AN ENSEMBLE APPROACH FOR DNS INTRUSION DETECTION

Karman Singh [a], Ali Mirferdos [b],Smit Patel [c]

[a] University Of Ottawa, Ottawa, Canada, ksing128@uottawa.ca

[b] University Of Ottawa, Ottawa, Canada, smirf045@uottawa.ca

[c] University Of Ottawa, Ottawa, Canada, spate314@uottawa.ca

Abstract

Technological advancements are at their zenith, with ever-increasing data and the technologies being developed to process it. However, security concerns and cyber attacks are burdensome baggage that comes along with it. Large-scale enterprises with access to data of an exorbitant number of users, store it on the cloud or on-site servers. An Intrusion Detection System (IDS) is hence critical for network security. Time after time, machine learning models have been created to combat this problem, yet this field remains a significant problem given the changes in the attacks with every passing day. The goal of this project is to achieve higher accuracy and low false negatives by implementing a combination of models.

The KISTI-IDS-2021 dataset consists of a training and testing set, the testing dataset does not have any labels. After removing the dataset containing more than 90% duplicates was balanced with smitten. The column contained sentence embeddings as word vectors were processed using average pooling and the resultant columns using feature decomposition techniques like PCA and auto-encoder. Further, the combination of baselines has been obtained by implementing an ensemble learning technique called Max Voting. The soft-voting technique's hyperparameters were found using grid-search techniques to learn the weight and assign them to the classifier.

Keywords

Intrusion detection systems, K-fold, Kappa score, Average pooling, max voting, feature decomposition

1. Introduction

In recent days, business disruptions have been fueling the accelerated pace of digital transformation. With these rapid changes, cyber security attacks have further become even more sophisticated. Wherein, network intrusion attacks can not be detected by firewalls alone. Archaic systems would depend on SIDS or Signature-based IDS, which uses pattern-matching techniques to identify attacks. That is, if an intrusion signature matches an existing attack pattern stored in the database, then an intrusion alert will be raised.

The recent advancements in AI have helped identify any type of network intrusion with ease whether it is signature-based, anomaly, host, or network based. However, using the ubiquitously available classification models might not help achieve the desired results. One essential metric to keep in mind while working on this technology is recall which lays more emphasis on false negatives. It is imperative for us to design an IDS which doesn't misclassify an attack given our problem statement. Therefore, to achieve a better-performing model, a combination of 3 different learners has been used for binary classification. Our test dataset does not contain the labels hence we have calculated the Kappa score of each algorithm's results amongst themselves to understand if our weighted hypothesis was true. This is also followed by a K-Fold cross-validation on the original dataset, supporting plots further conclude our results.

The rest of the paper is structured as follows. Section 2 presents the literature review. Section 3 describes the dataset the dataset. Section 4 explains the Methodology followed by Evaluation in section 5. Section 6 draws attention to the Ensemble Model. Finally, the section 7 is a discussion about the result by limitation and advantages in Section 8. Section 9 is about future work and finally conclusion in Section 10.

2. Literature review

The research and work on [1][2] suggest why research and work on IDS is still challenging and needs continual work. It was interesting to also see the metrics used by [3] since this research focuses on parameters like accuracy and precision more than recall. The advantages and disadvantages of using an ensemble of models have been highlighted by Sagi et Al [4] (get from ali paper). The paper also talks about how ensemble models can help avoid overfitting by averaging the results of different models. The soft voting in the max voting technique with different weights as parameters, helps us achieve even more flexibility after reviewing the results of each model. Instead of building an ensemble of simpler models, Khan et. al [5] proposed a two-stage semi-supervised feature learning model with two-

class and multi-class intrusion detection. The model however fell short on the UNSW-NB15 dataset and had an increased fit time. To optimize traditional ensemble algorithms, Kankarajan et al. [6] proposed a greedy randomized adaptive search procedure with Annealed Randomness Forest (GAR) to increase the diversity of NSL-KDD datasets. The authors have employed certain feature selection procedures and addressed both binary and multi-class classification problems in their proposed scheme with 32 and 10 features, respectively. The proposed method outperforms the traditional classification algorithms such as Naive Bayes, Random Forrest, and MLP by achieving 85.06% and 78.90% accuracies in binary and multi-class problems. However, the proposed model is slower at the classification task than RF and NB. Similar to the approach before the fit time was higher and one of the reasons why we have concentrated on fit time as a parameter as well.

Mukherjee et al [7] propose a Naive Bayes Classifier with Feature Reduction developed for Intrusion Detection in this paper. They claim that lowering the number of intrusion detection features may increase security. They evaluated Correlation-based Feature Selection, Information Gain, and Gain Ratio, then proposed the Feature Vitality Based Reduction Method for identifying input features that should be reduced. On the NSL KDD dataset, the naive Bayes classifier. The highest classifier accuracy of the FVBRM approach is 97.78%, with 98.7 TPR for DoS, 97% for normal, 98.8% for probe, 96.1 for r2l, and 64% for u2r. This helped us understand the importance of feature engineering in our dataset as well but given the fact that our dataset contained only 2 features, which will be explained further in section 3. To decrease the features and preserve the information contained in the sentence embeddings; we chose feature decomposition instead of feature selection.

Park et al. [8] performed experiments using Machine Learning's traditional method of Random Forest to classify the types of attacks. They defined 3 types of attacks (IDS, Malware, Shellcode). They used the Kyoto 2006+ dataset which was the latest dataset collected for developing the Intrusion Detection System. 17 features were selected for the experiment, and the results obtained were very interesting. The detection rate for IDS attack was 0.99% however the shellcode attack detection obtained very poor results. The size of the data was not sufficient, and training with the same size class may not be ideal for the machine learning approaches. Also note that the unknown attack class still shows a good performance, F1-score of 0.90, which suggests that there is a distinct pattern for the unknown attack. More intrusion detection techniques were explored also in the paper by Zhang et al. [9] which proposes an effective feature selection approach based on Bayesian Network classifiers and applied in network intrusion detection. By a detailed comparison with other commonly used methods, we find that after feature selection by the proposed approach,

it consumes less time to build classifiers and achieves better performance both in classification accuracy and true positive rates. Moreover, it uses fewer features to predict a high accuracy rate (98.98%).

We also explored papers by Ashfaq et al. [10], Singh et al. [11], Ansari et al. [12], and Cheng et al. [13] which talked about various other techniques for intrusion detection. A summary of each paper's metric and approach have been described in table 1.

Ref	Year	Contribution	Expected Result	Merit	Demerit
[5]	2019	A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection	TSDL model achieves 99.996% on KD99 and 89.134% on UNSW-NB15 test dataset	Lightweight, easy to train, less features, apt for real-time detection	Doesn't work well on complicated data, clearly seen with the difference in accuracy on UNSW-NB15 dataset
[6]	2016	An updated greedy approach for Random Forrest, GAR-Forrest, is used	GAR-Forrest accuracy Binary Classification - 85.06% Multi-class Classification - 78.90%	Employed feature selection techniques upon both binary and multi-class classification problems.	It is slower and does not work with multiple-class classification with more features.
[7]	2012	Using three different feature reduction techniques to limit the features needed and thus improve security. They've used a Naive Bayes Classifier with Feature Reduction	On the NSL KDD dataset, the naive Bayes classifier is evaluated to detect probing, DoS, U2R, and R2L attacks (remote to local). The highest classifier accuracy of the FVBRM approach is 97.78%, with 98.7 TPR for DoS, 97% for normal, 98.8% for probe, 96.1 for r2l, and 64% for u2r.	Identifying different types of attacks. Feature reduction helps in terms of having smaller datasets.	not comparing their results with a baseline and it's not justified why other feature reduction strategies aren't applied.
[8]	2018	Classification of Attack Types for Intrusion Detection Systems using a Machine Learning Algorithm	On the Kyoto 2006+ dataset, the random forest ML technique is expected to detect 3 different types of attacks (IDS, Malware, Shellcode) with utmost accuracy. The detection rate for IDS attacks is 99%	Compared to other commonly used methods, the proposed method in this paper chooses lees features and predicts with high accuracy.	The TPR (True Positive Rate) of U2R attacks is decreased. One possible reason is that the number of U2R records in the NSL-KDD training dataset is too small (only 52). Thus, the feature selection

			using Machine Learning's traditional method Random Forest.		approaches can't learn too much knowledge of their characters.
[9]	2013	An Effective feature selection approach for Network Intrusion Detection	Using Bayesian Network classifier, relevant features are selected from an available set, which is bound to predict with an accuracy rate of 98.98%	Random Forest has a good accuracy rate in detecting unknown attack classes.	The size of data was not sufficient, and training with the same size class may not be ideal for the machine learning approaches. 0.99), the detection of the Shellcode attack shows a poor performance as low as 0.16 of the F1 score. The IDS attack and normal classes did not show good results either.
[10]	2016	A fuzziness-based semi-supervised feed-forward NN is applied to the NSL-KDD dataset for intrusion detection.	Testing accuracy of single hidden layer Neural Network Exp 1 - 82.41% Exp 2 - 84.12%	The proposed method contains a variety of concepts of Dynamic Programming along with the SSL model, namely Neural Networks	Accuracy is low and is similar to classical ML models such as NB.
[11]	2015	Online Sequential Extreme Learning Machine is used on 3 different types of problems of intrusion detection	OS-ELM accuracy Binary Classification - 98.66% Multi-class Classification - 97.67% Kyoto Uni. Dataset - 96.37%	Combined two types of profiling (alpha and beta) with feature selection methods to conduct three different testing experiments to cover all testing cases.	It does not work with the smaller imbalance datasets, and it has been trained with one type of network traffic.
[12]	2021	GRU-based Deep Learning Approach for Network Intrusion Alert Prediction	GRU sequence based model : 78% good predictions with average error of 13%	One of the few papers that works on network intrusion alert prediction and not detection	Low accuracy, tested on Warden data on any security event data. Since model depends heavily on learning long sequences, CNN, LSTM should have been used for comparison

[13]	2021	Discovering Attack Scenarios via Intrusion Alert Correlation Using Graph Convolutional Networks	Alert GCN training accuracy : 95% Test accuracy : 92.42%	Graph convolutional networks outperforms traditional machine learning models.	Not compared with other GNN, no baseline deep learning model to compare with
[14]	2020	An ensemble learning IDS is proposed using GRU, CNN, and random forests. These systems are combined using "OR" logic and majority voting.	87.28% accuracy on the NSL KDD's "KDDTest+" dataset and 76.61% accuracy on the "KDDTest-21" with lower training time and lower needed computational resources	They've used the different capabilities of different kinds of classifiers to increase robustness, feature selection, and sequence analysis. They've used ensemble learning. Taking the case of 0-day attacks to increase the generalization.	The testing was limited and the whole system has a very large load but they didn't include real-time datasets. Also, they've only worked on binary classification and they don't identify the attack's type.
[15]	2020	An autoencoder implementation to detect 0-day attacks.	0-day detection accuracy of [89%-99%] for the NSL KDD dataset and [75%-98%] for the CICIDS 2017 dataset	Taking consideration of high true positive and low false positive rates. They've considered the case of 0-day attacks. Autoencoder achieved better results compared to SVM.	Needs massive datasets. Can't identify attack's type. The comparison doesn't justify why autoencoders are used.

3. Dataset

CDMC 2021 competition is based on the KISTI-IDS-2021 dataset which is a collection of real-world network IDS alerts that contain 25 types of attacks and payload information. The dataset consists of 153,289 alerts which are labeled as malicious(label:1) and benign (label:2). There are two divisions of the dataset: the aforementioned dataset and the test dataset (without labels). The test dataset does not have any labels given that this dataset was from a competition.

Category	Word Vector	Label
0	[[a1, a2,..,a100], [b1,b2,..,b100], [c1,c2,..,c100]]	2
12	[[a1, a2,..,a100], [b1,b2,..,b100], [c1,c2,..,c100], [d1,d2,..,d100]]	1
24	[[a1, a2,..,a100], [b1,b2,..,b100], [c1,c2,..,c100], [d1,d2,..,d100], …., [n1,n2,..,n100]]	2

Figure 2. Dataset description

Features: The dataset has 2 features apart from labels. The first is 'category' and the second is a 'word vector'. The category depicted the type of attack and is an essential feature. The main problem existed with how to process the word vectors column. The column for word vector was hard to interpret because all the words in the payload are embedded as 100 dimensions for each word. That is, each array had n arrays of 100 dimensions each. The dataset with its features has been described in figure 2.

We will first try to convert this word vector into numerical values interpretable to the algorithms. No preprocessing is required on categorical values of 'labels' and 'category'. It is a fairly new dataset and no research was found on this dataset on the day of completion of this report. The dataset despite being balanced had a big problem that we had to tackle, it had 93% of duplicates. More description about how we tackled it has been described in the next section.

4. Methodology

In our project, we first preprocessed the dataset by applying average pooling on the word vector of the sentence embedding. This resulted in 100 new features, essentially of a sentence and hence selecting features based on correlation or any other technique did not make sense. To preserve the information on these features, we decided to use feature decomposition using autoencoders and PCA.

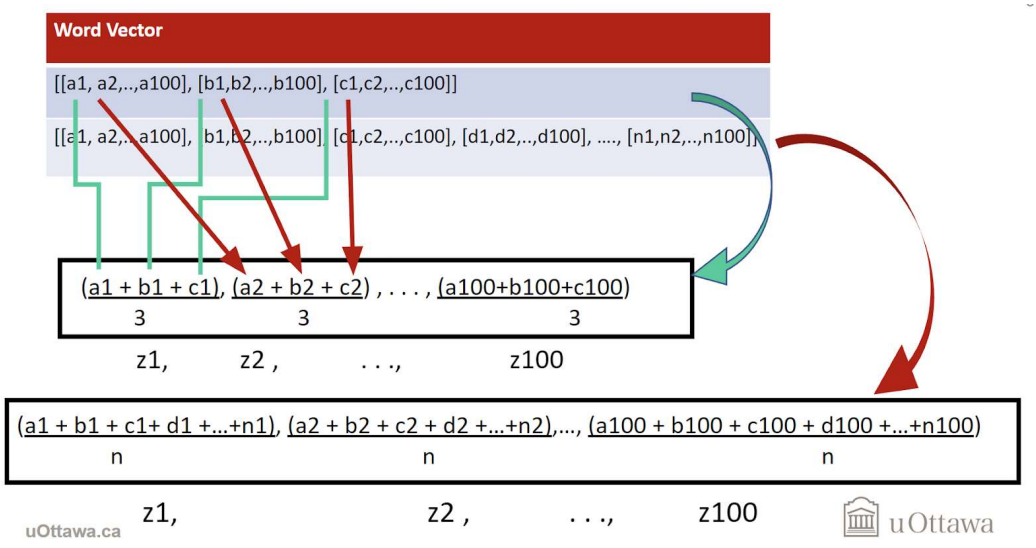


Figure 1. Average Pooling technique

Figure 1 describes the average pooling technique which gives the average of features in a patch. This technique helped us flatten a multidimensional array into different columns and pass them into traditional algorithms.

However, after applying average pooling and checking for duplicates we found that the dataset had 114,881 with surprisingly duplicates not belonging to just the minority class. The duplicates had 55,037 samples with label 2 and 59,844 samples with label 1. A retrospective was done to make sure that the duplicates were not created because of the average pooling technique. We verified that the original data had duplicates. Hence, we decided to apply our baselines on both datasets, one after removing duplicates and up sampling using smote and the other by simply using the original dataset. It should be noted that both the datasets were balanced hence the algorithms chosen for baseline did not have to be tailored to this problem.

4.1. Model Development Procedure

For the given problem, the test dataset was not labeled as it was from CDMC 2021, so the only option left was to split the training dataset. After we have split the training dataset into training and testing sets, supervised binary classification models are trained using the training dataset and evaluated using the testing dataset. This model development procedure included some baseline models and ensemble models. We used cross-validation for the evaluation as the primary evaluation technique to test the ability of the base learners and the ensemble model to predict the new data. It also helped with the flag problems such as overfitting and selection bias and generalized the models to independent datasets. 'Fit Time' was also one of the factors that were considered with cross-validation scores. So, the whole model development process can be summarized as:

The first step in the process was to apply a few base learners on both datasets, the original and the up sampled, to have a definite direction to move forward, which was critical considering the complexity and the challenges in the given problem. It was determined that the base learners produce more accuracy with the up sampled dataset. After that, the best dataset was selected based on various performance evaluation metrics of the base learners, and the optimization process began. In the optimization process, the dimensionality reduction techniques were applied to aggregate the 101 features that are present to avoid the curse of dimensionality. These dimensionality reduction techniques included Principal Component Analysis (PCA) and Auto encoder. The method that worked better with the selected dataset on the base learners was set as a standard.

Following that, we moved forward to the development of the Ensemble model. In this step, various experiments were conducted to get a higher performance than the standard model. The building of an ensemble model included the aggregation of the previous base learners with a voting classifier. Two different voting schemes, hard voting, and soft voting were tested among the voting classifier. After building the ensemble model, the kappa scores were calculated on the test data to check the coherence between the individual models. In the final step, all the models were coss-validated upon each performance metric. Figure 2. shows the graphical representation of the model development process.

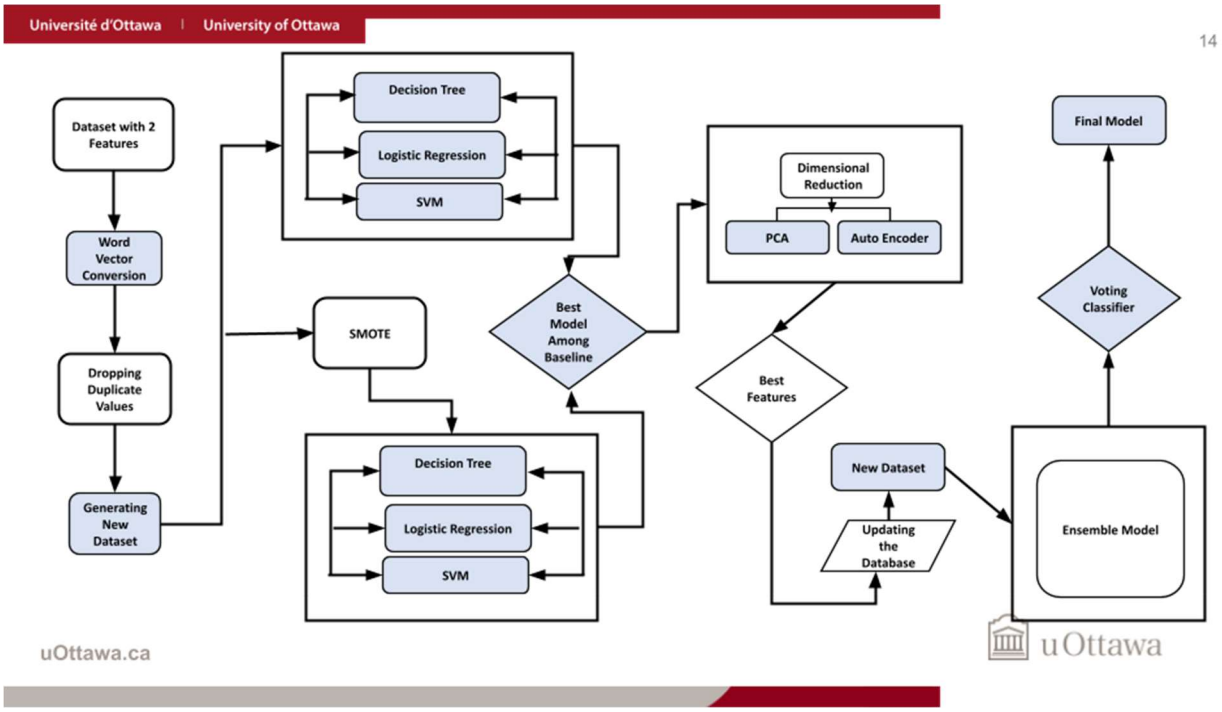


Figure 2. Methodology

4.2. Base Learners

Three models are used as base learners.

1. Decision Tree Classifier
2. Logistic regression
3. Support Vector Machine

A total of 3 base learners are selected and trained on both datasets, original and up sampled. We have used a Support Vector Machine, Decision Tree, and Logistic Regression for our base learners. These base learners are selected

because of their approaches, strengths, and weaknesses in handling a classification problem that we have noticed in the literature review. We have made this choice for base learners to have diversity in the classification approach.

4.2.1. Decision Tree

Decision Tree Classifier was selected because of its ability to use different feature subsets out of a total number of features and decision rules at various stages of classification. Decision Tree classifiers can observe and identify critical information that demonstrates activities such as the intrusion in a network, making them one of the most suitable models for an IDS system. Another reason for choosing the Decision Tree classifier was to have robustness yet higher performance in the classification. Decision Tree is simple and computationally lighter than bagging and boosting approaches such as XGBoost and Random Forest.

4.2.2. Logistic Regression

Logistic Regression (LR) is an efficient algorithm that adds diversity to the base learners that we have selected. Logistic regression models can help find data anomalies, which can be a predictive tool to identify fraud in a network.

4.2.3. Support Vector Machines

The primary goal for selecting Support Vector Machines (SVMs) is related to the fact that it produces higher accuracies when the data are linearly separable. The SVMs result is a separating hyperplane with the highest margins that is calculated by a perpendicular distance of support vectors to the hyperplane. Moreover, SVMs can easily handle linearly inseparable classification problems with ease, and it is less susceptible to outliers in the data. For this project, we have used SVM with a linear kernel.

5. Evaluation

Performance Evaluation is the most significant part of the model-building process. With this process, it can be determined how a particular model performs with a specific set of features and hyperparameters. So, to evaluate the performance of the baseline models, a total of four evaluation metrics were used, including Recall, Accuracy, F1 score, and G-Mean, where Recall is set as the primary evaluator.

The metrics used here are calculated with the help of a confusion matrix. A confusion matrix is an n-by-n matrix drawn using predicted and actual values of a sample. It describes the four different states, such as True Positives, False Negative, False Positive, and True Negative, of the observed samples.

5.1. Accuracy

Accuracy is the ratio of the predictions that are correctly predicted by the classification model and the total number of predictions.

5.2. Recall

Recall measures how many correct predictions were produced out of all the positive instances in the dataset. For an intrusion detection system, Recall is more important as it focuses more on False Negatives. An IDS system with a higher recall score will be more secure as it can identify assaults with a higher likelihood of occurring because the model will focus more on correctly classifying those events that were truly an attack.

5.3. F1 Score

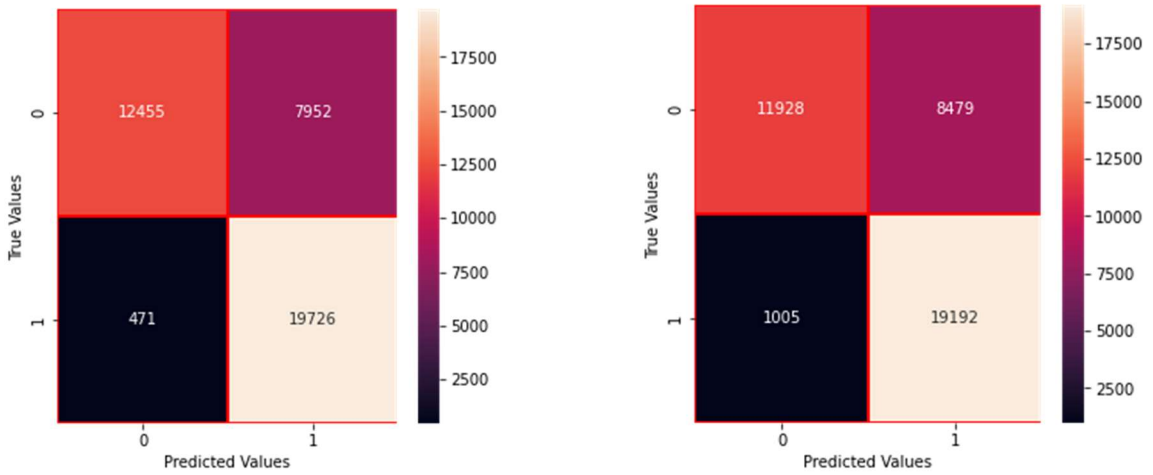
F1 score considers both precision and recall matrices. F1-Score is the harmonic mean of Precision and Recall, ranging from 0 to 1. It is a good indicator of the model's performance.

5.4. Geometric Mean

The geometric mean is calculated to have True-Negatives into consideration. It maps the relation between specificity and sensitivity.

5.5. Confusion matrices of the Baseline Models

These are the confusion matrices of the three baseline models applied on both the original and up sampled dataset.



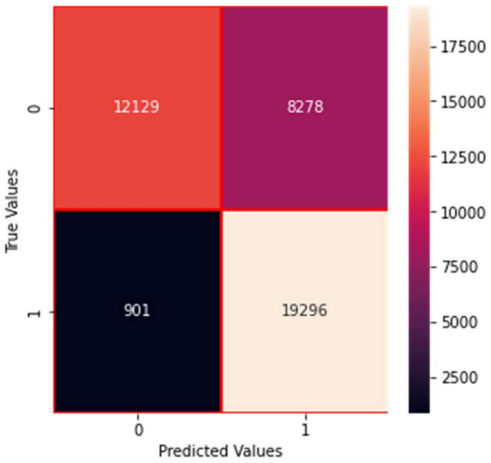


Figure 3. Confusion matrices of baselines on Original Dataset

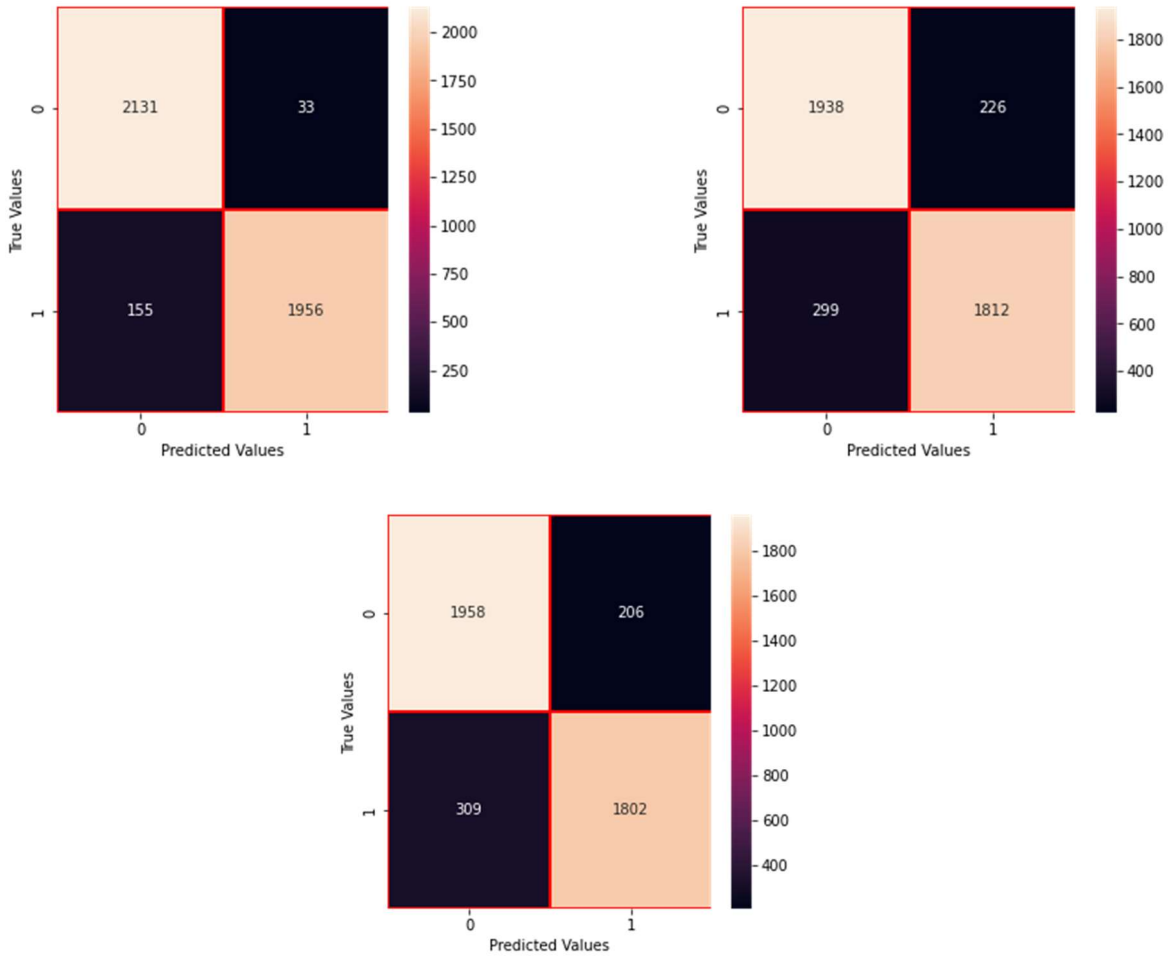


Figure 4. Confusion matrices of baselines applied on up sampled dataset

The above figures consist of three figures of the original dataset applied to the Decision Tree, Logistic Regression, and SVM, and the other three are of the up sampled dataset. Figure 3 shows the confusion matrix of the decision tree algorithm applied to the original dataset, it has a total of 32,181 sampled classified correctly, which is the highest value among the models applied to the original dataset.

Figure 3 displays the results achieved by Logistic regression and SVM, and both models have classified the false negatives similarly. The confusion matrix of the Decision tree with the up sampled dataset is shown in Figure 4, and it has the highest accuracy and recall value of any other model. It classifies 4087 samples correctly out of a total of 4275 samples. The performances of LR and SVM trained with up sampled datasets are shown in Figure 4.

5.6. Performance evaluation comparison of the baseline models

Model	Performance	Original	Up sampled
Decision Tree	Accuracy	79.25%	92.98%
	F1 Score	74.73%	93.11%
	Recall	61.03%	93.80%
	G-Mean	77.20%	95.52%
Logistic Regression	Accuracy	76.64%	88.53%
	F1 Score	71.55%	88.89%
	Recall	58.45%	90.61%
	G-Mean	74.52%	87.67%
SVM	Accuracy	77.39%	89.05%
	F1 Score	72.54%	89.46%
	Recall	59.43%	91.82%
	G-Mean	75.35%	87.88%

Table 3. Performance metrics of Base Learners

Table 3. shows the four different performance parameters calculated for Decision Tree, Logistic Regression, and SVM on both datasets. It can be concluded from Table 3. that on each of the four performance evaluation metrics the up sampled-balanced dataset produces better results on each of the base learners.

On the up sampled dataset, the Decision Tree has a Recall value of 93.80%, which is a 30% higher value than the same model applied to the original dataset. Similarly, the accuracy is nearly 15% higher in the Decision Tree on the up sampled dataset. A similar trend can be observed for the rest of the base learners.

To conclude, we finalized the up sampled dataset for further usage as it produces better results on every model and on every matrix than the original dataset.

5.7. Dimensionality Reduction

Now, in the finalized up sampled dataset, there are 101 columns, including 100 columns extracted from a word vector and one categorical column. As the dimensionality increases, the complexity of the problem also increases. To overcome this problem and to get higher performance with the baseline model, two-dimensionality reduction techniques are applied.

1. Principal Component analysis
2. Auto Encoder

Here, the end goal of this experiment is to come up with the best number of parameters that are aggregated from 101 features with the help of these methods.

5.7.1 PCA

Principal Component Analysis is a technique for analyzing large datasets containing a high number of dimensions and recasting the features along the principal components' axes. So, basically, the PCA aggregates the higher dimensional features and maps them on a newly formed axis. The first principal components will have the maximum variance, and as the number of component axis increases the variance will begin to decrease.

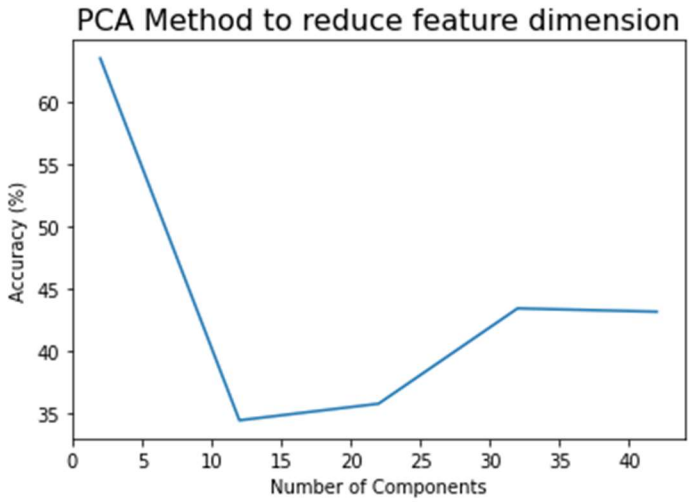


Figure 5. PCA Accuracy

It can be seen from Figure 5 that the highest performance of the PCA was achieved with 2 features, and as the number of features increased the accuracy decreased dramatically and reached below 35. After that, it started increasing gradually when the number of features was 12.

5.7.2. Autoencoder

Autoencoder is a neural network-based algorithm. It reduces the dimensions of the features and focuses more on relevancy, meaning that it gives the priority to the features that have higher relevance to the labels in a supervised learning problem when it comes to prediction. We have included three hidden layers in its structure. The first hidden layer consists of fifty neurons, followed by the layer with hundred neurons and the last layer with fifty neurons. The range of weights and biases that were fed to each prediction node in the auto-encoder is also predetermined. We have set the range of both from zero to four. The range of iterations was set from two to fifty. The accuracy achieved by the autoencoder on the different number of features is shown in Figure 6.

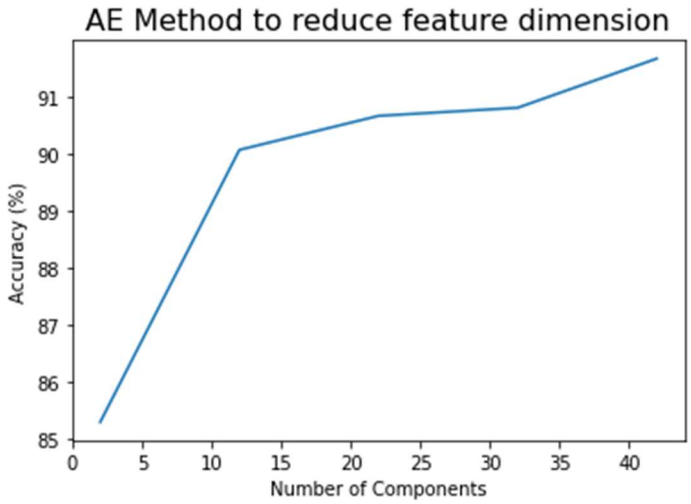


Figure 6. Auto Encoder Accuracy

It is evident from Figure 6. that the lowest accuracy was produced with only 2 features, and as the number of features increased, the accuracy also grew significantly to 12. After increasing the features further from 12, there is no considerable change in the accuracy. So, 12 is the *'elbow point'* of the Number of Components vs the Accuracy plot.

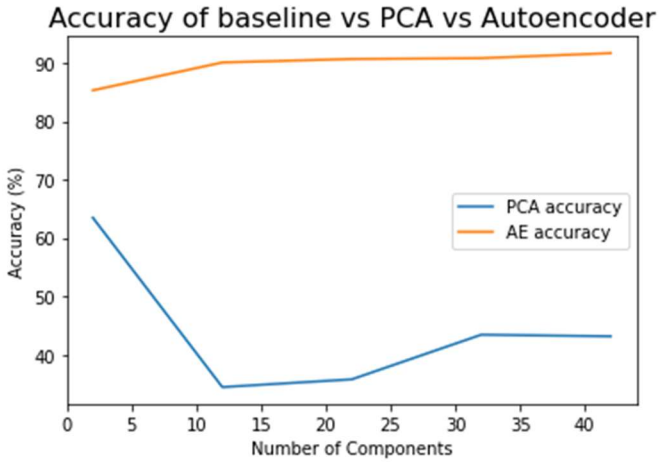


Figure 7. Comparison of PCA and Autoencoder

Figure 7 shows the comparison accuracies produced with PCA and Auto Encoder. The motivation for this selection was because Principal component analysis showed the highest accuracy with the highest accuracy at just 2 features. On the other hand, autoencoders had an elbow point at 12 features with more than 90% accuracy with increasing accuracy with an increase in the number of components.

Hence, we decided to move forward with 12 components aggregated from a total of 101 components by the autoencoder.

6. Ensemble Model

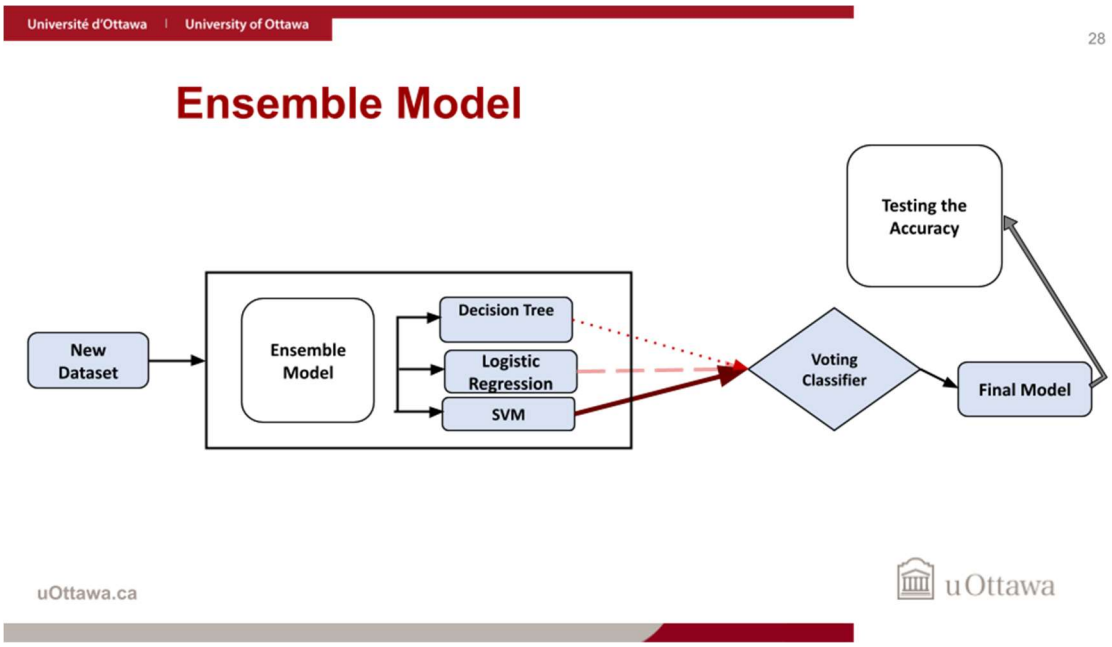


Figure 8. Work-flow of Ensemble Model

6.1. Dataset and Experimental Settings

First, the newly transferred dataset with 12 features was considered and fed to the model. This dataset was up sampled with SMOTE algorithm and consisted of 12 features generated by Auto Encoder. The ensemble model consists of the aggregation of the previous base learners, Decision Tree, Logistic Regression, and Support Vector Machine. The ensemble model works on the techniques of a voting classifier. Two different voting schemes are applied.

1. **Hard voting:** This technique is also known as majority voting, and in this technique, each of the three classifiers votes for a class (either 1 or 0), and the majority wins. In this scheme, each of the classifiers has a similar weightage assigned.
2. **Soft voting:** In this technique, the importance of the prediction result generated by the classifiers is different. This means that each classifier has different weights assigned to its result. We have assigned the weights of 3,2, and 1 to DT, LR, and SVM, respectively.

These weights were assigned according to the performance the base learners produced with the up sampled (transformed dataset). We have found that the ensemble model produces higher results on every performance metric with the soft voting technique than the hard voting. Hence, the soft voting was finalized.

6.2. Performance Evaluation

For the performance evaluation, the same four metrics were selected, Accuracy, Recall, F1- Score, and G-Mean. The accuracy to have the percentage of positive samples predicted out of all the samples. The recall was for the False Negatives to have a more reliable IDS that is less prone to errors. F1-Score, the harmonic mean between Precision and Recall, to consider both metrics. Finally, G-Mean for the sensitivity and Specificity. These metrics were calculated for the ensemble model on the transformed dataset with the help of a confusion matrix in the same way we previously calculated the values of these matrices for the base learners.

6.3. Results

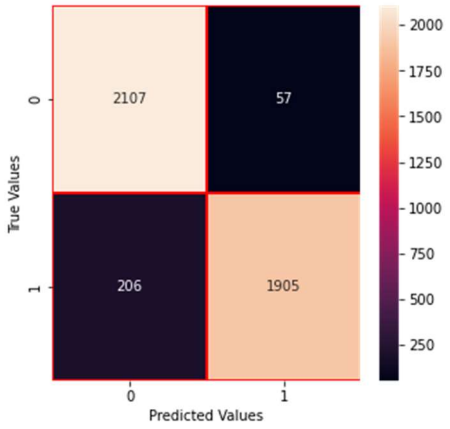


Figure 9. Confusion Matrix of Ensemble model

Figure 9 shows the confusion matrix of the ensemble model applied to the transformed dataset with 12 features. The ensemble model has classified correctly classified 4,012 out of 4,275 samples, which is by far the highest compared to the base learners. The Kappa Score values calculated for the three pairs consisting of the ensemble model and each of the base learners is shown in Table 4.

Performance	Value in %
Accuracy	93.84%
F1-Score	94.12%
Recall	97.36%
G- Mean	93.73%

Table 4. Performance of Ensemble Model

It can be observed from Table 4. that the ensemble model has outperformed Logistic Regression and SVM on all the metrics. The ensemble model has by far the highest Recall value of 97.36%, which is almost 8% higher than the Recall value of both base learners applied to the transformed dataset. The same trend can be observed for the Decision Tree. The ensemble model has an accuracy of 93.84%, whereas the Decision Tree had 92.98%. The Recall value and the value of F1-Score are 1% and 3.5% higher respectively in the ensemble model compared to the Decision Tree. However, for the G-Mean, strangely, the Decision Tree has a 2% higher value with 95.52%. To conclude, the ensemble model has given the best performance on every metric except G-Mean calculated on the Decision Tree base learner.

6.4. Kappa Score

The G-Mean value achieved on the ensemble model with the voting classifier was less than the baseline model. So, we calculated the values of the Kappa Score on the test dataset for each model. The primary reason for this was to check the coherence between the models. Kappa Score can also check which model is underperforming among the aggregated base learners in the ensemble model and let the overall score down.

Model	Kappa Score
Decision Tree Ensemble	0.9990
Logistic Regression Ensemble	0.7686
SVM Ensemble	-0.0660

Table 5. Kappa Scores

Table 5. shows the Kappa Score calculated for each model in the ensemble model. The Decision tree ensemble was the most coherent model with a Kappa Score of 0.999. It has an almost complete agreement with the ensemble model. The Logistic Regression achieved the Kappas Score of 0.7686. These two models were coherent with the overall results achieved by the Ensemble model. However, the SVM has a negative value of Kappa Score, -0.066. It showed that SVM had less agreement or a slight disagreement with the ensemble model.

6.5. Cross Validation Accuracy

The 5-fold cross-validation was performed on each performance metric.

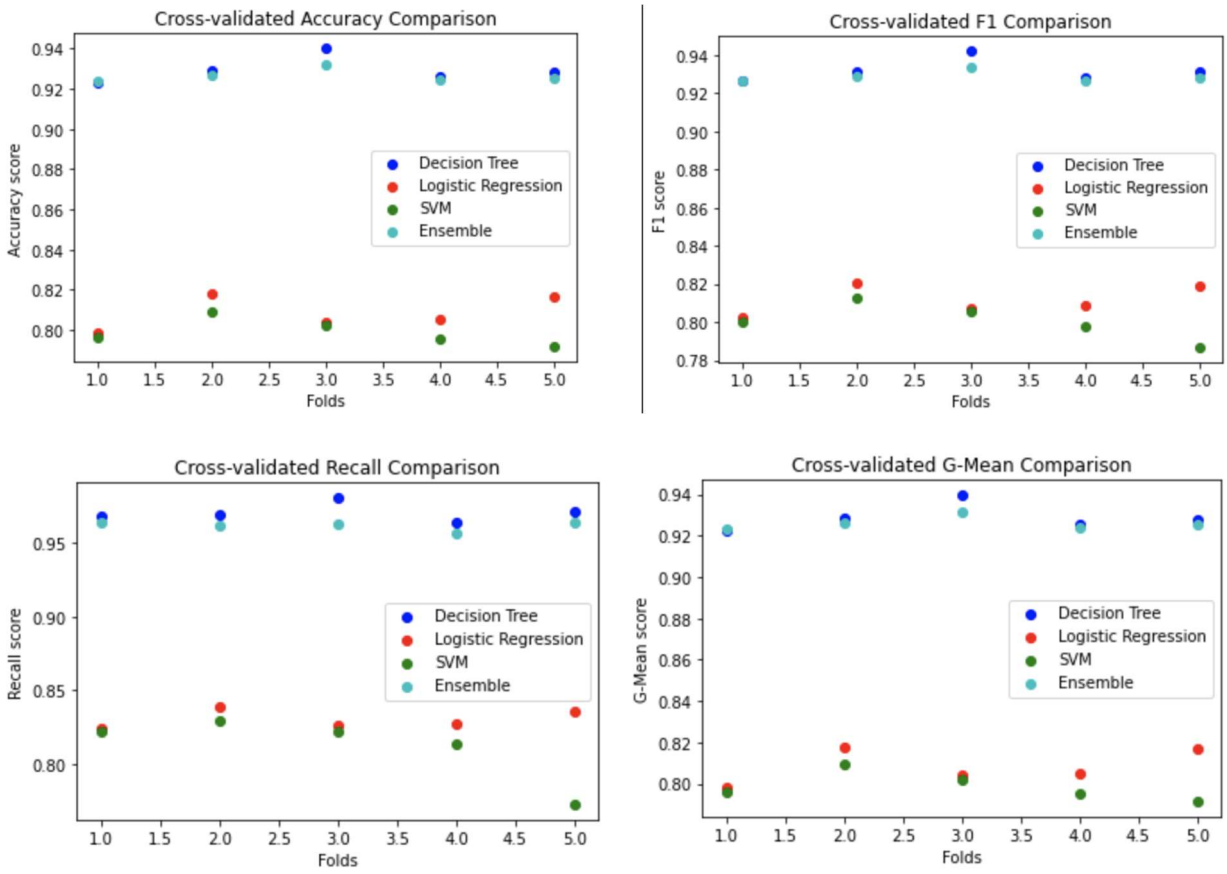


Figure 10. Cross-Validation Accuracies of various Performance Metrics

It can be concluded from Figure 10 that on all four-performance metrics that the accuracy score of the Decision Tree classifier and the ensemble model on each fold was nearly equal. However, in some instances, the accuracy of the

Decision Tree classifier was higher. SVM and Logistic Regression have accuracies far lesser than the Decision Tree and ensemble model.

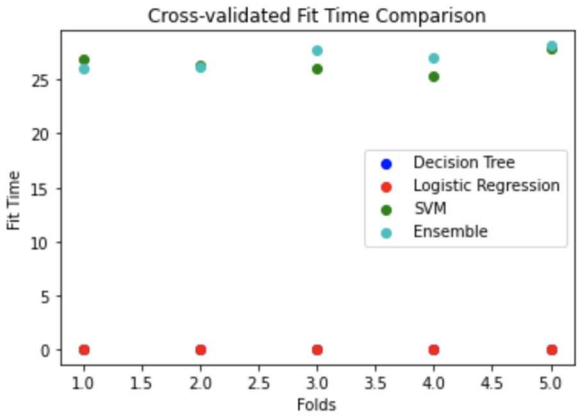


Figure 11. Cross-Validation Accuracies vs. Fit time

Figure 11 shows the fit time on each fold of cross-validation. The SVM and the ensemble model have equal fit times on the second fold. The ensemble model has the highest fit time on the more folds than any other model.

7. Discussion

According to the findings of our study, the performance of a network intrusion detection system can be improved by using an ensemble machine learning technique that uses majority soft voting. This is in comparison to using the individual baselines of logistic regression and support vector machine. However, when contrasted with the findings of each decision tree, the overall consequences were found to be unacceptable. Because a voting classifier utilizes the output that occurs the most often as the output for the final estimate, we presume that the models were not aligned correctly, which caused the results to degrade. The Kappa score validation let us determine that our theory is correct. It was discovered that the SVM model had a somewhat low agreement rate in comparison to the ensemble, but most of the time, the other models agreed with the ensemble.

During the cross-validation process, the average results for the other measures showed only a minor degree of variation; however, the average recollection showed a large degree of variation. Because recall is calculated by dividing the number of true positives by the sum of the true positives and the number of false negatives, a higher recall metric indicates that there were fewer false negatives. In addition, considering the high cost of FNs, this fact is very

pertinent to the process of intrusion detection. In addition to our prior idea about the alignment of the models, we feel that the ensemble model may be more sensitive owing to the multiplicative impact of its numerous models. This is something that we hypothesized before.

Because the performance of the current work may be evaluated from a variety of vantage points, we decided to utilize four different measurements to evaluate it. Since the recall takes into consideration the sensitivity of the model, namely the rate of False Negatives, it was beneficial to examine it independently since it was useful to do so. In comparison, the F1 score strikes a balance between how well you remember and how accurately you remember things. The geometric mean, also known as the G-mean, is an extra statistic that should be taken into account since it is the square root of the product of class-wise sensitivity. The ensemble performed better than the baselines in three out of the four measurements when the data was separated into train and test sets. Despite this, the g-mean came in at a lower value than the decision tree. This is evident in the cross-validation evaluation as well because the performance of the ensemble is worse than that of the decision tree. This assumption is supported by the FNs and FPs included in the confusion matrix. As was said before, we think that this is the result of the fact that the sensitivity of each model is increased when they are merged.

Our study adds to the growing body of information about the use of ensemble learning in the process of detecting intrusions into computer networks. When compared to the use of three distinct baselines, we found that the performance of the models improved when they were merged. This demonstrates that to increase the performance of a detection system, an ensemble approach may be used, provided that the base models are chosen in such a way as to have a better alignment. As a consequence of this, the predictions will be more reliable and accurate. This has great implications for the design and implementation of network security systems, as it implies that ensemble learning may be used to improve the accuracy and resilience of such systems. These implications are significant because they suggest that ensemble learning may be used to improve accuracy and resilience.

8. Limitations and advantages

Our results support the claim that using an ensemble approach with majority soft voting can improve the performance of a machine learning model compared to using the individual base models. However, our results do not support the claim that the ensemble approach always outperforms the individual base models, as we found that the ensemble model did not perform as well as the individual decision tree.

One limitation of our proposal is that the performance of the ensemble model was not consistently better than the individual base models. Additionally, the specific results of our study may not generalize to other datasets or tasks. Another potential limitation of our study is the size of the dataset used, which was relatively small compared to some other studies in this field. This may have affected the generalizability of our results, as the performance of machine learning models can sometimes improve with larger datasets. Additionally, the specific algorithms used in our study (logistic regression, SVM, and decision tree) may not be the most effective for all types of network traffic data.

Compared to the previous work, as the dataset we used was not the same as the benchmarks in the literature, we cannot provide a fair evaluation. Nevertheless, the results are aligned with the conclusions made by Andalib et al. [14] as their ensemble model also suggests that using different types of classifiers in a collaborative approach yields better results. Also using an autoencoder as a feature engineering phase supports the outcomes of Hindy et al. [15]

We also attempted to implement several anomaly detections approaches, as proposed by a large body of prior research. According to Kanakarajan et al. [6], regularised Greedy Forest was implemented. In addition, we used the random forest and iForest algorithms. As the nature of intrusion detection datasets is similar to that of anomaly and outlier detection systems, these methods perform well in this area. As our dataset was unusual in that it might be balanced to be dealt with as a classification system, we anticipated that the findings would not be as satisfactory as those found in the prior research. The initial testing of these procedures validated our assertion. However, this does not apply to all intrusion detection systems since the number of attack samples is far lower than the number of benign samples in the actual world. Through our study, we learned that using an ensemble approach with majority soft voting can improve the performance of a machine learning model compared to using the individual base models. We also discovered that incorporating feature engineering techniques, such as autoencoder, can further improve the performance of the ensemble model. These findings provide new knowledge about the potential benefits of using an ensemble approach and feature engineering in machine learning. Additionally, our experiments highlighted the importance of comparing the performance of different models and considering the specific characteristics of the dataset and task when choosing a model.

9. Future Work

In further research, it may be possible to investigate whether other approaches or larger data sets provide distinct results. Exploring additional ensemble learning methodologies, such as bagging or boosting, to improve the performance of the system may potentially have a significant impact on the results. To determine the robustness and generalizability of the system, it is necessary to test it on datasets that are both larger and more diverse. It is necessary to test the system's performance in real-world circumstances and compare it to other commercially available intrusion detection systems to determine whether or not it is practicable to use and whether or not it is effective. Our dataset has a very low number of different types of features arranged in a variety of categories. Consider integrating other components into the system, such as data on network traffic, to enhance its accuracy and make it capable of detecting a wider variety of intrusions. This would enable the system to protect the network. During the era of feature engineering, our only tools were principal component analysis and autoencoders, neither of which can be interpreted. When put into practice in the real world, a true system is required to produce results that can be explained. To grasp and correctly interpret the findings produced by the ensemble system, we have to investigate the usage of AI techniques that can be explained, such as methods of attribution or feature importance. The dataset preprocessing can be explored further by processing the word vector using strategies like max-pooling, padding, etc. Also, it’s important to consider tuning the weights that determine how the models are combined.

10. Conclusion

In conclusion, our research highlighted the potential advantages of employing an ensemble machine learning technique with majority soft voting to enhance the performance of a network intrusion detection system. The ensemble model outperformed the individual baselines of logistic regression and support vector machine, but not the decision tree. With 93% duplicates, the baselines helped us justify dropping the duplicates and up sampling the data ourselves. We also investigated the use of feature engineering approaches, such as autoencoder, and found that they enhanced the ensemble model's performance. We used autoencoder to further decompose the 101 features to only 12 components. The weights of the voting mechanism were also tuned by using grid-search. These results shed fresh light on the possible benefits of employing an ensemble method and feature engineering in machine learning and emphasize the necessity of evaluating the dataset and task-specific properties when selecting a model.

References

[1]C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, ‘‘A survey of intrusion detection techniques in cloud,’’ J. Netw. Comput. Appl., vol. 36, no. 1, pp. 42–57, 2013, doi: 10.1016/j.jnca.2012.05.003.

[2] R. Zuech, T. M. Khoshgoftaar, and R. Wald, ‘‘Intrusion detection and big heterogeneous data: A survey,’’ J. Big Data, vol. 2, no. 1, p. 3, 2015, DOI: 10.1186/s40537-015-0013-4.

[3] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, ‘‘Intrusion detection techniques in cloud environment: A survey,’’ J. Netw. Comput. Appl., vol. 77, pp. 18–47, Jan. 2017, doi: 10.1016/j.jnca.2016.10.015.

[4]Sagi, O., Rokach, L.: Ensemble learning: A survey. WIREs Data Mining and Knowledge Discovery 8(4) (Jul 2018). https://doi.org/10.1002/widm.1249

[5] Khan, F. A., Gumaei, A., Derhab, A., & Hussain, A. (2019). TSDL: A two-stage deep learning model for efficient network intrusion detection. IEEE Access, 7, 30373–30385.

[6] Kanakarajan, Navaneeth Kumar, and Kandasamy Muniasamy. "Improving the accuracy of intrusion detection using gar-forest with feature selection." Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015. Springer, New Delhi, 2016.

[7] Mukherjee, Saurabh, and Neelam Sharma. "Intrusion detection using naive Bayes classifier with feature reduction." Procedia Technology 4 (2012): 119-128.

[8] Kinam Park, Youngrok Song, Yun-Gyung Cheong. “Classification of Attack Types for Intrusion Detection Systems using a Machine Learning Algorithm”. School of Software Sungkyunkwan University Suwon-si, South Korea. 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications.

[9] Fengli Zhang, Dan Wang. “An Effective Feature Selection Approach for Network Intrusion Detection”. School of Computer Science & Engineering University of Electronic Science and Technology of China. 2013 IEEE Eighth International Conference on Networking, Architecture and Storage

[10] Ashfaq RA, Wang XZ, Huang JZ, Abbas H, He YL. Fuzziness based semi-supervised learning approach for the intrusion detection system. Information sciences. 2017 Feb 1;378:484-97.

[11] Singh, Raman, Harish Kumar, and R. K. Singla. "An intrusion detection system using network traffic profiling and online sequential extreme learning machine." Expert Systems with Applications 42.22 (2015): 8609-8624.

[12] Ansari, M. S., Bartoš, V., & Lee, B. (2022). GRU-based deep learning approach for network Intrusion Alert Prediction. Future Generation Computer Systems, 128, 235–247.

[13] Cheng, Q., Wu, C., & Zhou, S. (2021). Discovering attack scenarios via intrusion alert correlation using graph convolutional networks. IEEE Communications Letters, 25(5), 1564–1567.

[14] Andalib, Amir, and Vahid Tabataba Vakili. "An autonomous intrusion detection system using an ensemble of advanced learners." 2020 28th Iranian conference on electrical engineering (ICEE). IEEE, 2020.

[15] Hindy, Hanan, Robert Atkinson, Christos Tachtatzis, Jean-Noël Colin, Ethan Bayne, and Xavier Bellekens. "Utilising deep learning techniques for effective zero-day attack detection." Electronics 9, no. 10 (2020): 1684