electronics

Article

Sampling-Based Machine Learning Models for Intrusion Detection in Imbalanced Dataset

Zongwen Fan [1,†], Shaleeza Sohail [2,†], Fariza Sabrina [3,*,†] and Xin Gu [4,†]

[1] College of Computer Science and Technology, Huaqiao University, Xiamen 361021, China; zongwen.fan@hqu.edu.cn
[2] College of Engineering, Science and Environment, The University of Newcastle, Callaghan, NSW 2308, Australia; shaleeza.sohail@newcastle.edu.au
[3] School of Engineering and Technology, Central Queensland University, Rockhampton, QLD 4701, Australia
[4] Lincoln Institute of Higher Education, Sydney, NSW 2000, Australia; xin.gu@lincolnau.nsw.edu.au
* Correspondence: f.sabrina@cqu.edu.au
† These authors contributed equally to this work.

Abstract: Cybersecurity is one of the important considerations when adopting IoT devices in smart applications. Even though a huge volume of data is available, data related to attacks are generally in a significantly smaller proportion. Although machine learning models have been successfully applied for detecting security attacks on smart applications, their performance is affected by the problem of such data imbalance. In this case, the prediction model is preferable to the majority class, while the performance for predicting the minority class is poor. To address such problems, we apply two oversampling techniques and two undersampling techniques to balance the data in different categories. To verify their performance, five machine learning models, namely the decision tree, multi-layer perception, random forest, XGBoost, and CatBoost, are used in the experiments based on the grid search with 10-fold cross-validation for parameter tuning. The results show that both the oversampling and undersampling techniques can improve the performance of the prediction models used. Based on the results, the XGBoost model based on the SMOTE has the best performance in terms of accuracy at 75%, weighted average precision at 82%, weighted average recall at 75%, weighted average F1 score at 78%, and Matthews correlation coefficient at 72%. This indicates that this oversampling technique is effective for multi-attack prediction under a data imbalance scenario.

Keywords: cybersecurity; oversampling technique; undersampling technique; multi-class classification; machine learning

Citation: Fan, Z.; Sohail, S.; Sabrina, F.; Gu, X. Sampling-Based Machine Learning Models for Intrusion Detection in Imbalanced Dataset. *Electronics* **2024**, *13*, 1878. https://doi.org/10.3390/electronics13101878

Academic Editors: Abdussalam Elhanashi, Pierpaolo Dini and Andreas Mauthe

Received: 8 March 2024
Revised: 26 April 2024
Accepted: 1 May 2024
Published: 11 May 2024

1. Introduction

The Internet of Things (IoT) is used in a huge number of smart applications that exchange data and perform tasks autonomously like smart agriculture, smart homes, and smart cities [1]. IoT devices produce a huge volume of data that is utilized by smart applications. Due to their large number, distributed nature, and diverse functionalities, IoT networks always face a huge number of cyberattacks [2]. Therefore, securing IoT networks from attackers is critical to ensure the safe and reliable use of these devices and applications. The heterogeneity of the devices and protocols, and the limited processing power, memory, and energy of many IoT devices, are some of the major challenges in choosing security measures [3].

Recently machine learning has been used to detect cyberattacks on smart applications. These machine learning algorithms create security models for detecting attacks based on the training data that consists of data regarding normal and malicious traffic [4]. In most of the datasets used by these algorithms, the proportion of data related to normal and attack classes may be imbalanced. The imbalanced dataset is a big obstacle in detecting attacks accurately, especially for minority classes [5].

In machine learning, it is common to encounter datasets that are imbalanced, where one class has significantly more or fewer instances than other classes. An imbalanced dataset can play a role in creating a biased model favoring the majority classes and performing poorly on the minority classes [6]. Common approaches that have been used to overcome this challenge are using oversampling and undersampling techniques. By randomly duplicating records in smaller or minority classes, oversampling tries to match the number of records to the bigger or majority classes. This can also be accomplished by artificially creating new records or instances. Some of the oversampling approaches include synthetic minority oversampling technique (SMOTE) [7], distributed random oversampling [8], BorderlineSMOTE [9], borderline oversampling with support vector machine (SVM) [10], adaptive synthetic sampling (ADASYN) [11], etc.

On the other hand, undersampling is based on the idea that the number of records in majority classes is to be reduced in order to keep them the same in number as the instances in minority classes. This can be achieved by randomly removing instances from the majority class or selecting a representative subset of instances based on certain criteria. Some of the undersampling approaches are condensed nearest-neighbor undersampling [12], Tomek Links method [13], edited nearest neighbors [14], one-sided selection [15], and instance hardness threshold [16].

Both oversampling and undersampling techniques have their advantages and disadvantages [17,18]. Oversampling can be effective in increasing the number of instances in the minority class and reducing the bias toward the majority class. By providing more instances in the minority class, oversampling can help the model learn the patterns and characteristics of the minority class and improve its ability to generalize to new instances. However, it can also lead to overfitting and generate synthetic instances that do not accurately represent the minority class. Undersampling can be effective in reducing the number of instances in the majority class and focusing on the most informative instances. Undersampling can help to focus on the most informative instances and reduce the noise and redundancy in the dataset as well. However, it can also lead to information loss and remove instances that are important for the model's performance.

It is important to point out that oversampling and undersampling techniques should be used with caution and in conjunction with different preprocessing techniques, such as feature selection and normalization [18]. Furthermore, the choice of oversampling or undersampling technique should be based on the specific characteristics of the dataset and the goals of the machine learning task [19]. In this paper, we focus on exploring the effect of multiple oversampling and undersampling approaches for attack detection for different machine learning algorithms. The IoT dataset we chose shows very poor detection performance for most of the minority classes if no oversampling or undersampling approach is used. The main contributions of the paper are as follows:

- We identify that the traditional machine learning algorithms may not detect minority attack classes when no sampling technique is used. Therefore, in reality, for network attack detection systems, such a limitation may result in failure to detect some of the attacks completely, which emphasizes the importance of using sampling techniques for imbalanced datasets.
- We thoroughly investigate the effect of different oversampling and undersampling techniques on the performance of multiple traditional and ensemble machine learning algorithms.
- We identify the best sampling approach for network attack detection using one of the latest IoT datasets.

Section 2 provides a discussion and analysis of the relevant literature in two areas: machine learning models and relevant research using the IOTID20 dataset. Section 3 provides the details of machine learning models and sampling techniques. Section 4 provides an in-depth analysis of the collected results for the intrusion detection system. A summary of the paper and some options about the possible future work are provided in Section 5.

2. Related Work

In this section, we provide an overview of the machine learning approaches used in the domain of cybersecurity. We also present recent efforts on the use of oversampling and undersampling approaches to improve attack detection for imbalanced datasets. It is important to mention here that oversampling and undersampling approaches are used effectively in improving overall attack detection; however, most of the literature does not specify the detection accuracy for the minority classes. Due to a small proportion of records for the minority classes, the overall detection accuracy can be very high even when the system fails to detect smaller classes completely.

A random forest-based attack detection approach is proposed that uses smart feature selection to improve attack prediction performance [20]. The use of oversampling and some feature selection approaches is explored for imbalanced datasets in the area of cybersecurity, specifically intrusion detection. Decision trees were used for different binary and muti-class attack detection models, and the models performed reasonably [21]. The multilayer perceptron network showed very good anomaly detection abilities with a small number of features for multi-class problems [22]. A combination of random forest and optimization approaches produced very good results for classifying cyberattacks and reducing false alarm rates [23].

The overall accuracy of these proposed approaches was good, but the prediction accuracy for all minority classes was not investigated. Hence, the performance for all attack types cannot be analyzed and compared to this work. With a very small sample size for most minority classes, the overall attack detection accuracy can be very high, but the detection for some of the minority classes may be very low or even zero. This results in completely missing some cyberattacks, which may have a catastrophic effect on the security systems.

An intrusion detection system using an artificial neural network provides very high attack detection when the hyperparameters of the neural network are tuned [24]. Another artificial neural network-based intrusion detection system for binary classification showed promising results for a simulated IoT network [25]. An artificial neural network-based approach for three different levels of classification of attacks is proposed while tuning the hyperparameters for optimal performance [26]. With the proper tuning of hyperparameters, the neural network model showed very high accuracy for most of the cases.

The ANN approach provides a good option for intrusion detection systems and has high performance but faces the challenges of being complex, computationally expensive, and requiring the selection of hyperparameters. Also, the low detection rate for minority classes still exists in the above-mentioned ANN-based approaches, so the above-mentioned works did not consider that specifically.

Ensemble classifiers provide an excellent option for gaining the combined benefits of two different algorithms. Jabbar et al. [27] used an ensemble classifier for the binary detection problem of cyberattacks and combined random forest with another approach. The same research group also proposed ADTree and KNN ensemble classifiers for detecting cyberattacks [28]. To reduce the time of model building and training, a tree-based approach was combined with a bagging method for the classification of attacks [29]. Most of the above-mentioned approaches provide high accuracy for overall attack detection; however, when it comes to multi-class attack detection, the detection rate of smaller or minority attack classes is a great challenge.

Karthik and Krishnan [30] proposed a novel approach to detect IoT attacks using a combination of random forest techniques with a novel oversampling approach. The proposed method was evaluated on different datasets and compared with several approaches; it showed good results in terms of accuracy, precision, recall, and F1 score. Bej et al. [31] proposed a new oversampling technique for imbalanced datasets. The minority samples were scaled and stretched to create new samples for smaller classes. With extensive experiments and testing on numerous imbalanced datasets, the proposed approach showed very promising results.

We used the IOTID20 dataset for testing our approach. Qaddoura et al. [32] addressed the class imbalance issue in the IoTID20 dataset by considering clustering and oversampling techniques. Support vector machine (SVM) with an oversampling technique was investigated for classification and achieved good performance for attack detection at the binary level, where only attacks and normal classes were detected. Farah [33] compared the performance of multiple techniques to detect attacks in the IOTID20 dataset and detected some classes of attacks. However, subcategory attacks that were of minority classes were not detected. Krishnan. Nawaz and Lin [34] compared the attack detection considering random forest, XGBoost, and SVC approaches for detecting normal and attack classes only and achieved very high accuracy. However, the minority class detection was not targeted, as subcategory-based attack detection was not considered.

The main motivation of this work is to detect cyberattacks belonging to minority classes when imbalanced datasets are considered for attack detection, which is a significant concern in almost all datasets in this domain, as previously discussed. In the Section 4, we provide detection accuracy, precision, recall, and F1 score for all attack classes, including minority classes that have very few samples, to show that our approach significantly improves the detection of these small attack classes. In Table 1, we compare our work with other existing studies that use the same dataset to show that we succeeded in detecting minority classes, which were not considered by the other studies due to the small sample sizes for these classes.

Table 1. Comparison with existing approaches.

Research Work	Approaches Used	Detection	Results Provided
Qaddoura et al. [32]	SLFN, SVM	Binary Detection (anomaly/attack)	Only for normal attack
Farrah [33]	Multiple	Binary and Category	Only for normal attack and Categories (majority classes)
Krishnan, Nayaz and Liu [34]	SVC, XGBoost RF	Binary	Only for normal attack
Our work	DT, RF MLP, XGBoost CatBoost	Multi-class Detection	All normal attack, Categories (majority classes) and Subcategories (minority classes)

Undersampling is one efficient method to handle imbalanced datasets as it focuses on reducing the number of samples from the majority classes. An undersampling approach based on the theory of evidence is proposed in evidential undersampling [35]. This approach considers a very important factor, which is to avoid removing meaningful samples. The samples in the majority classes are assigned a soft evidential label after removing unclear samples. When tested with different ML algorithms, this approach outperformed some basic undersampling approaches. An undersampling approach based on consensus clustering is proposed to handle imbalanced learning [36]. The consensus clustering-based scheme used a different combination of clustering algorithms for the undersampling purpose. The results obtained with different ML algorithms showed that different combinations can produce very different results. A novel two-step undersampling approach is proposed [37]. Firstly, the majority class is considered for similar instances, which are grouped together into subclasses. Then, from those subclasses, unrepresentative data samples are removed. The proposed approach performed significantly better than other undersampling approaches.

3. Methods

In this section, we first introduce the machine learning algorithms. Next, we describe the oversampling techniques, followed by the undersampling techniques. Finally, we present the flowchart of the prediction model for the intrusion detection prediction. For researchers to duplicate the outcomes, we have shared our code with the GitHub repository [38]. All undersampling and oversampling approaches that we used are from imblearn library [39]. A snapshot of example source code is shown in Figure 1. All undersampling and oversampling source codes are provided in our GitHub repository [38]. Beside the ReadMe file, five Python Notebook files uploaded to the Github repository, which are ADASYN-sent to github.ipynb, Baseline.ipynb, InstaceHardnessThreshold-sent to github.ipynb, RandomUnderSampler-sent to github.ipynb, and SMOTE-sent to github.ipynb.

```
def kFoldCV(model, data, n_fold=10):
    diff = int(len(data)/n_fold)
    results = np.zeros((1, 4))
    predictY = data[:,-1].astype('int')
    targetY = deepcopy(predictY).astype('int')
#     predictY = deepcopy(data[:,-1]).astype('int')
    cv = StratifiedKFold(n_splits=n_fold)
    X, y = data[:,:-1],data[:,-1].astype('int')
    begin = 0
    for fold, (train_index, test_index) in enumerate(cv.split(X, y)):
        X_train, y_train = X[train_index], y[train_index]
#         print(X_train.shape,y[test_index].shape)
        sc = StandardScaler()
        X_train = sc.fit_transform(X_train)
        X_train, y_train = InstanceHardnessThreshold().fit_resample(X_train, y_train)
        X_test = sc.transform(X[test_index])
        predictY[begin:begin+len(X_test)] = model.fit(X_train, y_train).predict(X_test)
        targetY[begin:begin+len(X_test)] = y[test_index]
        begin += len(X_test)
#         targetY[begin:end] = test[:,-1]
#         predictY[begin:end] = model.fit(X_train, y_train).predict(X_test)
    t = classification_report(targetY, predictY)
    print(t)
    print(matthews_corrcoef(targetY, predictY))
```

Figure 1. A snapshot of the code for InstanceHardnessThreshold undersampling method.

3.1. Machine Learning Algorithms

In this section, five frequently utilized machine learning algorithms are briefly explained, including the decision tree (DT), multilayer perceptron (MLP), random forest (RF), extreme gradient boosting (XGBoost), and category boosting (CatBoost).

3.1.1. DT

The DT [40] is a tree-based supervised algorithm that can be used for classification tasks. A DT algorithm is constructed by three types of nodes, which are the decision node, change node, and end node. Each type of node has its task. More specifically, the decision node indicates a choice that needs to be determined. Consequently, the chance node analyzes the probabilities of the results. The end node presents the ultimate result of a decision pathway. By calculating the value of each option in the tree, the DT is able to achieve promising results by minimizing the risk and maximizing the likelihood.

The primary purpose of the DT algorithm is to obtain the measure of information gain. Specifically, the DT model first evaluates the entropy, as given in Equation (1). After that, the conditional entropy is calculated using Equation (2). Finally, the information gain is obtained using Equation (3).

$$H(D) = -\sum_{k=1}^{K} p_k log_2 P_k, \quad (1)$$

$$H(D|A) = -\sum_{i=1}^{n} p_i H(D|A = x_i), \tag{2}$$

$$g(D, A) = H(D) - H(D|A), \tag{3}$$

where D is a given dataset, K represents the count of categories, n stands for the number of features, p_k denotes the probability rate of the kth category, and p_i signifies the probability of the feature A in the ith subset.

3.1.2. MLP

The MLP is composed of three types of layers, which are the input layer, the hidden layer, and the output layer [41]. Every layer is linked to its neighboring layers. Similarly, every neuron within the hidden and output layers is connected to all neurons in the preceding layer via a weight vector. Each layer proceeds its own computation. The output of each layer is generated by passing the weighted sum of inputs and bias terms through a non-linear activation function, which then becomes the input for the subsequent layer. In the input layer, the number of neurons corresponds to the number of input features, while the output layer represents the model's output. For a binary classification, a single neuron will be generated as the result. The hidden layer neurons reside between the input and output layers, forming connections with both. These interconnected neurons enable communication and information exchange among themselves. Through adjusting weights in the connections between neurons, the MLP can mimic the information analysis and processes like a human brain.

For a binary classification problem, the MLP generates one single neuron in the output layer where its value can be obtained using Equation (4).

$$f(X) = \boldsymbol{W}^2 g(\boldsymbol{W}^1 \boldsymbol{X} + b^1) + b^2, \tag{4}$$

where $\boldsymbol{W}^1$ represents the weights, and b^1 denotes a bias term in the transition from the input layer to the neighboring hidden layer. Likewise, $\boldsymbol{W}^2$ represents the weights, and b^2 denotes a bias term when passing from the hidden layer to the output layer. $g(\cdot)$ signifies an activation function.

3.1.3. RF

The RF is an ensemble algorithm that leverages multiple decision trees [42]. By constructing numerous decision trees using bootstrap samples, the RF algorithm enhances prediction accuracy and stability. It effectively addresses overfitting issues by utilizing resampling and feature selection techniques. During the training process, the RF generates multiple sub-datasets, each containing the same number of samples as the original training set, through resampling. For each sub-dataset, individual decision trees are trained using a recursive partitioning approach. This involves searching for the best feature splits within the selected features. Ultimately, the RF algorithm combines the predictions from all decision trees by taking their average as the final output.

3.1.4. XGBoost

XGBoost is also an ensemble algorithm [43]. To achieve the final result, this algorithm employs gradient boosting to aggregate multiple outcomes from the decision tree-based algorithms. To scale down the impact of overfitting, this algorithm uses shrinkage and feature subsampling techniques. The XGBoost method is tailored to real-world applications that require high computation time and storage memory. Therefore, it is well suited for applications that necessitate parallelization, distributed computing, out-of-core computing, and cache optimization. Additionally, this ensemble algorithm enables parallel tree boosting, alternately referred to as gradient-boosted decision tree and gradient boosting machine.

Gradient boosting aims to discover the function that most effectively approximates the data by optimizing Equation (5).

$$Obj = \sum_i L(\hat{y}_i, y_i) + \sum_i \Omega(f_i), \tag{5}$$

$$\Omega(f) = \gamma T + (1/2)\lambda ||\mathbf{w}||^2, \tag{6}$$

where L represents a convex loss function that quantifies the dissimilarity between the target value y_i and the predicted value $\hat{y}$. The weight vector is denoted by $\mathbf{w}$, f_i refers to the i^{th} function, T signifies the number of leaves in the tree, and $\Omega(f)$ penalizes model complexity, while γT and $\lambda ||\mathbf{w}||^2$ impose constant penalties for each additional tree leaf and extreme weights, respectively.

3.1.5. CatBoost

CatBoost, based on categorical boosting, is an ensemble model that is effective for prediction tasks involving categorical features [44]. Distinguished from the other gradient boosting algorithms, an ordered boosting technique is employed to mitigate the issue of target leakage. Furthermore, it effectively resolves the issues with categorical features by replacing the original features with one or more numerical values. Constructed on the foundation of the traditional gradient boosting-based algorithms that can lead to the overfitting problem, CatBoost addresses this issue by utilizing random permutation for leaf value estimation to minimize the issue of overfitting. It can rapidly construct a model for big data projects with a high level of generalization. By combining many base estimators, this algorithm is able to build a strong competitive prediction model that achieves better performance than random selection.

3.2. Oversampling Techniques

One or more classes that are characterized with few examples are referred to as minority classes, while one or more classes with a significant number of examples are known as majority classes. When minority classes and majority classes are in the same dataset, they cause an imbalanced class distribution. As a common practice in dealing with a binary (two-class) classification problem, class 0 is recognized as the majority class, and class 1 signifies the minority class.

3.2.1. SMOTE

One of the most significant obstacles in classification problems is that there are far more majority classes than minority ones. To overcome this, an oversampling technique was adopted to balance the data before the prediction models were trained. The synthetic minority oversampling technique (SMOTE) was applied to oversample the data, and it was found that it effectively mitigates the overfitting problem of the prediction model to the majority class [7]. The SMOTE is an oversampling technique that generates synthetic samples for minority classes. To mitigate the overfitting problems, this algorithm generates new instances by utilizing the interpolation between the positive instances that are in proximity, with a focus on the feature space.

The SMOTE first randomly selects a minority class instance a. Then, the algorithm searches for its k nearest neighbors which are also minority classes. Consequently, one of the k nearest neighbors b is identified by chance. Connecting a and b to form a line segment in the feature space creates synthetic instances. As a result, the synthetic instances are created as a convex combination of the two chosen neighboring instances a and b.

3.2.2. ADASYN

ADASYN represents a generalized version of SMOTE. This algorithm also creates synthetic instances to oversample the minority classes. However, it considers the density distribution that determines the number of synthetic instances to be generated for sam-

ples, which is difficult to learn [11]. By doing so, this algorithm dynamically adjusts the decision boundaries based on the samples difficult to learn. This is where the fundamental distinction exists between ADASYN and SMOTE.

3.3. Undersampling Techniques

Undersampling techniques are tailored to address the issues of skewed distribution in classification datasets.

3.3.1. RandomUnderSampler

Random undersampling, also termed RandomUnderSampler, arbitrarily selects samples from the majority class and then deletes them from the training dataset [45]. This approach is regarded as the simplest undersampling technique. Although it is simple, it is effective. However, this algorithm is not without limitations. A drawback of this technique is that samples are eliminated without considering their potential usefulness or importance in determining the decision boundary between the classes. In random undersampling models, the instances in the majority class are deleted randomly to reach a balanced distribution. This potentially results in the removal of valuable information.

3.3.2. InstanceHardnessThreshold

Instance hardness threshold (InstanceHardnessThreshold) is an undersampling method that can be used to alleviate class imbalance by removing samples with the aim of balancing the dataset [16]. In other words, the samples classified with a low probability will be removed from the dataset. Consequently, the prediction model can be trained based on the simplified dataset. The probability of misclassification for each sample is defined by the hardness threshold, which is considered the core difference between the instance hardness threshold and other undersampling techniques.

3.4. Flowchart of a Prediction Model for the Intrusion Detection Prediction

A flowchart for intrusion detection prediction using the machine learning models, namely the DT, MLP, RF, XGBoost, and CatBoost, is presented in Figure 2. As illustrated in the figure, parameter tuning incorporated a grid search with 10-fold cross-validation. Specifically, the given dataset was preprocessed. After that, the grid search with 10-fold cross-validation tuned the model parameters with the use of different sampling techniques, including the oversampling techniques (SMOTE and ADASYN) and the undersampling techniques (RandomUnderSampler and InstanceHardnessThreshold). The dataset was divided into ten subsets, with each subset used once as the validation data while the model was trained on the remaining nine subsets. This process was repeated 10 times, with each subset serving as the validation data exactly once. The average performance across all folds was then computed to evaluate the model's performance under different hyperparameter configurations. Finally, with the best parameters obtained, the prediction model was trained and evaluated.

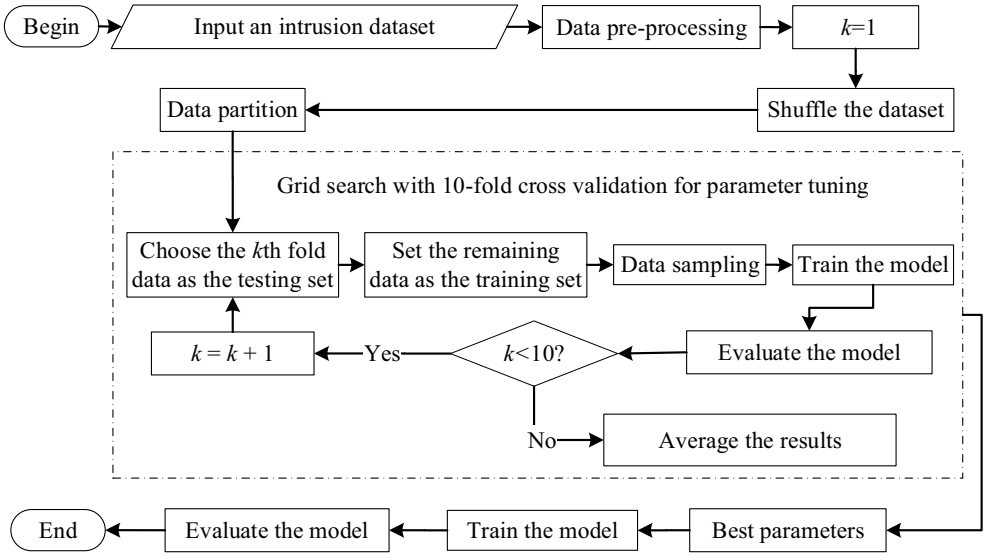

Figure 2. A flowchart showing the prediction procedure for the intrusion detection prediction.

4. Experimental Setup and Results

The main motivation behind this work is to evaluate the efficacy of undersampling and oversampling approaches to improve minority class detection for imbalanced datasets using different machine learning approaches. With extensive experimentation, we compared the detection accuracy of IoT attacks belonging to minority classes using matching learning models outlined in Section 3.

4.1. System Setup

The experiments were conducted and assessed using the Python programming language within Jupyter Notebook from the Anaconda distribution. We utilized the sklearn library in our implementation. Furthermore, the experiments were executed on a computer system featuring an i7-12700H CPU operating at 2.30 GHz and 64.0 GB of RAM.

4.2. Data Description

Datasets are required to train machine learning models for attack detection. Without a dataset, no training/testing of models can be carried out. In recent years, several datasets have been developed and made available by researchers, including UNSW-NB15 [46] and Bot-IoT [47]. Specifically, the IoTID20 dataset [48] was chosen for this research as this is one of the most recent datasets established in the IoT scenario for depicting realistic and up-to-date network traffic features [26]. A number of studies have been conducted based on this dataset for experimentation. Ullah and Mahmoud [48] used this dataset. Anomalous activities in attack detection were divided into binary, category, and subcategory levels. Five machine learning models (i.e., DT, MLP, RF, XGBoost, and CatBoost) were applied to detect IoT attacks. As minority attack classes are important in cybersecurity, we focused on the detection of subcategory attacks (nine-class classification). The IoTID20 dataset, developed by Ullah and Mahmoud [48], contains 86 attributes, including 3 categorical attributes. The categorical attribute is Sub_Cat. The data types in Sub_Cat data column are Mirai-Ackflooding, Mirai-Hostbruterforceg, Mirai-HTTP Flooding, Mirai-UDF Flooding, MITM ARP Spoofing, DoS-Synflooding, Scan Hostport, Scan Port OS, and Normal. The distributions of nine data types of Sub_Cat are presented in Figure 3.

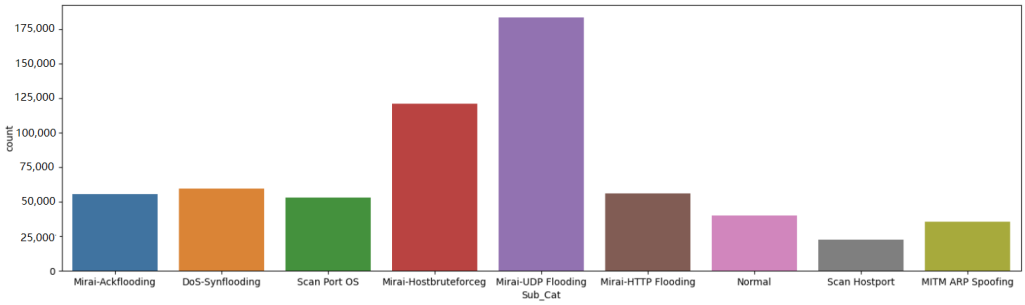

Figure 3. IoTID20 data distribution based on Sub_Cat.

4.3. Experimental Settings

Machine learning models are sensitive to the setting of hyperparameters; hence, we applied the grid search approach to select the best hyperparameters. For a given model, the grid search approach exhaustively generates several candidate values from a list of hyperparameter values to determine the optimal hyperparameters. In this paper, the grid search was used with 10-fold cross-validation in the machine learning models for parameter tuning. To be specific, for the DT, candidate values of [3, 4, 5] were considered for the maximum tree depth [49]. For the MLP, candidate values of [100, 200, 300] for the neuron number in the hidden layer and [100, 200, 300] for the maximum iterations were considered. For the ensemble models (RF, XGBoost, and CatBoost), candidate values of [10, 100, 200] for the number of estimators and [3, 4, 5] for the maximum tree depth in each estimator were considered [50]. The final results were obtained based on the models trained using the optimal hyperparameters. Take the RF as an example; nine pairs of hyperparameters ([10, 3], [10, 4], [10, 5], [100, 3], [100, 4], [100, 5], [200, 3], [200, 3], [200, 4], and [200, 5]) were used to trained and evaluate the model. The pair with the best results were utilized as the optimal hyperparameters for the final evaluation.

For the evaluation of different sampling-based machine learning models, macro-accuracy was used to calculate the accuracy for each class independently and then determine the unweighted average. It treats all classes equally regardless of their imbalance. Similar to macro-accuracy, these metrics account for class imbalance by considering precision, recall, and F1 score for each class and then taking the macro-average. Moreover, weighted accuracy was utilized to take the class imbalance by assigning higher weights to minority classes during calculation. It was calculated as the average accuracy of each class weighted by the number of instances in each class. Similar to weighted accuracy, these metrics account for class imbalance by considering precision, recall, and F1 score for each class and then taking the weighted average. Further, we introduced the Matthews correlation coefficient (MCC) [51] to evaluate the model performance.

4.4. Multi-Class Classification Based on Oversampling

In this section, we compare the results obtained from the oversampling-based machine learning models, namely the DT, MLP, RF, XGBoost, and CatBoost, as shown in Tables 2–6, based on SMOTE and ADASYN. As we can see from Table 2, without the use of oversampling techniques, some results of the DT have zero values for precision, recall, and F1 score. This is because the number of MITM ARP Spoofing (35,377), Mirai-Ackflooding (55,124), and Scan Hostport (22,192) is far less than that of Mirai-UDP Flooding (183,189), making the DT overfit to the majority class, while with the use of SMOTE and ADASYN, this problem can be alleviated. Specifically, MITM ARP Spoofing can be detected with the use of SMOTE, while MITM ARP Spoofing and Mirai-Ackflooding can be detected with the use of ADASYN. Among all the results, the baseline DT has the best MCC.

Table 2. Oversampling for multi-class classification using the DT.

Label	Baseline				SMOTE			ADASYN		
	Precision	**Recall**	**F1 Score**	**Support**	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
DoS-Synflooding	1.00	0.99	1.00	59,391	1.00	1.00	1.00	0.60	1.00	0.75
MITM ARP Spoofing	0.00	0.00	0.00	35,377	0.28	0.41	0.33	0.27	0.70	0.39
Mirai-Ackflooding	0.00	0.00	0.00	55,124	0.00	0.00	0.00	0.32	0.12	0.17
Mirai-HTTP Flooding	1.00	0.00	0.00	55,818	0.25	0.86	0.39	0.27	0.60	0.37
Mirai-Hostbruteforceg	0.70	0.59	0.64	121,178	0.61	0.30	0.40	0.99	0.19	0.32
Mirai-UDP Flooding	0.58	0.97	0.72	183,189	1.00	0.59	0.74	1.00	0.59	0.74
Normal	0.84	0.80	0.81	40,073	1.00	0.61	0.76	0.52	0.83	0.64
Scan Hostport	0.00	0.00	0.00	22,192	0.00	0.00	0.00	0.00	0.00	0.00
Scan Port OS	0.43	0.94	0.59	53,073	0.38	0.95	0.54	0.42	0.79	0.55
Accuracy			0.62	625,415			0.54			0.53
Macro avg	0.50	0.48	0.42	625,415	0.50	0.52	0.46	0.49	0.53	0.44
Weighted avg	0.58	0.62	0.53	625,415	0.64	0.54	0.54	0.68	0.53	0.51
MCC			0.55	625,415			0.50			0.48

Oversampling for multi-class classification using the MLP can be found in Table 3. As we can see from the table, similar to the results in Table 2, some results of the MLP have low values for precision, recall, and F1 score (e.g., Mirai-Ackflooding) without the use of oversampling techniques. However, with the use of SMOTE and ADASYN, the performance of the minority attack precision can be improved. For example, the F1 score of Mirai-Ackflooding attack is improved from 28% to 38% and 42% with the use of SMOTE and ADASYN, respectively. The F1 score of the Scan Hostport attack is improved from 11% to 34% and 33% with the use of SMOTE and ADASYN, respectively. In addition, with the use of the ADASYN technique, the performance of MLP can be improved from 63% macro-average accuracy to 68% and from 70% weighted average F1 score to 73%. However, the ADASYN-based MLP sacrifices the performance of accuracy and MCC.

Table 3. Oversampling for multi-class classification using the MLP.

Label	Baseline				SMOTE			ADASYN		
	Precision	**Recall**	**F1 Score**	**Support**	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
DoS-Synflooding	1.00	1.00	1.00	59,391	1.00	1.00	1.00	0.97	1.00	0.98
MITM ARP Spoofing	0.79	0.80	0.79	35,377	0.72	0.91	0.80	0.58	0.86	0.69
Mirai-Ackflooding	0.32	0.25	0.28	55,124	0.32	0.49	0.38	0.32	0.42	0.36
Mirai-HTTP Flooding	0.33	0.32	0.32	55,818	0.32	0.47	0.38	0.33	0.46	0.38
Mirai-Hostbruteforceg	0.72	0.84	0.77	121,178	0.91	0.73	0.81	0.81	0.66	0.73
Mirai-UDP Flooding	0.80	0.79	0.80	183,189	1.00	0.70	0.82	1.00	0.70	0.82
Normal	0.96	0.88	0.91	40,073	0.91	0.91	0.91	0.70	0.96	0.81
Scan Hostport	0.62	0.06	0.11	22,192	0.27	0.47	0.34	0.27	0.42	0.33
Scan Port OS	0.62	0.84	0.71	53,073	0.65	0.62	0.63	0.57	0.47	0.52
Accuracy			0.71	625,415			0.71			0.67
Macro avg	0.68	0.64	0.63	625,415	0.68	0.70	0.68	0.62	0.66	0.63
Weighted avg	0.71	0.71	0.70	625,415	0.78	0.71	0.73	0.74	0.67	0.69
MCC			0.66	625,415			0.66			0.62

Oversampling for multi-class classification using the RF can be found in Table 4. As we can see from the table, the F1 score results from the baseline are low, while the results from SMOTE and ADASYN are greatly improved. For example, for [Mirai-Ackflooding, Mirai-HTTP Flooding, Scan Hostport], their results are improved from [1%, 8%, 0%] to [44%, 41%, 16%] and [29%, 39%, 16%] using SMOTE and ADASYN, respectively. This is very important in real-world intrusion detection applications when the intrusion will severely affect user information security. According to the results, the SMOTE-based RF has the best performance in terms of macro-average F1 score, weighted average F1 score, and MCC.

Table 4. Oversampling for multi-class classification using the RF.

Label	Baseline				SMOTE			ADASYN		
	Precision	**Recall**	**F1 Score**	**Support**	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
DoS-Synflooding	1.00	0.99	1.00	59,391	1.00	1.00	1.00	0.95	1.00	0.97
MITM ARP Spoofing	0.92	0.16	0.27	35,377	0.32	0.88	0.47	0.31	0.58	0.40
Mirai-Ackflooding	0.41	0.00	0.01	55,124	0.30	0.44	0.36	0.30	0.28	0.29
Mirai-HTTP Flooding	0.37	0.05	0.08	55,818	0.31	0.41	0.35	0.30	0.53	0.39
Mirai-Hostbruteforceg	0.45	0.91	0.60	121,178	0.89	0.22	0.35	0.96	0.20	0.34
Mirai-UDP Flooding	0.66	0.90	0.76	183,189	0.98	0.71	0.82	0.99	0.70	0.82
Normal	0.85	0.81	0.83	40,073	0.80	0.85	0.82	0.52	0.95	0.67
Scan Hostport	0.91	0.00	0.00	22,192	0.18	0.14	0.16	0.14	0.18	0.16
Scan Port OS	0.73	0.24	0.36	53,073	0.47	0.83	0.60	0.32	0.55	0.40
Accuracy			0.62	625,415			0.60			0.56
Macro avg	0.70	0.45	0.43	625,415	0.58	0.61	0.55	0.53	0.55	0.49
Weighted avg	0.64	0.62	0.54	625,415	0.72	0.60	0.60	0.70	0.56	0.56
MCC			0.55	625,415			0.55			0.51

Table 5. Oversampling for multi-class classification using the XGBoost.

Label	Baseline				SMOTE			ADASYN		
	Precision	**Recall**	**F1 Score**	**Support**	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
DoS-Synflooding	1.00	1.00	1.00	59,391	1.00	1.00	1.00	1.00	1.00	1.00
MITM ARP Spoofing	0.96	0.96	0.96	35,377	0.92	0.98	0.95	0.86	0.99	0.92
Mirai-Ackflooding	0.30	0.31	0.30	55,124	0.29	0.55	0.38	0.30	0.43	0.36
Mirai-HTTP Flooding	0.31	0.29	0.30	55,818	0.28	0.31	0.29	0.31	0.44	0.36
Mirai-Hostbruteforceg	0.84	0.95	0.89	121,178	0.97	0.86	0.91	0.93	0.87	0.90
Mirai-UDP Flooding	0.81	0.78	0.79	183,189	1.00	0.70	0.83	1.00	0.70	0.83
Normal	0.94	0.92	0.93	40,073	0.91	0.94	0.92	0.74	0.98	0.84
Scan Hostport	0.96	0.42	0.59	22,192	0.49	0.60	0.54	0.44	0.65	0.53
Scan Port OS	0.79	0.88	0.83	53,073	0.80	0.83	0.82	0.82	0.59	0.69
Accuracy			0.76	625,415			0.75			0.74
Macro avg	0.77	0.72	0.73	625,415	0.74	0.75	0.74	0.71	0.74	0.71
Weighted avg	0.76	0.76	0.76	625,415	0.82	0.75	0.77	0.80	0.74	0.76
MCC			0.72	625,415			0.72			0.70

Table 6. Oversampling for multi-class classification using the CatBoost.

Label	Baseline				SMOTE			ADASYN		
	Precision	**Recall**	**F1 Score**	**Support**	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
DoS-Synflooding	1.00	1.00	1.00	59,391	1.00	1.00	1.00	0.98	1.00	0.99
MITM ARP Spoofing	0.90	0.90	0.90	35,377	0.83	0.97	0.89	0.74	0.98	0.84
Mirai-Ackflooding	0.31	0.29	0.30	55,124	0.31	0.60	0.41	0.32	0.45	0.37
Mirai-HTTP Flooding	0.32	0.27	0.29	55,818	0.31	0.33	0.32	0.32	0.45	0.38
Mirai-Hostbruteforceg	0.81	0.93	0.87	121,178	0.95	0.83	0.88	0.92	0.82	0.86
Mirai-UDP Flooding	0.79	0.80	0.79	183,189	1.00	0.70	0.83	1.00	0.70	0.82
Normal	0.94	0.91	0.92	40,073	0.89	0.93	0.91	0.72	0.97	0.83
Scan Hostport	0.92	0.39	0.55	22,192	0.47	0.59	0.52	0.43	0.64	0.52
Scan Port OS	0.77	0.85	0.81	53,073	0.80	0.81	0.81	0.81	0.58	0.68
Accuracy			0.75	625,415			0.75			0.73
Macro avg	0.75	0.70	0.71	625,415	0.73	0.75	0.73	0.69	0.73	0.70
Weighted avg	0.75	0.75	0.75	625,415	0.82	0.75	0.77	0.79	0.73	0.75
MCC			0.70	625,415			0.71			0.69

Oversampling for multi-class classification using the XGBoost can be found in Table 5. As we can see from the table, XGBoost has no zero results. The reason could be that it is an ensemble model that integrates multiple estimators for the final prediction. Although the baseline of XGBoost outperforms the DT, MLP, and RF in Tables 2–4, with the use of SMOTE and ADASYN, the performance of the XGBoost model can be further improved, from 76% weighted average precision to 82% and 80%, respectively. According to the results, the SMOTE-based XGBoost has the best weighted average F1 score and MCC.

Oversampling for multi-class classification using the CatBoost can be found in Table 6. As we can see from the table, similar to the results in Table 5, the CatBoost model has no zero results. The reason could be that CatBoost is also an ensemble model that can reduce the effect of the imbalance problem. Although the baseline of CatBoost outperforms the DT, MLP, and RF in Tables 2–4, its performance is slightly worse than the XGBoost model in Table 5. However, with the use of SMOTE, its performance can be further improved, from 75% weighted average F1 score to 77%. According to the results, the SMOTE-based model has the best performance for accuracy, precision, recall, F1 score, and MCC.

4.5. Multi-Class Classification Based on Undersampling

Although the oversampling techniques are able to improve machine learning models' performance in the dataset of imbalanced scenarios, in this study, we conducted experiments using undersampling techniques to see if the machine learning models could still be improved. We compare the results obtained from the undersampling-based machine learning models, namely the DT, MLP, RF, XGBoost, and CatBoost, as shown in Tables 7–11, based on the RandomUnderSampler and InstanceHardnessThreshold. It is worth noting that, with the use of undersampling techniques, fewer samples are used for training, which can improve the training efficiency. This is more practical in real-world applications, especially in big data scenarios. As we can see from Table 7, with the use of RandomUnderSampler and InstanceHardnessThreshold, the performance of DT can be improved. This is because the RandomUnderSampler is utilized to randomly remove the samples, which will remove some important information. However, with the use of InstanceHardnessThreshold, the performance of DT can be improved from [62%, 42%, 53%, 0.55] to [66%, 58%, 66%, 0.61] in terms of accuracy, macro-average F1 score, weighted average, and MCC, respectively.

Table 7. Undersampling for multi-class classification using DT.

Label	Baseline				RandomUnderSampler			InstanceHardnessThreshold		
	Precision	Recall	F1 Score	Support	Precision	Recall	F1 Score	Precision	Recall	F1 Score
DoS-Synflooding	1.00	0.99	1.00	59,391	1.00	1.00	1.00	1.00	0.99	0.99
MITM ARP Spoofing	0.00	0.00	0.00	35,377	0.33	0.53	0.41	0.65	0.56	0.60
Mirai-Ackflooding	0.00	0.00	0.00	55,124	0.30	0.00	0.01	0.31	0.56	0.40
Mirai-HTTP Flooding	1.00	0.00	0.00	55,818	0.25	0.85	0.39	0.34	0.10	0.16
Mirai-Hostbruteforceg	0.70	0.59	0.64	121,178	0.70	0.33	0.45	0.54	0.80	0.64
Mirai-UDP Flooding	0.58	0.97	0.72	183,189	1.00	0.59	0.74	0.98	0.71	0.82
Normal	0.84	0.80	0.81	40,073	0.95	0.67	0.79	0.82	0.81	0.81
Scan Hostport	0.00	0.00	0.00	22,192	0.17	0.17	0.17	0.86	0.04	0.08
Scan Port OS	0.43	0.94	0.59	53,073	0.44	0.89	0.59	0.64	0.75	0.69
Accuracy			0.62	625,415			0.56			0.66
Macro avg	0.50	0.48	0.42	625,415	0.57	0.56	0.50	0.68	0.59	0.58
Weighted avg	0.58	0.62	0.53	625,415	0.70	0.56	0.56	0.72	0.66	0.66
MCC			0.55	625,415			0.52			0.61

Undersampling for multi-class classification using the MLP model can be found in Table 8. As shown in the table, unlike the results in Table 2, although with fewer samples obtained using the InstanceHardnessThreshold model, the undersampling-based MLP can greatly outperform its baseline. Without the use of undersampling techniques, sample results have low values, while with the use of RandomUnderSampler and InstanceHardnessThreshold, this problem can be improved. In addition, comparing the two undersampling techniques, the performance of InstanceHardnessThreshold-based MLP outperforms RandomUnderSampler-based MLP. The macro-average F1 score of the baseline is 63%, while the macro-average F1 score of InstanceHardnessThreshold-based MLP is 66%; the weighted average F1 score of the baseline is 70%, while the weighted average F1 score of InstanceHardnessThreshold-based MLP is 72%. According to the results, the MLP has the best performance for MCC.

Table 8. Undersampling for multi-class classification using MLP.

Label	Baseline				RandomUnderSampler			InstanceHardnessThreshold		
	Precision	Recall	F1 Score	Support	Precision	Recall	F1 Score	Precision	Recall	F1 Score
DoS-Synflooding	1.00	1.00	1.00	59,391	1.00	1.00	1.00	1.00	1.00	1.00
MITM ARP Spoofing	0.79	0.80	0.79	35,377	0.59	0.93	0.72	0.79	0.78	0.78
Mirai-Ackflooding	0.32	0.25	0.28	55,124	0.31	0.56	0.40	0.30	0.49	0.37
Mirai-HTTP Flooding	0.33	0.32	0.32	55,818	0.31	0.33	0.32	0.31	0.43	0.36
Mirai-Hostbruteforceg	0.72	0.84	0.77	121,178	0.81	0.59	0.68	0.87	0.72	0.79
Mirai-UDP Flooding	0.80	0.79	0.80	183,189	1.00	0.70	0.82	1.00	0.70	0.82
Normal	0.96	0.88	0.91	40,073	0.90	0.89	0.90	0.98	0.85	0.91
Scan Hostport	0.62	0.06	0.11	22,192	0.24	0.47	0.31	0.19	0.46	0.27
Scan Port OS	0.62	0.84	0.71	53,073	0.59	0.55	0.57	0.67	0.60	0.63
Accuracy			0.71	625,415			0.67			0.69
Macro avg	0.68	0.64	0.63	625,415	0.64	0.67	0.64	0.68	0.67	0.66
Weighted avg	0.71	0.71	0.70	625,415	0.75	0.67	0.69	0.78	0.69	0.72
MCC			0.66	625,415			0.62			0.64

Table 9. Undersampling for multi-class classification using RF.

Label	Baseline				RandomUnderSampler			InstanceHardnessThreshold		
	Precision	Recall	F1 Score	Support	Precision	Recall	F1 Score	Precision	Recall	F1 Score
DoS-Synflooding	1.00	0.99	1.00	59,391	1.00	1.00	1.00	1.00	1.00	1.00
MITM ARP Spoofing	0.92	0.16	0.27	35,377	0.32	0.89	0.47	0.90	0.16	0.27
Mirai-Ackflooding	0.41	0.00	0.01	55,124	0.30	0.40	0.34	0.30	0.42	0.35
Mirai-HTTP Flooding	0.37	0.05	0.08	55,818	0.30	0.46	0.36	0.34	0.20	0.25
Mirai-Hostbruteforceg	0.45	0.91	0.60	121,178	0.90	0.22	0.35	0.43	0.92	0.59
Mirai-UDP Flooding	0.66	0.90	0.76	183,189	0.99	0.70	0.82	0.93	0.73	0.82
Normal	0.85	0.81	0.83	40,073	0.79	0.85	0.82	0.73	0.84	0.78
Scan Hostport	0.91	0.00	0.00	22,192	0.17	0.11	0.13	0.70	0.00	0.01
Scan Port OS	0.73	0.24	0.36	53,073	0.47	0.83	0.60	0.00	0.00	0.00
Accuracy			0.62	625,415			0.60			0.60
Macro avg	0.70	0.45	0.43	625,415	0.58	0.61	0.54	0.59	0.47	0.45
Weighted avg	0.64	0.62	0.54	625,415	0.73	0.60	0.60	0.63	0.60	0.57
MCC			0.55	625,415			0.55			0.54

Undersampling for multi-class classification using the RF can be found in Table 9. As we can see from the table, with the use of undersampling techniques, the performance of RF is improved. For the macro average F1 score and weighted average F1 score of RF, their results are improved from [43%, 54%] to [54%, 60%] and [45%, 57%] with the use of RandomUnderSampler and InstanceHardnessThreshold, respectively. According to all the results, the RF and RandomUnderSampler-based RF have the same MCC.

Table 10. Undersampling for multi-class classification using XGBoost.

Label	Baseline				RandomUnderSampler			InstanceHardnessThreshold		
	Precision	Recall	F1 Score	Support	Precision	Recall	F1 Score	Precision	Recall	F1 Score
DoS-Synflooding	1.00	1.00	1.00	59,391	1.00	1.00	1.00	1.00	1.00	1.00
MITM ARP Spoofing	0.96	0.96	0.96	35,377	0.89	0.98	0.93	0.94	0.87	0.90
Mirai-Ackflooding	0.30	0.31	0.30	55,124	0.31	0.53	0.39	0.28	0.49	0.35
Mirai-HTTP Flooding	0.31	0.29	0.30	55,818	0.31	0.41	0.35	0.28	0.39	0.32
Mirai-Hostbruteforceg	0.84	0.95	0.89	121,178	0.97	0.84	0.90	0.96	0.81	0.88
Mirai-UDP Flooding	0.81	0.78	0.79	183,189	1.00	0.70	0.83	1.00	0.70	0.83
Normal	0.94	0.92	0.93	40,073	0.90	0.94	0.92	0.99	0.86	0.92
Scan Hostport	0.96	0.42	0.59	22,192	0.48	0.61	0.54	0.31	0.85	0.45
Scan Port OS	0.79	0.88	0.83	53,073	0.81	0.82	0.81	0.88	0.53	0.66
Accuracy			0.76	625,415			0.75			0.71
Macro avg	0.77	0.72	0.73	625,415	0.74	0.76	0.74	0.74	0.72	0.70
Weighted avg	0.76	0.76	0.76	625,415	0.82	0.75	0.78	0.83	0.71	0.75
MCC			0.72	625,415			0.72			0.68

Table 11. Undersampling for multi-class classification using CatBoost.

Label	Baseline			RandomUnderSampler				InstanceHardnessThreshold		
	Precision	Recall	F1 Score	Support	Precision	Recall	F1 Score	Precision	Recall	F1 Score
DoS-Synflooding	1.00	1.00	1.00	59,391	1.00	1.00	1.00	1.00	1.00	1.00
MITM ARP Spoofing	0.90	0.90	0.90	35,377	0.80	0.97	0.87	0.89	0.83	0.86
Mirai-Ackflooding	0.31	0.29	0.30	55,124	0.31	0.52	0.39	0.28	0.49	0.36
Mirai-HTTP Flooding	0.32	0.27	0.29	55,818	0.32	0.43	0.37	0.28	0.38	0.32
Mirai-Hostbruteforceg	0.81	0.93	0.87	121,178	0.95	0.81	0.88	0.94	0.79	0.86
Mirai-UDP Flooding	0.79	0.80	0.79	183,189	1.00	0.70	0.82	1.00	0.70	0.83
Normal	0.94	0.91	0.92	40,073	0.89	0.93	0.91	0.99	0.85	0.92
Scan Hostport	0.92	0.39	0.55	22,192	0.46	0.60	0.52	0.29	0.83	0.43
Scan Port OS	0.77	0.85	0.81	53,073	0.80	0.79	0.80	0.87	0.53	0.66
Accuracy			0.75	625,415			0.75			0.71
Macro avg	0.75	0.70	0.71	625,415	0.73	0.75	0.73	0.73	0.71	0.69
Weighted avg	0.75	0.75	0.75	625,415	0.81	0.75	0.77	0.82	0.71	0.74
MCC			0.70	625,415			0.71			0.67

Undersampling for multi-class classification using the XGBoost can be found in Table 10. As shown in the table, similar to the results of RF in Table 9, with the use of undersampling techniques, the performance of baseline has similar results compared to the RandomUnderSampler-based XGBoost. The reason could be that although the RandomUnderSampler removes samples, the XGBoost model can still maintain its high performance by integrating multiple estimators for the final prediction. Although the baseline of XGBoost outperforms the DT, MLP and RF in Tables 7–9, with the use of RandomUnderSampler, the performance of the XGBoost model can be further improved, from 76% weighted average F1 score to 78%. From the results, it is evident that in terms of the macro-average F1 score, weighted average F1 score, and MCC, RandomUnderSampler-based XGBoost has the best performance.

Undersampling for multi-class classification using the CatBoost can be found in Table 11. As we can see from the table, although the baseline of CatBoost outperforms the DT, MLP and RF in Tables 7–9, its performance is slightly worse than the XGBoost model in Table 10. Based on all the results, RandomUnderSampler-based model has the best performance in terms of macro-average F1 score, weighted average F1 score, and MCC.

4.6. Comparison of Machine Learning Using Different Sampling Techniques for Multi-Class Classification

Based on the results of oversampling techniques in Tables 2–6, the ensemble models (XGBoost, and CatBoost) have better performance than the single models (DT and MLP). This is reasonable because the ensemble model is able to reduce the overfitting problem by aggregating multiple estimators for final prediction. In addition, with the use of SMOTE, all machine learning models used can be further improved. Based on all the results, the SMOTE-based XGBoost has the best performance for this multi-class classification task. Based on the results of undersampling techniques in Tables 7–11, the ensemble models (XGBoost and CatBoost) have better performance than the single models (DT and MLP).

We can also see from Table 12 that XGBoost and CatBoost have high performance with the use of oversampling or undersampling techniques. It is surprising to see that with the use of undersampling techniques, the performance of machine learning models is similar to the oversampling-based machine learning models. The reasons could be that (1) undersampling reduces the size of the majority class, which can simplify the learning task for the model; (2) oversampling techniques may introduce synthetic instances into the minority class, which could potentially add noise to the dataset; (3) undersampling ensures that the model focuses on the most relevant instances in the dataset by reducing the dominance of the majority class, which could lead to better discrimination between classes and improved model performance; and (4) oversampling techniques may generate synthetic instances that are outliers in feature space, potentially affecting the model's performance. However, the choice between undersampling and oversampling should focus

on the specific problems. According to all the results, the SMOTE-based XGBoost and CatBoost have the best performance for this multi-class classification task, with an accuracy of 75%, a weighted average precision of 82%, a weighted average recall of 75%, and a weighted average F1 score of 77%, while the RandomUnderSampler-based CatBoost has similar performance with their results of [75%, 81%, 75%, 77%] for [accuracy, weighted average precision, weighted average recall, weighted average F1 score].

Table 12. Comparison of machine learning using different sampling techniques for multi-class classification.

Method	**Baseline**				**SMOTE**				**ADASYN**			
	Accuracy	**Precision**	**Recall**	**F1 Score**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
DT	0.62	0.58	0.62	0.54	0.64	0.54	0.54	0.60	0.53	0.68	0.53	0.51
MLP	0.71	0.71	0.71	0.70	0.71	0.78	0.71	0.73	0.67	0.74	0.67	0.69
RF	0.62	0.64	0.62	0.54	0.60	0.72	0.60	0.60	0.56	0.70	0.56	0.56
XGBoost	0.76	0.76	0.76	0.76	0.75	0.82	0.75	0.77	0.74	0.80	0.74	0.76
CatBoot	0.75	0.75	0.75	0.75	0.75	0.82	0.75	0.77	0.73	0.79	0.73	0.75
Method	**Baseline**				**RandomUnderSampler**				**InstanceHardnessThreshold**			
DT	0.62	0.58	0.62	0.54	0.56	0.70	0.56	0.56	0.66	0.72	0.66	0.66
MLP	0.71	0.71	0.71	0.70	0.67	0.75	0.67	0.69	0.69	0.78	0.69	0.72
RF	0.62	0.64	0.62	0.54	0.60	0.73	0.60	0.60	0.60	0.63	0.60	0.57
XGBoost	0.76	0.76	0.76	0.76	0.75	0.82	0.75	0.78	0.71	0.83	0.71	0.75
CatBoot	0.75	0.75	0.75	0.75	0.75	0.81	0.75	0.77	0.71	0.82	0.71	0.74

5. Conclusions and Future Work

In the context of a smart home environment, one of the major challenges is the users' inability to understand and take the necessary security precautions. However, the data of attacks are imbalanced, making it difficult to accurately predict the right category. In this paper, five machine learning models were used for security attack detection on smart applications. In addition, oversampling and undersampling techniques were introduced to solve the imbalanced problem. The results show that the SMOTE-based XGBoost has the best performance with the best accuracy, weighted average precision, weighted average recall, weighted average F1 score, and MCC, with values of 75%, 82%, 75%, 77%, and 72%, respectively. This indicates that these sampling techniques are effective for multi-attack prediction. Further, without the use of sampling techniques, the traditional machine learning models could not detect minority attack classes in some cases, while the model with the sampling techniques used was able to address this problem. This indicates that this sampling-based model is effective for intrusion detection.

However, deploying machine learning models in the real-world IoT environment presents several challenges. The implementation of these models in IoT devices, which often have limited resources, introduces several considerations that can impact their effectiveness and feasibility. In addition, the computational requirements of the models directly impact their deployment on IoT devices. For model training, the computation time ranged from 295 s to 56,777 s depending on which machine learning model and sampling technique was used.

5.1. Threats to Validity

In terms of threats to validity, the first is data quality. The data used to train models or inform research should be accurate and reliable. This means ensuring that data collection methods are valid and that the data are free from errors or biases. In addition, given the class imbalance problem discussed in this paper, sampling techniques (oversampling or undersampling) should be applied to alleviate this problem. In addition, the security of the system is also important. The system should be designed to resist tampering and ensure that it operates as intended without being compromised. Finally, cognitive understanding

of system results is crucial for allowing users to understand how decisions are made. By doing so, users can trust the system and understand its limitations.

5.2. Future Work

For future work, considering that the undersampling-based XGBoost model showed the best performance, we will investigate more of the RandomUnderSampler technique to further address the imbalanced problem. In addition, as the deep learning approaches have feature learning capabilities, we will investigate how to integrate feature generation into the ensemble models to further improve detection accuracy. Finally, considering the sampling techniques that can be used to improve the prediction performance for imbalanced data, more advanced techniques (e.g., generative adversarial networks [52–55]) will be investigated to generate synthetic data and handle imbalance problems.

Author Contributions: Conceptualization, S.S., F.S., X.G. and Z.F.; methodology, Z.F.; validation, Z.F., X.G., F.S. and S.S.; formal analysis, X.G., S.S., Z.F. and F.S.; investigation, X.G., S.S., Z.F., and F.S.; resources, F.S., X.G., S.S., and Z.F.; data curation, X.G., S.S., Z.F. and F.S.; writing—original draft preparation, Z.F.; writing—review and editing, S.S., F.S., Z.F. and X.G.; visualization, Z.F. and X.G.; supervision, X.G., S.S., Z.F. and F.S.; project administration, Z.F. All authors have read and agreed to the published version of the manuscript.

Funding: This study was conducted without external funding or financial support.

Data Availability Statement: The data analyzed during this study are available via public bibliographic databases and can be found on Github at: https://github.com/Zongwen-Fan/SamplingML (accessed on 7 March 2024).

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

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
SMOTE	Synthetic minority oversampling technique
SVM	Support vector machine
ADASYN	Adaptive synthetic sampling
DT	Decision tree
MLP	Multilayer perceptron
RF	Random forest
XGBoost	Extreme gradient boosting
CatBoost	Category boosting

References

1. Perwej, Y.; Haq, K.; Parwej, F.; Mumdouh, M.; Hassan, M. The internet of things (IoT) and its application domains. *Int. J. Comput. Appl.* **2019**, *975*, 182. [CrossRef]
2. Hafeez, I.; Antikainen, M.; Ding, A.Y.; Tarkoma, S. IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 45–59. [CrossRef]
3. Farooq, U.; Tariq, N.; Asim, M.; Baker, T.; Al-Shamma'a, A. Machine learning and the Internet of Things security: Solutions and open challenges. *J. Parallel Distrib. Comput.* **2022**, *162*, 89–104. [CrossRef]
4. Shafiq, M.; Tian, Z.; Sun, Y.; Du, X.; Guizani, M. Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city. *Future Gener. Comput. Syst.* **2020**, *107*, 433–442. [CrossRef]
5. Rani, M. Effective network intrusion detection by addressing class imbalance with deep neural networks multimedia tools and applications. *Multimed. Tools Appl.* **2022**, *81*, 8499–8518. [CrossRef]
6. Pirizadeh, M.; Alemohammad, N.; Manthouri, M.; Pirizadeh, M. A new machine learning ensemble model for class imbalance problem of screening enhanced oil recovery methods. *J. Pet. Sci. Eng.* **2021**, *198*, 108214. [CrossRef]
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. Moreo, A.; Esuli, A.; Sebastiani, F. Distributional random oversampling for imbalanced text classification. In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, Pisa, Italy, 17–21 July 2016; pp. 805–808.

9. Han, H.; Wang, W.Y.; Mao, B.H. Borderline-SMOTE: A new over-sampling method in imbalanced datasets learning. In *Advances in Intelligent Computing, Proceedings of the International Conference on Intelligent Computing, ICIC 2005, Hefei, China, 23–26 August 2005*; Springer: Berlin/Heidelberg, Germany, 2005; Part I, pp. 878–887.
10. Nguyen, H.M.; Cooper, E.W.; Kamei, K. Borderline over-sampling for imbalanced data classification. *Int. J. Knowl. Eng. Soft Data Paradig.* **2011**, *3*, 4–21. [CrossRef]
11. 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.
12. Siddappa, N.G.; Kampalappa, T. Adaptive condensed nearest neighbor for imbalance data classification. *Int. J. Intell. Eng. Syst.* **2019**, *12*, 104–113. [CrossRef]
13. Elhassan, T.; Aljurf, M. Classification of imbalance data using tomek link (T-Link) combined with random under-sampling (RUS) as a data reduction method. *Glob. J. Technol. Optim S* **2016**, *1*, 1–11
14. Putrada, A.G.; Abdurohman, M.; Perdana, D.; Nuha, H.H. Shuffle Split-Edited Nearest Neighbor: A Novel Intelligent Control Model Compression for Smart Lighting in Edge Computing Environment. In *Information Systems for Intelligent Systems, Proceedings of the ISBM 2022*; Springer: Singapore, 2023; pp. 219–227.
15. Kubat, M.; Matwin, S. Addressing the curse of imbalanced training sets: One-sided selection. In Proceedings of the 14th International Conference on Machine Learning, San Francisco, CA, USA, 8–12 July 1997; Volume 97, p. 179.
16. Smith, M.R.; Martinez, T.; Giraud-Carrier, C. An instance level analysis of data complexity. *Mach. Learn.* **2014**, *95*, 225–256. [CrossRef]
17. Shelke, M.S.; Deshmukh, P.R.; Shandilya, V.K. A review on imbalanced data handling using undersampling and oversampling technique. *Int. J. Recent Trends Eng. Res* **2017**, *3*, 444–449.
18. Wongvorachan, T.; He, S.; Bulut, O. A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining. *Information* **2023**, *14*, 54. [CrossRef]
19. Liu, A.Y.c. The Effect of Oversampling and Undersampling on Classifying Imbalanced Text Datasets. Ph.D. Thesis, The University of Texas at Austin, Austin, TX, USA, 16 August 2004.
20. Negandhi, P.; Trivedi, Y.; Mangrulkar, R. Intrusion detection system using random forest on the NSL-KDD dataset. In *Emerging Research in Computing, Information, Communication and Applications, Proceedings of the ERCICA 2018*; Springer: Singapore, 2019; Volume 2, pp. 519–531.
21. Panigrahi, R.; Borah, S.; Bhoi, A.K.; Ijaz, M.F.; Pramanik, M.; Kumar, Y.; Jhaveri, R.H. A consolidated decision tree-based intrusion detection system for binary and multiclass imbalanced datasets. *Mathematics* **2021**, *9*, 751. [CrossRef]
22. Yin, Y.; Jang-Jaccard, J.; Xu, W.; Singh, A.; Zhu, J.; Sabrina, F.; Kwak, J. IGRF-RFE: A hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 Dataset. *J. Big Data* **2023**, *10*, 15. [CrossRef]
23. Chaithanya, P.; Gauthama Raman, M.; Nivethitha, S.; Seshan, K.; Sriram, V.S. An efficient intrusion detection approach using enhanced random forest and moth-flame optimization technique. In *Computational Intelligence in Pattern Recognition, Proceedings of the CIPR 2019*; Springer: Singapore, 2020; pp. 877–884.
24. Choraś, M.; Pawlicki, M. Intrusion detection approach based on optimised artificial neural network. *Neurocomputing* **2021**, *452*, 705–715. [CrossRef]
25. Hodo, E.; Bellekens, X.; Hamilton, A.; Dubouilh, P.L.; Iorkyase, E.; Tachtatzis, C.; Atkinson, R. Threat analysis of IoT networks using artificial neural network intrusion detection system. In Proceedings of the 2016 International Symposium on Networks, Computers and Communications (ISNCC), Yasmine Hammamet, Tunisia, 11–13 May 2016; pp. 1–6. [CrossRef]
26. Sohail, S.; Fan, Z.; Gu, X.; Sabrina, F. Multi-tiered Artificial Neural Networks model for intrusion detection in smart homes. *Intell. Syst. Appl.* **2022**, *16*, 200152. [CrossRef]
27. Jabbar, M.; Aluvalu, R.; Reddy, S.S.S. RFAODE: A novel ensemble intrusion detection system. *Proc. Comput. Sci.* **2017**, *115*, 226–234. [CrossRef]
28. Jabbar, M.A.; Aluvalu, R.; Reddy, S.S.S. Cluster based ensemble classification for intrusion detection system. In Proceedings of the 9th International Conference on Machine Learning and Computing, Singapore, 24–26 February 2017; pp. 253–257.
29. Gaikwad, D.; Thool, R.C. Intrusion detection system using bagging ensemble method of machine learning. In Proceedings of the 2015 International Conference on Computing Communication Control and Automation, Pune, India, 26–27 February 2015; pp. 291–295.
30. Karthik, M.G.; Krishnan, M.M. Hybrid random forest and synthetic minority over sampling technique for detecting internet of things attacks. *J. Ambient. Intell. Humaniz. Comput.* **2021**, 1–11. [CrossRef]
31. Bej, S.; Davtyan, N.; Wolfien, M.; Nassar, M.; Wolkenhauer, O. LoRAS: An oversampling approach for imbalanced datasets. *Mach. Learn.* **2021**, *110*, 279–301. [CrossRef]
32. Qaddoura, R.; Al-Zoubi, A.M.; Almomani, I.; Faris, H. A Multi-Stage Classification Approach for IoT Intrusion Detection Based on Clustering with Oversampling. *Appl. Sci.* **2021**, *11*, 3022. [CrossRef]
33. Farah, A. Cross Dataset Evaluation for IoT Network Intrusion Detection. Ph.D. Thesis, University of Wisconsin Milwaukee, Milwaukee, WI, USA, December 2020.
34. Krishnan, S.; Neyaz, A.; Liu, Q. IoT Network Attack Detection using Supervised Machine Learning. *Int. J. Artif. Intell. Expert Syst.* **2021**, *10*, 18–32.

35. Grina, F.; Elouedi, Z.; Lefevre, E. Evidential undersampling approach for imbalanced datasets with class-overlapping and noise. In *Modeling Decisions for Artificial Intelligence, Proceedings of the 18th International Conference, MDAI 2021, Umeå, Sweden, 27–30 September 2021*; Springer: Cham, Switzerland, 2021; pp. 181–192.
36. Onan, A. Consensus clustering-based undersampling approach to imbalanced learning. *Sci. Program.* **2019**, *2019*, 5901087. [CrossRef]
37. Tsai, C.F.; Lin, W.C.; Hu, Y.H.; Yao, G.T. Under-sampling class imbalanced datasets by combining clustering analysis and instance selection. *Inf. Sci.* **2019**, *477*, 47–54. [CrossRef]
38. Fan, Z.; Sohail, S.; Sabrina, F.; Gu, X. The Code of Sampling-Based Machine Learning Models for Intrusion Detecion. 2024. Available online: https://github.com/Zongwen-Fan/SamplingML (accessed on 8 April 2024).
39. Imbalanced-Learn Documentation. Available online: https://imbalanced-learn.org/stable/ (accessed on 20 December 2023).
40. Zhou, H.; Zhang, J.; Zhou, Y.; Guo, X.; Ma, Y. A feature selection algorithm of decision tree based on feature weight. *Expert Syst. Appl.* **2021**, *164*, 113842. [CrossRef]
41. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **1958**, *65*, 386–408. [CrossRef] [PubMed]
42. Ho, T.K. Random decision forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 14–16 August 1995; Volume 1, pp. 278–282.
43. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
44. Zhang, Y.; Zhao, Z.; Zheng, J. CatBoost: A new approach for estimating daily reference crop evapotranspiration in arid and semi-arid regions of Northern China. *J. Hydrol.* **2020**, *588*, 125087. [CrossRef]
45. Batista, G.E.; Prati, R.C.; Monard, M.C. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explor. Newsl.* **2004**, *6*, 20–29. [CrossRef]
46. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive dataset for network intrusion detection systems (UNSW-NB15 network dataset). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 10–12 November 2015; pp. 1–6. [CrossRef]
47. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
48. Ullah, I.; Mahmoud, Q. A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks. In *Advances in Artificial Intelligence, Proceedings of the Canadian Conference on AI, Ottawa, ON, Canada, 13–15 May 2020*; Springer: Cham, Switzerland, 2020.
49. Fan, Z.; Gou, J. Predicting body fat using a novel fuzzy-weighted approach optimized by the whale optimization algorithm. *Expert Syst. Appl.* **2023**, *217*, 119558. [CrossRef]
50. Fan, Z.; Gou, J.; Weng, S. A Novel Fuzzy Feature Generation Approach for Happiness Prediction. *IEEE Trans. Emerg. Top. Comput. Intell.* **2024**, *8*, 1595–1608. [CrossRef]
51. McDonnell, K.; Murphy, F.; Sheehan, B.; Masello, L.; Castignani, G. Deep learning in insurance: Accuracy and model interpretability using TabNet. *Expert Syst. Appl.* **2023**, *217*, 119543. [CrossRef]
52. Lim, W.; Yong, K.S.C.; Lau, B.T.; Tan, C.C.L. Future of generative adversarial networks (GAN) for anomaly detection in network security: A review. *Comput. Secur.* **2024**, *139*, 103733. [CrossRef]
53. Liu, L.; Wang, P.; Lin, J.; Liu, L. Intrusion detection of imbalanced network traffic based on machine learning and deep learning. *IEEE Access* **2020**, *9*, 7550–7563. [CrossRef]
54. Pan, Z.; Niu, L.; Zhang, L. UniGAN: Reducing mode collapse in GANs using a uniform generator. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 37690–37703.
55. Kim, J.; Jeong, K.; Choi, H.; Seo, K. GAN-based anomaly detection in imbalance problems. In *Proceedings of the Computer Vision–ECCV 2020 Workshops: Glasgow, UK, 23–28 August 2020*; Springer: Cham, Switzerland, 2020; Part VI, pp. 128–145.