderSampler.html#imblearn.under_sampling.RandomUnderSampler (accessed on 12 December 2022).
7. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
8. Han, H.; Wang, W.-Y.; Mao, B.-G. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In Proceedings of the International Conference on Intelligent Computing, Hefei, China, 23–26 August 2005. [CrossRef]
9. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008; pp. 1322–1328.
10. Abdi, L.; Hashemi, S. To Combat Multi-class Imbalanced Problems by Means of Over-sampling Techniques. *IEEE* **2016**, *28*, 238–251. [CrossRef]
11. Imbalanced-Learn, RandomUnderSampler. Available online: https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html (accessed on 5 January 2023).
12. Shamsudin, H.; Yusof, U.; Jayalakshmi, A.; Akmal Khalid, M. Combining Oversampling and Undersampling Techniques for Imbalanced Classification: A Comparative Study Using Credit Card Fraudulent Transaction Dataset. In Proceedings of the 2020 IEEE 16th International Conference on Control & Automation, Singapore, 9–11 October 2020.
13. Barandela, R.; Sánchez, J.S.; García, V.; Rangel, E. Strategies for Learning in Class Imbalance Problems. *Pattern Recognit.* **2003**, *36*, 849–851. [CrossRef]
14. Vandewiele, G.; Dehaene, I.; Kovács, G.; Sterckx, L.; Janssens, O.; Ongenae, F.; De Backere, F.; De Turck, F.; Roelens, K.; Decruyenaere, J.; et al. Overly Optimistic Prediction Results on Imbalanced Data: Flaws and benefits of Applying Over-sampling. *Artif. Intell. Med.* **2020**. *preprint*. [CrossRef] [PubMed]
15. Bajer, D.; Zonć, B.; Dudjak, M.; Martinović, G. Performance Analysis of SMOTE-based Oversampling Techniques When Dealing with Data Imbalance. In Proceedings of the 2019 International Conference on Systems, Signals and Image Processing (IWSSIP), Osijek, Croatia, 5–7 June 2019; pp. 265–271. [CrossRef]
16. Bagui, S.; Simonds, J.; Plenkers, R.; Bennett, T.A.; Bagui, S. Classifying UNSW-NB15 Network Traffic in the Big Data Framework Using Random Forest in Spark. *Int. J. Big Data Intell. Appl.* **2021**, *2*, 39–61. [CrossRef]
17. Koziarski, M. CSMOUTE: Combined Synthetic Oversampling and Undersampling Technique for Imbalanced Data Classification. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; pp. 1–8. [CrossRef]
18. Liu, A.Y. The Effect of Oversampling and Undersampling on Classifying Imbalanced Text Datasets. Ph.D. Thesis, The University of Texas at Austin, Austin, TX, USA, 2004.
19. Estabrooks, A.; Jo, T.; Japkowicz, N. A Multiple Resampling Method for Learning from Imbalanced Data Sets. *Comput. Intell.* **2004**, *20*, 18–36. [CrossRef]
20. Gonzalez-Cuautle, D.; Hernandez-Suarez, A.; Sanchez-Perez, G.; Toscano-Medina, L.K.; Portillo-Portillo, J.; Olivares-Mercado, J.; Perez-Meana, H.M.; Sandoval-Orozco, A.L. Synthetic Minority Oversampling Technique for Optimizing Classification Tasks in Botnet and Intrusion-Detection-System Datasets. *Appl. Sci.* **2020**, *10*, 794. [CrossRef]
21. Bagui, S.S.; Mink, D.; Bagui, S.C.; Ghosh, T.; Plenkers, R.; McElroy, T.; Dulaney, S.; Shabanali, S. Introducing UWF-ZeekData22: A Comprehensive Network Traffic Dataset Based on the MITRE ATT&CK Framework. *Data* **2023**, *8*, 18. [CrossRef]
22. Bagui, S.; Mink, D.; Bagui, S.; Ghosh, T.; McElroy, T.; Paredes, E.; Khasnavis, N.; Plenkers, R. Detecting Reconnaissance and Discovery Tactics from the MITRE ATT&CK Framework in Zeek Conn Logs Using Spark's Machine Learning in the Big Data Framework. *Sensors* **2022**, *22*, 7999. [CrossRef] [PubMed]
23. Han, J.; Kamber, M.; Pei, J. *Data Mining: Concepts and Techniques*; Morgan Kaufmann: Burlington, MA, USA, 2022.
24. Brieman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 1.

25. SparkApache StringIndexer. Available online: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.StringIndexer.html. (accessed on 1 March 2023).
26. Understand TCP/IP Addressing and Subnetting Basics. Available online: https://docs.microsoft.com/en-us/troubleshoot/windows-client/networking/tcpip-addressing-and-subnetting (accessed on 1 March 2023).
27. Service Name and Transport Protocol Port Number Registry. Available online: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml (accessed on 2 March 2023).
28. Scikit Learn 3.3 Metrics and Scoring: Quantifying the Quality of Predictions. Available online: https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score. (accessed on 12 February 2023).
29. Powders, D.M.W. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *J. Mach. Learn. Technol.* **2011**, *2*, 37–63.
30. sklearn.metrics.precision_recall_fscore_support. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html (accessed on 12 February 2023).