INTRUSION DETECTION IN SOFTWARE DEFINED NETWORKS WITH IMBALANCED ATTACK CLASSES

Sotiris Chatzimiltis[1], Suraj Rohira Lucas[1], Mohammad Shojafar[1], Mahdi Boloursaz Mashhadi[1], Rahim Tafazolli[1]
[1]5GIC & 6GIC, Institute for Communication Systems (ICS), University of Surrey, Guildford, UK

Corresponding author: Sotiris Chatzimiltis, sc02449@surrey.ac.uk

Abstract – *Software Defined Networks (SDNs) have revolutionized the way modern networks are managed and orchestrated. This sophisticated infrastructure can provide numerous benefits but at the same time introduce several security challenges. A centralized controller holds the responsibility of managing the network traffic, thus making it an attractive target to attackers. Intrusion Detection Systems (IDSs) play a crucial role in identifying and addressing security threats within the SDN. In this paper, we developed an SDN-IDS system by utilizing machine learning techniques for anomaly detection to identify deviations in network behavior. This is specifically challenging due to the fact that we only have a few samples from several of the attack classes, i.e. minority classes. Five machine learning algorithms were employed to train the SDN-IDS, and ultimately the most appropriate one was chosen. Moreover, we applied the SMOTE and Tomek link re-samplings on the dataset as well as a cost-sensitive learning technique to enhance the classification performance of the minority attacks. The Decision Tree (DT) model, trained on a feature-reduced and resampled dataset using cost-sensitive learning, achieved an impressive overall performance with 99.87% accuracy and an F1-score of 99.87. Additionally, it demonstrated a classification accuracy above 99% in identifying 11 out of the 15 possible traffic classes.*

Keywords – Imbalanced data, intrusion detection system, machine learning, software defined network.

1. INTRODUCTION

Software Defined Networks (SDNs) are a fundamental technology driving the evolution of modern networks, allowing more efficient and agile management of network resources while allowing the development of advance networking applications. SDNs enable dynamic and centralized control of network resources. By decoupling the network's control plane from the data plane, network administrators can easily manage the entire network infrastructure through software applications rather than configuring individual devices (e.g. routers, switches) one by one [1].

Transitioning from traditional networks to SDNs introduces plenty of advantages such as programmability and automation. Because of the structural design of the SDN, security concerns and potential threats arise within each of its components including the controller, northbound API and southbound API. The controller essentially is the core of an SDN. Gaining access to the controller will enable the intruder to manipulate the network. Further attacks can include Denial of Service (DoS), paralyzing the network, and man in the middle attacks that disrupt the integrity of the communications [2].

To mitigate these security threats, SDNs should employ best practices such as authentication and authorization mechanisms, encryption of communications, traffic monitoring and intrusion and vulnerability detection. Developing an Intrusion Detection System (IDS) can be achieved using different techniques or strategies such as signature-based and anomaly-based ones [3]. Signature-based intrusion detection systems operate by comparing a list of known threats with the packets traversing through the network. On the other hand, anomaly-based IDS utilizes machine learning (ML) techniques to create models that can distinguish between normal and malicious behaviors. This paper employs different machine learning techniques in order to create an anomaly-based intrusion detection system which is able to distinguish normal and malicious network traffic.

Furthermore, network traffic datasets used for IDS are usually mostly comprised of benign data traffic while having only a relatively small fraction of malicious data that indicates security threats or attacks. The dataset utilized in this study was sourced from the International Telecommunication Union (ITU) in collaboration with the industry and consists of real network traffic from users, extracted from an SDN, combined with other known benchmark datasets. It contains around 2.3 million records each with 78 features, with data representing both benign traffic and 14 different types of network attacks, making it well-suited for the evaluation of intrusion detection systems. In our case, benign data comprised more than 80% of the total dataset, whereas certain attacks were significantly underrepresented. For example, only 6 instances represented the heartbleed attack, the SQL injection web attack had only 12 records, and the infiltration attack was represented by 22 instances in the training dataset. This data distribution results in an imbalanced dataset, where the number of

instances in each class (benign versus malicious) is significantly different. In the context of anomaly detection, this imbalance poses a particular challenge as the IDS must accurately identify the rare occurrences of malicious traffic within a predominantly benign traffic environment. Imbalanced data usually leads to a biased model towards the majority class. To address this problem, several techniques can be employed such as resampling the data, the use of appropriate ML techniques such as ensemble methods and cost-sensitive learning. From our perspective to tackle this problem, resampling techniques such as SMOTE [4] and Tomek's link [5] were adopted in order to alleviate data imbalances between classes, ensemble methods such as XGBoost [6] were used, and cost-sensitive learning [7] was applied in an attempt to improve the SDN-IDS performance on classifying minority attacks.

This paper builds on the work of the authors that recently won the "AI/ML in 5G" competition ranking *first* in the "ITU-ML5G-PS-006: Intrusion and Vulnerability Detection in Software-Defined Networks (SDN)" challenge organised by the ITU [8].

Contributions of the paper. The main contributions of the paper are as follows:

1. We propose and implement an SDN-IDS architecture to defend itself from data-traffic attacks that is trained in the application layer and deployed between the the control layer and the infrastructure layer.

2. We present several methodologies to tackle imbalances in data classes, incorporating both SMOTE oversampling and the Tomek link's undersampling techniques, in conjunction with cost-sensitive learning approaches. Our findings demonstrate that employing these techniques enhances the performance of machine learning classifiers in effectively detecting minority attacks.

3. Employing ML methodologies for the SDN-based IDS, incorporating classifiers such as decision trees, random forest and XGBoost.

4. Assessing the effectiveness of the proposed intrusion detection system through an extensive analysis of performance metrics on the benchmark dataset.

The article is structured as follows: Firstly, Section 2 provides an overview of related works. Following that, Section 3 delves into the methodology employed for this project. Section 4 presents a performance analysis for the various experiments conducted and discusses the findings. Lastly, Section 5 summarizes the paper and presents key conclusions.

2. RELATED WORK

The security of SDNs has been extensively explored in the literature. Scott-Hayward et al. [9] examined the new security challenges introduced by SDNs and the need for further exploration of methods to improve their security. Maleh et al. [2] also delved into the potential vulnerabilities, attacks and mitigation strategies in SDN networks. Furthermore, Sultana et al. [3] provided a detailed analysis of the adoption of ML and Deep Learning (DL) techniques to create IDSs for SDNs. IDSs constitute a main defense mechanism in a plethora of environments such as SDNs, the Internet of Things (IoT) and vehicular networks, with ML techniques being the main developing tool.

An extensive literature on ML-based IDSs in the SDN environment exists. Several SDN IDSs based on deep learning have been suggested. Boukria and Guerroumi [10] utilized a feed-forward Deep Neural Network (DNN) to protect the communication channel between the SDN control layer and the SDN infrastructure layer against false data injection attacks. Tang et al. [11] also used DNNs alongside feature selection algorithms to reduce the number of features to create an SDN IDS. ElSayed et al. [12] presented a hybrid IDS that combines Convolutional Neural Networks (CNN) with the SD-Reg regularization technique. Moreover, Ravi et al. [13], adopted a deep learning approach, employing a feature-fused Gated Recurrent Unit (GRU) network to detect and categorize attacks in SDN-enabled Internet of Things (IoT) environments. At the same time, Chatzimiltis et al. [14] created a collaborative SDN-based smart grid IDS utilizing split and federated learning techniques.

In addition, several papers proposed different techniques to create SDN-IDSs to counteract Distributed Denial of Service (DDoS) attacks. Singh et al. [15] proposed a DDoS vehicle-to-infrastructure (V21) ML-based IDS using SDNs. Meti et al. [16] created an IDS to detect DDoS attacks against the SDN controller using Support Vector Machines (SVM) and Neural Networks. In contrast, Chen et al. [17] used the XGBoost ensemble model, in an attempt to demonstrate better results than single ML models. Finally, Haider et al. [18], proposed a deep CNN ensemble framework for efficient DDoS detection.

While the existing literature provides substantial contributions toward improving the security of SDN environments through various IDS approaches, several limitations and gaps remain. Some of the above-mentioned work focuses only on deep learning models without addressing the challenges of data imbalance or feature selection, which can significantly affect the performance of IDSs, particularly in detecting minority attacks. Additionally, some studies primarily focus on DDoS detection, overlooking the broader spectrum of potential threats in SDN environments. In contrast, our work addresses these gaps by leveraging a novel SDN dataset

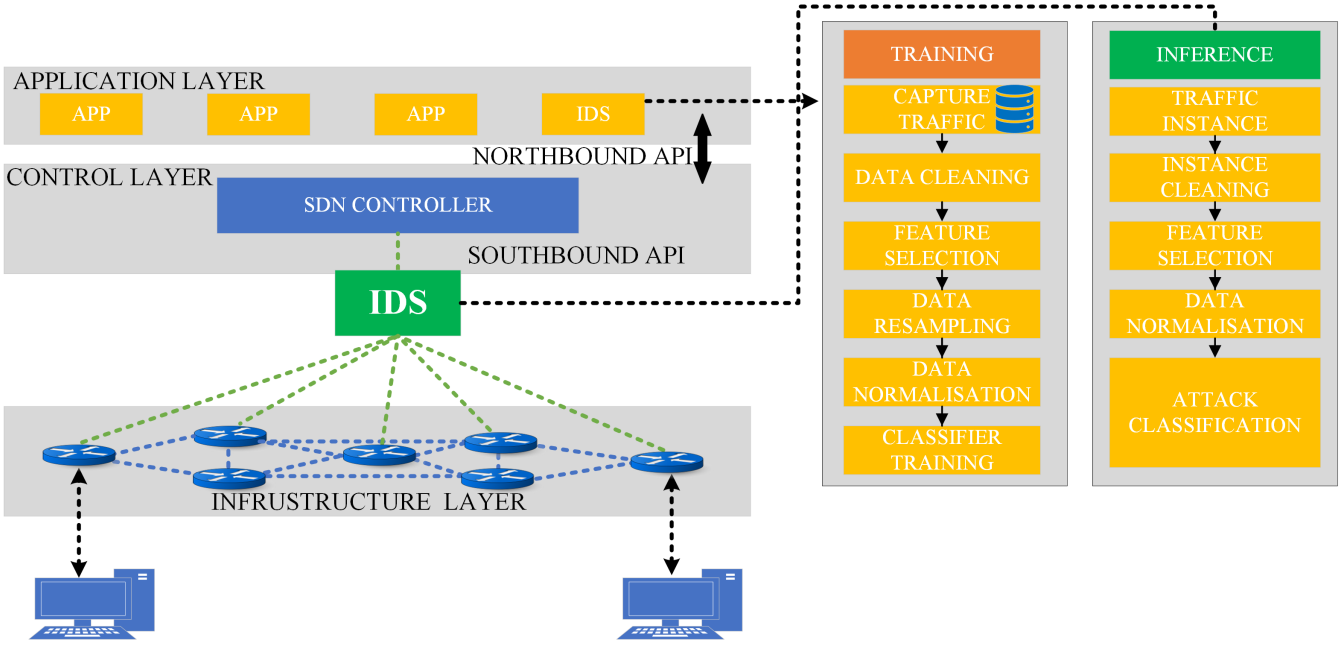


Figure 1 – SDN-IDS architecture.

(Section 4.1), applying several ML models, and incorporating feature selection and data resampling techniques, as detailed in Section 3.2 and Section 3.3, to tackle the issue of data imbalance. This allows for more effective detection across a wider range of attacks, including minority attack types.

3. METHODOLOGY

This section provides an overview of the SDN-IDS architecture and conducts a comprehensive analysis of the individual parts of our proposed intrusion detection system. The source code and dataset of the implementation is available at our GitHub repository[1].

3.1 Proposed SDN-IDS architecture

Figure 1 depicts the proposed architecture for our SDN-IDS. The SDN-IDS undergoes training in the application layer, utilizing the plethora of memory and computational resources, and is positioned between the control and the infrastructure layer of the SDN so inference is conducted closer to the edge of the network. The IDS undergoes training by capturing traffic between end devices and the infrastructure layer. The captured traffic is subsequently preprocessed and a classifier is trained. The trained classifier, in conjunction with the preprocessing pipeline, forms the final IDS prepared for inference. During inference, new traffic from end devices is sent to the IDS for preprocessing. The classifier then determines whether the traffic instance is malicious or benign, triggering alerts for the network manager.

[1] `https://github.com/ITU-AI-ML-in-5G-Challenge/sotirischatzimiltis_sdn_ids`.

3.2 Data preprocessing

3.2.1 Data cleaning

The initial dataset consisted of 78 numerical and categorical features representing various network traffic attributes, such as destination port, flow duration and number of packets forwarded. Additionally, it included a label column indicating whether each instance was benign or malicious, specifying the class of the attack. The first step in data preprocessing involved data cleaning. During this phase, corrupted training and testing instances that contained NaN or infinite values were eliminated. Additionally, duplicate training records were removed to enhance the integrity of the dataset. Subsequently, missing values in both training and testing instances, denoted by -1, were replaced with the mean value of the corresponding training column. For example, if the ”flow duration” column had missing values, these were replaced with the mean flow duration calculated from the training data. In addition, 8 columns with no information (those containing only zero values) were excluded. Furthermore, categorical values in the label column were transformed into numerical equivalents using label encoding. For example, benign traffic was encoded as 0, Bot traffic as 1, DDoS as 2 and so on, for a total of 15 different types of traffic. The complete mapping of traffic types to their numerical equivalents is shown in the first two columns of Table 1.

3.2.2 Feature selection

Feature selection was the second step of the data preprocessing stage. Feature selection helps discarding irrelevant features that negatively affect the performance

Table 1 – Original train and test dataset flow type breakdown.

	Traffic Flow Type	Training Set	Test Set
0	**Benign**	1,337,046	410,865
1	**BOT**	1,228	354
2	**DDoS**	80,653	23,160
3	**DoS GoldenEye**	6,480	1,861
4	**DoS Hulk**	109,334	41,626
5	**DoS Slowhttptest**	3,302	994
6	**DoS slowloris**	3,411	1,048
7	**FTP-Patator**	3,819	1,436
8	**Hearbleed**	6	2
9	**Infiltration**	22	6
10	**Portscan**	66,659	28,728
11	**SSH-Patator**	2,086	1,067
12	**Web Attack Brute Force**	935	272
13	**Web Attack SQL Injection**	12	4
14	**Web Attack XSS**	410	117

of the model. This reduction in feature dimensionality also helps in reducing the overall time taken to train an ML model. In our approach, the features remaining after data cleaning, where non-contributory features were removed, were evaluated based on their importance in classification using a random forest classifier. A 5-fold stratified cross-validation dataset was created in order to determine the importance of the features. The importance of each feature over every fold was averaged. The average importance of each feature was added, in descending order, until 90% of the total importance or the the specific number of features were accumulated. All the features that didn't contribute were then discarded.

3.2.3 Data resampling

The third data preprocessing stage was to handle the data imbalance between the different types of network traffic. Since the training dataset was composed of almost 90% of benign class samples a model could easily overfit during training. There were two main methods of handling data imbalances at this stage. The first scheme was oversampling the minority classes, whereas the second way was undersampling the majority classes. The Synthetic Minority Over-Sampling technique (SMOTE) [4] was used to oversample the minority classes. SMOTE tries to equalize the class distribution in the dataset by creating artificial instances for the minority classes. The number of instances to be created are indicated by the user. Figure 2 illustrates an example of the SMOTE oversampling algorithm with the use of the three nearest neighbors to create the new instance.

For majority-class undersampling two methods were adapted in order to choose the best one. The first method utilized the Tomek-links [5] and the Nearest Neighbors (NN) model, to reduce the size of the initial dataset while trying not to degrade the performance

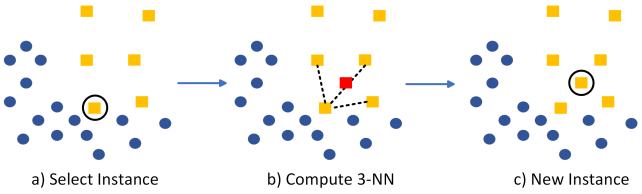


Figure 2 – SMOTE oversampling illustration.

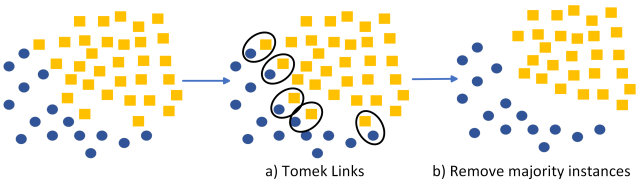


Figure 3 – Tomek links undersampling example.

Algorithm 1 Data Resampling Algorithm

Input: D, $\mathcal{M}$, $\mathcal{S}$, k
Output: D'

for each $m \in \mathcal{M}$ **do**
  mInstances ←D[label] == m
  new_Instances ←SMOTE(mInstances,$\mathcal{S}(m)$,k)
  D.append(new_Instances)
end for
for each $x \in D$ **do**
  y ←TOMEK_Link(x,D)
  if y != null **then** D.remove(y)
  end if
end for

of the model. If two samples of different classes are the nearest neighbors then the sample of the majority class is removed. The second method is called Repeated Edited Nearest Neighbors (RENN) [19] that again uses the NN model and repeatedly applies the edited nearest neighbor rule. The edited nearest neighbor computes the nearest neighbors of every sample that belong to a class to be undersampled. If the majority or all the neighbors of a sample belong to the same class then the sample remains in the set else the sample is discarded. All of the methods used were implemented by [20]. For final experiments the Tomek's technique was used. Figure 3 shows an undersampling example.

Algorithm 1 details how the resampling algorithm works. The algorithm takes as inputs the dataset to be resampled (D), the set of minority classes ($\mathcal{M}$), the number of desired samples for each minority class ($\mathcal{S}$), and the number of nearest neighbors to use (k) to create the resampled instances. Here, $\mathcal{S}(m)$ represents the number of desired instances for the m^{th} minority class. Firstly, SMOTE oversampling is applied for every minority class, creating the desired number of instances. After oversampling the new dataset is undersampled. Finally, the algorithm outputs the new resampled dataset (D').

Table 2 – Performance metrics used for evaluation.

Metric	Equation
Accuracy	$(TP+TN)/(TP+FP+TN+FN)$
Precision	$(TP/(TP+FP))$
Recall	$(TP/(TP+FN))$
F1-score	$(2*Precision*Recall)/(Precision+Recall)$
False Positive Rate	$(FP/(FP+TN))$

3.2.4 Data normalization

The last refinement step involved feature normalization, performed just before the training of each machine learning model. This normalization process utilized the standard scaler, which subtracted the mean of each column and divided by the standard deviation to ensure consistent scaling across features.

3.3 Machine learning models

The selected models for this paper include Decision Trees (DT), Random Forest (RF), and K-Nearest Neighbors (K-NN), an additional bagging classifier and the XGBoost ensemble classifier. These models are favored for their extensive use in IDS, ease of implementation, and support for multiclass classification. Decision trees employ sequential testing to classify instances logically, offering comprehensive interpretations. Bagging classifiers utilize an ensemble of base tree classifiers and use majority voting for final classification, outperforming individual models. K-nearest neighbors classifies new data points based on the most common class among neighboring data points in the feature space. Finally, the XGBoost boosting classifier was adopted, to improve the classification performance.

An additional component that would be beneficial for some ML models (DT & RF) is cost-sensitive learning. Cost-sensitive learning entails assigning varying misclassification costs to different classes, allowing the algorithm to prioritize minimizing the cost associated with misclassifying the minority class. The underlying principle is that instances from the minority class should incur a heavier penalty for misclassification compared to instances from the majority class.

4. RESULTS AND DISCUSSION

This section presents the results of all the experiments performed. Firstly, cross-validation results are analyzed to observe the generalization ability of the ML models used and to select the best performing models to be used for test set evaluation. Subsequently, the selected models were evaluated using the test set. Finally, an ablation study was conducted to analyze the effects of feature selection, data resampling and cost-sensitive learning in the final classification performance.

4.1 Data

For performance comparisons, we adopt the benchmark dataset provided by ULAK for the ITU "ITU-ML5G-PS-006: Intrusion and Vulnerability Detection in Software-Defined Networks (SDN)" challenge [8]. The initial training set was constituted of 1 783 356 records and the initial test set of 512 077 records. Each record had 78 features to begin with and a label column describing the category of the data flow. During the data cleaning phase, 1811 training records and 537 test records were firstly removed. In addition, 166 142 duplicate training records were removed. The training set after data cleaning had 1 615 403 and the new test set 511 540 records. Moreover, all records (training and test sets) with feature values of -1 were replaced with the mean value of the respective training feature column. Finally, during the initial feature analysis (e.g. extraction feature statistics and number unique values), 8 features were dropped since they only contained zero values, providing no useful information. Now each sample had 70 features to be represented with. Table 1 shows the different data flow types and gives a breakdown of how those labels were distributed through the original dataset.

The final dataset used for training the machine learning classifiers was extracted after using the random forest feature importance algorithm and keeping the 35 most important features and was subsequently resampled increasing the number of instances for the minority attacks.

4.2 Performance metrics

Table 2 shows the metrics used to evaluate the performance of an ML model. When dealing with imbalanced data, the accuracy measurement cannot reflect the true performance of an ML technique. A simple classifier can predict all samples to be in the majority class (benign data), thus achieving a high accuracy model but failing to classify instances belonging to minority classes [21] leading to a useless overall model. Metrics such as precision, recall and F1-score were used to better capture the ability of the classifier.

4.3 SDN-IDS performance evaluation

After introducing and analyzing the components of the IDS pipeline in Section 3, the subsequent section pro-

Table 3 – SDN-IDS weighted average performance evaluation for 5-Fold cross-validation using the final dataset.

Model	Precision	Recall	F1-Score	Accuracy
DT	0.9983	0.9983	0.9983	0.9983
RF	0.9981	0.9980	0.9980	0.9980
K-NN	0.9971	0.9971	0.9971	0.9971
Bagging	0.9984	0.9984	0.9984	0.9984
XGBoost	0.9986	0.9986	0.9986	0.9986

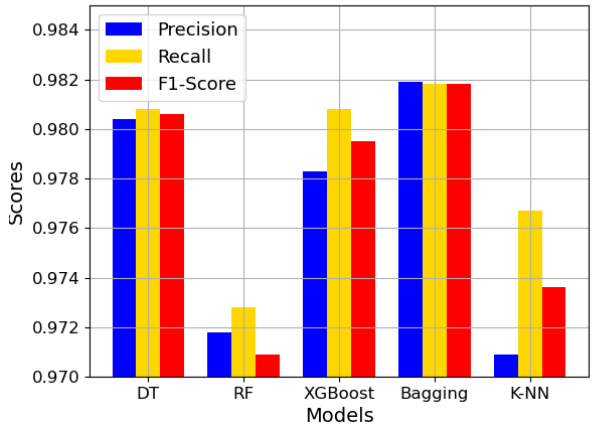


Figure 4 – SDN-IDS macro average performance evaluation for 5-Fold cross-validation using the final dataset.

Table 4 – SDN-IDS weighted Average performance evaluation on the test set.

Model	Precision	Recall	F1-Score	Accuracy
DT	0.9988	0.9987	0.9987	0.9987
RF	0.9988	0.9987	0.9987	0.9987
K-NN	0.9957	0.9954	0.9955	0.9954
Bagging	0.9986	0.9986	0.9986	0.9986
XGBoost	0.9989	0.9989	0.9989	0.9989

vides a comprehensive analysis of the final IDS's performance evaluation. The average 5-fold cross-validation performance for all the five ML models is presented below. Cross-validation results constitute the starting step of the models' performance evaluation, since models are judged for their generalizing ability. Weighted and macro-averages were calculated to assess the model's overall performance, with and without taking into account the class distribution across instances respectively. The macro-average indicates the overall performance across all labels, offering insight into the general quality of predictions for each label. On the other hand, the weighted average provides a performance measure that considers the class distribution, revealing how well the model performs even in cases where the majority of instances are benign.

Figure 4 illustrates the macro-average 5-Fold performance of each ML model trained with the original dataset, while the corresponding weighted performance metrics are presented in Table 3. Additionally, Table 4 displays the weighted average performance evaluation of the test set when the models were trained with the final dataset. It can be inferred that all ML models demonstrated the ability to generalize, exhibiting high F1-scores and accuracies during the 5-fold cross-validation. Notably, the XGBoost classifier achieved the best overall test performance with an F1-score of 99.89 and an accuracy of 99.89%. In contrast, the K-NN model exhibited relatively lower performance, with an F1-score of 99.55 and an accuracy of 99.54%. Due to its suboptimal performance and longer execution time, the K-NN model was excluded from subsequent evaluations.

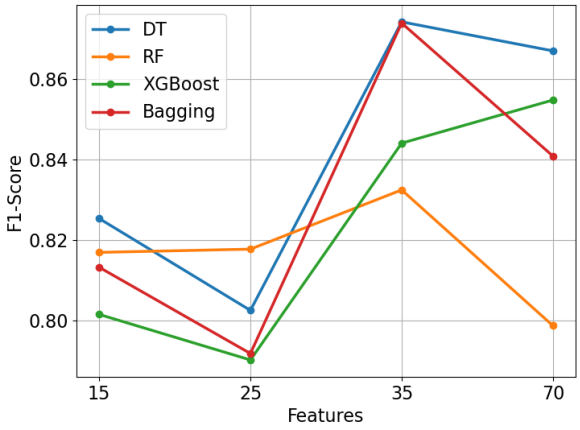


Figure 5 – Macro-average F1-score for every ML model with different numbers of features extracted with random forest feature importance.

Furthermore, tables 5 and 6 offer a thorough overview of the individual performance of each ML model concerning the various minority attacks. The omission of the remaining attacks from the analysis arises from their near-perfect performance achieving accuracy and F1-score of at least 99% and 98.7 respectively. However, accurately classifying certain minority attacks, such as SQL injection and XSS web attacks, remains challenging due to the limited number of training instances and the potential overlap of features between different web attacks.

Ultimately, the FPR for the benign class was computed to ascertain how frequently a malicious instance is inaccurately identified as normal. The false positive rates varied from 0.064% (XGBoost) to 1.31% (K-NN). To illustrate this, the XGBoost model misclassified one attack instance as benign for every 1753 instances, while K-NN made this misclassification every 77 instances.

4.4 Effects of feature selection

As stated in Section 3.2.2, the original dataset underwent further processing to reduce the number of representation features, aiming to decrease training and inference times and exclude features with minimal impact on the classification process. Figure 5 illustrates the macro-average F1-score of every ML model for different numbers of features when extracted using the random forest feature importance method on the original dataset.

The reduction in features from 70 to 35 features when using the feature importance method generally enhances the macro-average performance of all machine learning models, except for XGBoost. XGBoost may experience a slight decline in performance with a reduced number of features, but this decrease is considered negligible. Moreover, the further decrease in the number of fea-

Table 5 – SDN-IDS performance evaluation for minority attacks in the test set.

Model	DT			RF			K-NN			XGB			Bag.		
	Prec.	**Rec.**	**F1-Sc.**	**Prec.**	**Rec.**	**F1-Sc.**	**Prec.**	**Rec.**	**F1-Sc.**	**Prec.**	**Rec.**	**F1-Sc.**	**Prec.**	**Rec.**	**F1-Sc.**
Bot	0.773	0.904	0.833	0.686	0.955	0.798	0.604	0.783	0.681	0.719	0.938	0.814	0.764	0.879	0.817
Heartbleed	1.000	1.000	1.000	1.0000	0.5000	0.667	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500	0.667
Infiltration	0.857	1.000	0.923	1.000	0.500	0.667	0.286	0.333	0.308	1.000	0.500	0.667	0.750	0.500	0.600
Brute Force	0.748	0.732	0.740	0.818	0.529	0.643	0.748	0.577	0.652	0.776	0.702	0.738	0.768	0.743	0.755
SQL Injection	1.000	0.250	0.400	1.000	0.250	0.400	0.022	0.500	0.041	1.000	0.250	0.400	0.500	0.250	0.333
XSS	0.444	0.504	0.472	0.411	0.752	0.532	0.379	0.588	0.462	0.467	0.598	0.524	0.466	0.530	0.496

Table 6 – SDN-IDS Accuracy for the minority attacks

	DT	**RF**	**K-NN**	**XGB**	**Bag.**
Bot	90.40%	95.48%	78.25%	93.79%	87.85%
Heartbleed	100%	50%	100%	100%	50%
Infiltration	100%	50%	33.33%	50%	50%
Brute Force	73.16%	52.94%	57.72%	70.22%	74.26%
SQL Injection	25%	25%	50%	25%	25%
XSS	50.43%	75.21%	58.97%	59.83%	52.99%

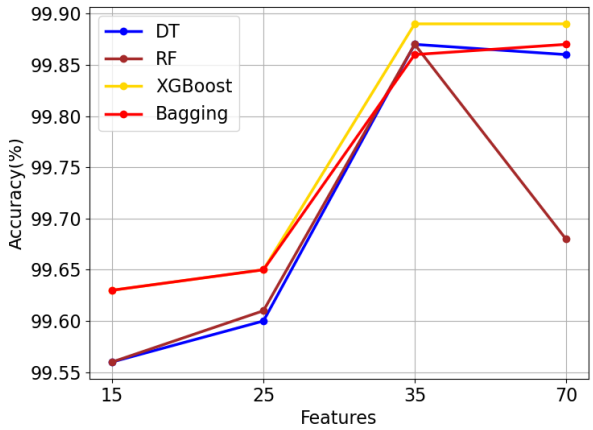
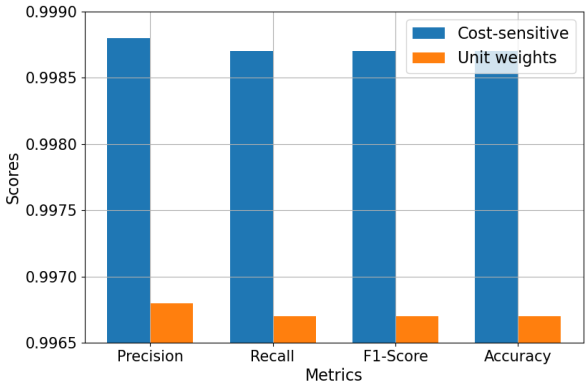


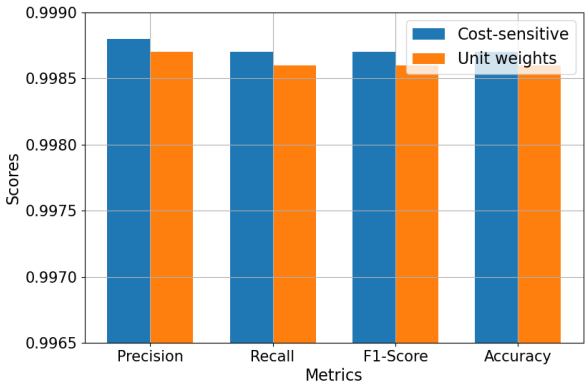
Figure 6 – Accuracy for every ML model when different numbers of features were used for training.

tures, while beneficial in reducing complexity, comes at the expense of performance degradation. Therefore, for this particular problem, the selected optimal number of features was determined to be 35. Finally, it's important to note that the optimal number of features varies across classifiers and feature selection methods.

To assess and contrast the models' effectiveness, Figure 6 demonstrates the accuracy achieved by each model using 70, 35, 25 and 15 features, while the rest of the IDS pipeline remained unchanged. The results clearly indicate that utilizing 35 features yields optimal performance, in our case, either surpassing or nearly matching the performance observed when all features were employed. This underscores the conclusion that employing 35 features is ideal, for our scenario, as it results in a reduced feature space without substantial loss in overall classifier performance.



(a) Performance comparison of DT model with and without cost-sensitive learning.



(b) Performance comparison of RF model with and without cost-sensitive learning.

Figure 7 – Comparison of performance metrics for DT and RF models for cost-sensitive learning.

4.5 Effects of data resampling

This subsection conducts a comprehensive comparison of the resampling techniques introduced in Section 3.2.3. Table 7 outlines the average performance of the models in both resampled and non-resampled scenarios. The results indicate that resampling enhances the average performance of three out of four ML models. Notably, XGBoost is the sole model that exhibits slight superior performance without resampling, achieving an accuracy of 99.90% and an F1-score of 99.90.

Nevertheless, for a more insightful understanding of how

Table 7 – Average performance of models with and without resampling.

	Resampling				No Resampling			
Model	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Accuracy**
DT	**0.9988**	**0.9987**	**0.9987**	**0.9987**	0.9985	0.9985	0.9985	0.9985
RF	**0.9988**	**0.9987**	**0.9987**	**0.9987**	0.9987	0.9985	0.9985	0.9985
XGBoost	0.9989	0.9989	0.9989	0.9989	**0.999**	**0.999**	**0.999**	**0.999**
Bagging	**0.9986**	**0.9986**	**0.9986**	**0.9986**	0.9973	0.9973	0.9973	0.9973

Table 8 – Performance of the minority classes with and without resampling.

	Resampling				No Resampling			
Model	**DT**		**XGBoost**		**DT**		**XGBoost**	
Attacks	**F1-Sc.**	**Acc.**	**F1-Sc.**	**Acc.**	**F1-Sc.**	**Acc.**	**F1-Sc.**	**Acc.**
Bot	**0.833**	**0.904**	0.814	**0.938**	0.830	0.853	**0.852**	0.836
Infiltration	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Heartbleed	0.923	1.000	0.667	0.500	0.923	1.000	0.667	0.500
Brute Force	**0.740**	**0.732**	0.738	0.702	0.728	0.728	**0.769**	**0.812**
Sql Injection	0.400	0.250	0.400	0.250	0.400	0.250	0.400	0.250
XSS	**0.472**	**0.504**	**0.524**	**0.598**	0.407	0.401	0.390	0.333

resampling impacts the classification process, it is essential to examine the alterations in performance on minority classes. Table 8 provides details on the performance of DT and XGBoost models in handling minority attacks, comparing scenarios with and without resampling. In the case of the DT classifier, resampling proves beneficial in enhancing the classification of minority attacks. Resampling helps both classifiers in the classification of XSS attacks, as, prior to resampling, both classifiers exhibited accuracy levels below 50%, indicating performance worse than random guessing. However, the limited number of instances for minority attacks in the test set poses a challenge in the performance evaluation of the classifiers.

4.6 Effects of cost-sensitive learning

Lastly, in this paper, we aim to highlight the impact of cost-sensitive learning on addressing the issue of imbalanced datasets. We specifically focus on cost-sensitive learning within the DT and RF models for the purposes of this analysis. DT seemed to have the highest performance enhancement, in contrast with RF, when cost-sensitive learning was applied as illustrated in Figure 7. The DT model exhibited an improvement in performance, with accuracy increasing from 99.68% to 99.87%. Additionally, the F1-score saw a nearly 0.2 increment, reaching 99.87. In contrast, RF experienced only a marginal increase. Finally, Table 9 showcases that cost-sensitive learning can give a further improvement on the classification performance of minority attacks.

Table 9 – DT performance analysis on minority classes with and without cost-sensitive learning.

	Cost-Sensitive		Unit Weights	
Attacks	**F1-Sc.**	**Acc.**	**F1-Sc.**	**Acc.**
Bot	0.833	0.904	0.794	0.867
Infiltration	1.000	1.000	1.000	1.000
Heartbleed	0.923	1.000	0.600	0.500
Brute Force	0.740	0.732	0.754	0.739
Sql Injection	0.400	0.250	0.286	0.250
XSS	0.472	0.504	0.475	0.530

5. CONCLUSION

This paper introduces an SDN-IDS architecture that integrates anomaly-based intrusion detection techniques with ML models. The system preprocesses data through cleaning, encoding and applying a feature selection algorithm to reduce dimensionality. Following this, data resampling is executed, and normalization is applied before inputting the processed data into ML models. Additionally, the paper conducts a thorough analysis, emphasizing the impact of feature selection, data resampling and cost-sensitive learning on the performance of the ML classifier in the context of minority attacks. Finally, the Decision Tree (DT) model, trained on a feature-reduced and resampled dataset using cost-sensitive learning, achieved an impressive overall performance with 99.87% accuracy and an F1-score of 99.87.

REFERENCES

[1] Gunjan P Tank, Anmol Dixit, Alekhya Vellanki, and D. Annapurna. “Software-Defined Networking The New Norm for Networks”. In: 2012. URL: https://api.semanticscholar.org/CorpusID:53052362.

[2] Yassine Maleh, Youssef Qasmaoui, Khalid El Gholami, Yassine Sadqi, and Soufyane Mounir. "A comprehensive survey on SDN security: threats, mitigations, and future directions". In: *Journal of Reliable Intelligent Environments* 9.2 (Feb. 2022), pp. 201–239. DOI: 10.1007/s40860-022-00171-8. URL: https://doi.org/10.1007/s40860-022-00171-8.

[3] Nasrin Sultana, Naveen Chilamkurti, Wei Peng, and Rabei Alhadad. "Survey on SDN based network intrusion detection system using machine learning approaches". In: *Peer-to-Peer Networking and Applications* 12.2 (Jan. 2018), pp. 493–501. DOI: 10.1007/s12083-017-0630-0. URL: https://doi.org/10.1007/s12083-017-0630-0.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (June 2002), pp. 321–357. DOI: 10.1613/jair.953. URL: https://doi.org/10.1613%2Fjair.953.

[5] Ivan Tomek. "Two Modifications of CNN". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-6.11 (1976), pp. 769–772. DOI: 10.1109/TSMC.1976.4309452.

[6] Sergio González, Salvador García, Javier Del Ser, Lior Rokach, and Francisco Herrera. "A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities". In: *Information Fusion* 64 (2020), pp. 205–237. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2020.07.007. URL: https://www.sciencedirect.com/science/article/pii/S1566253520303195.

[7] Charles X Ling and Victor S Sheng. "Cost-sensitive learning and the class imbalance problem". In: *Encyclopedia of machine learning* 2011 (2008), pp. 231–235.

[8] *Machine Learning for SDN security: improving intrusion and vulnerability detection.* URL: https://challenge.aiforgood.itu.int/match/matchitem/81.

[9] Sandra Scott-Hayward, Gemma O'Callaghan, and Sakir Sezer. "Sdn Security: A Survey". In: *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*. 2013, pp. 1–7. DOI: 10.1109/SDN4FNS.2013.6702553.

[10] Sarra Boukria and Mohamed Guerroumi. "Intrusion detection system for SDN network using deep learning approach". In: *2019 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS)*. Vol. 1. 2019, pp. 1–6. DOI: 10.1109/ICTAACS48474.2019.8988138.

[11] Tuan A Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. "Deep learning approach for Network Intrusion Detection in Software Defined Networking". In: *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*. 2016, pp. 258–263. DOI: 10.1109/WINCOM.2016.7777224.

[12] Mahmoud Said ElSayed, Nhien-An Le-Khac, Marwan Ali Albahar, and Anca Jurcut. "A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique". In: *Journal of Network and Computer Applications* 191 (2021), p. 103160. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2021.103160. URL: https://www.sciencedirect.com/science/article/pii/S1084804521001739.

[13] Vinayakumar Ravi, Rajasekhar Chaganti, and Mamoun Alazab. "Deep Learning Feature Fusion Approach for an Intrusion Detection System in SDN-Based IoT Networks". In: *IEEE Internet of Things Magazine* 5.2 (2022), pp. 24–29. DOI: 10.1109/IOTM.003.2200001.

[14] Sotiris Chatzimiltis, Mohammad Shojafar, Mahdi Boloursaz Mashhadi, and Rahim Tafazolli. "A Collaborative Software Defined Network-Based Smart Grid Intrusion Detection System". In: *IEEE Open Journal of the Communications Society* 5 (2024), pp. 700–711. DOI: 10.1109/OJCOMS.2024.3351088.

[15] Pranav Kumar Singh, Suraj Kumar Jha, Sunit Kumar Nandi, and Sukumar Nandi. "ML-Based Approach to Detect DDoS Attack in V2I Communication Under SDN Architecture". In: *TENCON 2018 - 2018 IEEE Region 10 Conference*. 2018, pp. 0144–0149. DOI: 10.1109/TENCON.2018.8650452.

[16] Nisharani Meti, D G Narayan, and V. P. Baligar. "Detection of distributed denial of service attacks using machine learning algorithms in software defined networks". In: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2017, pp. 1366–1371. DOI: 10.1109/ICACCI.2017.8126031.

[17] Zhuo Chen, Fu Jiang, Yijun Cheng, Xin Gu, Weirong Liu, and Jun Peng. "XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-Based Cloud". In: *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*. 2018, pp. 251–256. DOI: 10.1109/BigComp.2018.00044.

[18] Shahzeb Haider, Adnan Akhunzada, Iqra Mustafa, Tanil Bharat Patel, Amanda Fernandez, Kim-Kwang Raymond Choo, and Javed Iqbal. "A Deep CNN Ensemble Framework for Efficient DDoS Attack Detection in Software Defined Networks". In:

IEEE Access 8 (2020), pp. 53972–53983. DOI: 10.1109/ACCESS.2020.2976908.

[19] "An Experiment with the Edited Nearest-Neighbor Rule". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-6.6 (1976), pp. 448–452. DOI: 10.1109/TSMC.1976.4309523.

[20] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning". In: *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5. URL: http://jmlr.org/papers/v18/16-365.html.

[21] Yuchun Tang, Yan-Qing Zhang, Nitesh V. Chawla, and Sven Krasser. "SVMs Modeling for Highly Imbalanced Classification". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.1 (2009), pp. 281–288. DOI: 10.1109/TSMCB.2008.2002909.

AUTHORS

Sotiris Chatzimiltis received a B.Sc. degree in computer science from the University of Cyprus, Cyprus, in 2021, and an M.Sc. degree in computer vision, machine learning and robotics from the University of Surrey, U.K., in 2022. He is a PhD student at the Institute for Communication Systems at the University of Surrey. His current research interests include machine learning, intrusion detection systems, distributed AI and Open RAN security. He won the ITU challenge on Intrusion and Vulnerability Detection in Software-Defined Networks in 2023.

Suraj Rohira Lucas graduated with a B.Eng in computer and Internet engineering from the University of Surrey, UK, in 2024. His research interests include intrusion detection systems for software-defined networks and the application of machine learning in network security.

Mohammad Shojafar (Senior Member, IEEE) is a Senior Lecturer (Associate Professor) in network security and an Intel Innovator, professional ACM member and ACM distinguished speaker, a fellow of the Higher Education Academy, and a Marie Curie alumnus, working in the 5G & 6G Innovation Centre (5G/6GIC), Institute for Communication Systems (ICS), at the University of Surrey, UK. Before joining 5G/6GIC, he was a senior researcher and a Marie Curie fellow in the SPRITZ Security and Privacy Research group at the University of Padua, Italy. Dr Mohammad secured around £1.9M as PI in various EU/UK projects, including ORAN-TWIN (funded by EPSRC/DSIT CHEDDAR Hub UK;2024), D-XPERT (funded by I-UK/UK;2024), 5G MoDE (funded by DSIT/UK;2023), 5G ONE4HDD (funded by DSIT/UK;2023), TRACE-V2X (funded by EU/MSCA-SE;2023), AUTOTRUST (funded by ESA/EU;2021), PRISENODE (funded by EU/MSCA-IF:2019), and SDN-Sec (funded by Italian Government:2018). He was also COI of various UK/EU projects like HiPER-RAN (funded by DSIT/UK;2023), APTd5G project (funded by EPSRC/UKI-FNI:2022), ESKMARALD (funded by UK/NCSC;2022), GAUChO, S2C and SAMMClouds (funded by Italian Government;2016-2018). He received his PhD in ICT from Sapienza University of Rome, Rome, Italy, in 2016 with an "Excellent" degree. He is an associate editor in *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Intelligent Transportation Systems*, and Computer Networks. For additional information: https://www.surrey.ac.uk/people/mohammad-shojafar

Mahdi Boloursaz Mashhadi (Senior Member, IEEE) is a Lecturer at the 5G/6G Innovation Centre (5G/6GIC) at the Institute for Communication Systems (ICS), University of Surrey (UoS), UK. Prior to joining ICS, he was a post-doctoral research associate at the Intelligent Systems and Networks (ISN) Research Group, Imperial College London, 2019-2021. He received B.S., M.S., and Ph.D. degrees in mobile telecommunications from the Sharif University of Technology (SUT), Tehran, Iran, in 2011, 2013 and 2018, respectively. He was a visiting research associate with the University of Central Florida, Orlando, USA, in 2018, and Queen's University, Ontario, Canada, in 2017. He has more than 40 peer-reviewed publications and patents in the areas of wireless communications, machine learning and signal processing. He received the Best Paper Award from the IEEE EWDTS 2012 conference, and the Exemplary Reviewer Award from the IEEE ComSoc in 2021 and 2022. He has served as a panel judge for the International Telecommunication Union (ITU) to evaluate innovative submissions on applications of AI/ML in 5G and beyond wireless networks since 2021. He is an associate editor for the Springer Nature Wireless Personal Communications Journal.

Rahim Tafazolli (Senior Member, IEEE) is currently a Professor in mobile and personal communications and the Director of the Institute of Communication Systems (ICS) and the 5G and 6G Innovation Centre, University of Surrey. He has been active in research for over 30 years and published more than 1200 research papers. He has been a technical advisor to many mobile companies, and has lectured, chaired and been invited as a keynote speaker to a number of IEE and IEEE workshops and conferences. He served as the chairperson for the EU Expert Group on Mobile Platform (e-mobility SRA), the chairperson for the Post-IP working Group in e-mobility, and the past chairperson of WG3 of WWRF. He is nationally and internationally known in the field of mobile communications. In May 2018, he was appointed as a Regius Professor in electronic engineering for recognition of his exceptional contributions to digital communication technologies over the past 30 years. He was elected as a fellow of the U.K. Royal Academy of Engineering, in 2020. He is a fellow of IET and the Wireless World Research Forum (WWRF).