

# Evaluation of network intrusion detection with machine learning and deep learning using ensemble methods on CICIDS-2017 dataset

Lanka Rakesh  
Department of CSE  
Manipal University Jaipur  
Jaipur, India  
rakesh.lanka6733@gmail.com

Lav Upadhyay  
Department of CSE  
Manipal University Jaipur  
Jaipur, India  
lavupadhyay@gmail.com

Pochamreddy Mukesh Reddy  
Department of CCE  
Manipal University Jaipur  
Jaipur, India  
mukeshreddy662369@gmail.com

**Abstract**—An intrusion detection system (IDS), also referred to as an IDS, is a type of network security equipment that monitors network communications in real-time and, should any potentially hostile transmissions be identified, either sends out alarms or implements active reaction measures. In this context, numerous researchers have attempted to improve intrusion detection performance by combining traditional machine learning models with alternative optimization techniques. Although the present intrusion detection model has the potential to considerably improve performance, there are still certain persistent problems, such as faulty detection and data preparation operations, which both have the potential to reduce accuracy. Using the CICIDS2017 dataset, we provided an analysis model of various classifiers in our paper, including random forest, stacking set-1 (Naive Bayes, K-Nearest Neighbours, Random Forest), bagging, boosting (XGB), LSTM, CNN, and stacking set-2 (LSTM and CNN). Boosting (XGB) performed better than all models across all feature sets overall.

**Index Terms**—Cyber Security Machine Learning, IDS, Feature Selection

## I. INTRODUCTION

We need to make our online systems more secure because there are more and more threats to their safety. There have been more cases of hackers breaking into systems and stealing sensitive information. An intrusion or attack happens when someone sends harmful data from a hacked computer or server. These attacks happen because there are already problems with the system, like mistakes by users, mistakes in how it's set up, or mistakes in the software. When all these problems come together, it can lead to complicated intrusions.

Creating a reliable security system is important because threats to networked systems are increasing in number and becoming more complicated. To build effective intrusion detection systems (IDS), we need new strategies. Machine learning techniques, like statistics and machine learning, can help us predict network attacks.

IDS has trouble with computing costs, the false positive rate (FPR), and the false negative rate (FNR) despite the integration of machine learning. The IDS's projected speed and precision are hampered by these issues. Processes for feature extraction and selection might be difficult due to

the abundance of alternatives. High-dimensional data with irrelevant or duplicate information might impair performance, reducing the accuracy and overall effectiveness of network intrusion detection systems (NIDS).

In order to overcome these difficulties, pre-processing techniques called feature selection methods have been developed. By identifying crucial features, lowering computing load, and boosting data processing, feature selection seeks to increase NIDS performance. It has been observed that a smaller number of features can frequently still produce appropriate classification results, although it is important to avoid leaving out important predictors.

However, relying solely on one feature selection approach might exclude valuable features. Combining multiple methods could yield accurate classification predictors. Different feature selection algorithms have unique evaluation criteria, which may lead to inconsistencies in feature selection and, subsequently, reduced prediction accuracy.

This study's main goal was to suggest a unique NIDS framework that keeps pertinent candidate characteristics in mind when making feature selections. The diversity of features may improve selection consistency and prediction accuracy. Eight files were initially divided into assault categories, and 20 percent of each group was combined. Due to its adaptability for huge datasets, Recursive Feature Elimination (RFE), an iterative backward selection procedure, was used. It enhances the generalizability and interpretability of models.

In this study, we developed a novel method for NIDS, used RFE for feature selection, and compared the effectiveness of various deep learning and machine learning models. Testing was conducted using random forest, stacking, bagging, and boosting. By improving precision and effectiveness, ensuring real-time detection, scalability, and adaptation to changing threats, these contributions considerably advance intrusion detection systems. The methods and results of the research show potential for enhancing cybersecurity, developing more reliable intrusion detection systems, and advancing the discipline.

## II. RELATED WORK

In the area of network intrusion detection systems (NIDS), numerous notable contributions and related research have shed a lack of information on practical methods to improve detection precision and system efficiency. In 2019, Ali et al. investigated the potential and usefulness of the Random Forest algorithm in developing reliable intrusion detection models. Their findings emphasized how crucial careful feature selection is to enhancing NIDS performance. Zhang et al. (2018) investigated the effectiveness of ensemble approaches for intrusion detection, including stacking, bagging, and boosting [1]. In particular for managing unbalanced datasets, their study demonstrated the advantages of integrating multiple classifiers to improve overall detection capabilities. With a focus on boosting-based methods, Li et al. (2020) advanced the discipline by emphasizing how well they increase detection rates while reducing false positives. To attain the best results, it should be noted that the significance of educated feature selection practices was highlighted [2]. Alkinani et al. (2021) proposed an integrated framework that combined the benefits of Random Forest, Support Vector Machines, and feature selection. Hybrid models were also investigated. Through a variety of approaches, these works together expand the field of intrusion detection, paving the way for novel and flexible NIDS systems. While the Recursive Feature Elimination (RFE) approach is used in your research and models like Random Forest, Stacking, Bagging and Boostings are evaluated, it follows this trend of utilizing cutting-edge methods to strengthen cybersecurity defenses and build robust intrusion detection systems [3].

## III. PROPOSED METHODOLOGY

### A. Dataset

The CICIDS dataset, which consists of 8 CSV files, was the dataset used in this investigation. It is a sizable dataset that includes a range of assault types. Our first step was to eliminate any duplicates from the eight CSV files because there were a lot of them, which led to the formation of a new dataset. This non-redundant dataset has a substantial number of cases in total. The extraction of about 20 percent of the non-redundant dataset was our main goal. We chose about 20 percent of each assault type to do this. A new integrated dataset was built by combining these subsets.

### B. Dataset Memory Reduction

All features—aside from the label feature—are in the float64 data type after normalization and label encoding, whereas the label feature is in the int64 data type. This dataframe uses 345.3 MB of RAM. We transformed all features—aside from the label feature—to the float32 data type and the label feature to the int8 data type in order to shrink the dataframe size without rounding off the values in the features. The new dataframe size requires 155.3 MB of RAM.

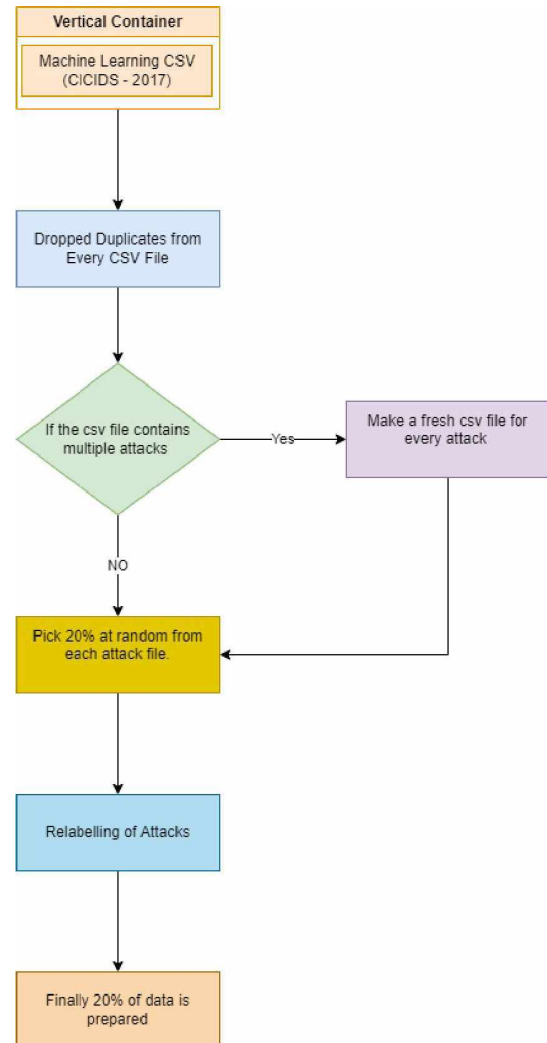


Fig. 1. Dataset Preparation

### C. Training and Splitting of Dataset

We split the dataset into 70% for training and 30% for testing.

### D. Data Pre-Processing

We first verified our dataset for infinite and null values during the data preprocessing stage and removed them. Next, we normalized all characteristics with the exception of the label feature. For categorical encoding for the label feature, we used Label Encoder.

### E. Dataset Memory Reduction

We did memory reduction on the dataset to reduce training time and improve memory use. All features except the label feature were in the float64 data type after normalization and label encoding, and the label feature was in the int64 data type. This data frame used 345.3 MB of memory. Our objective was to shrink the data frame without rounding the feature values. To do this, we changed the data types of all features—aside

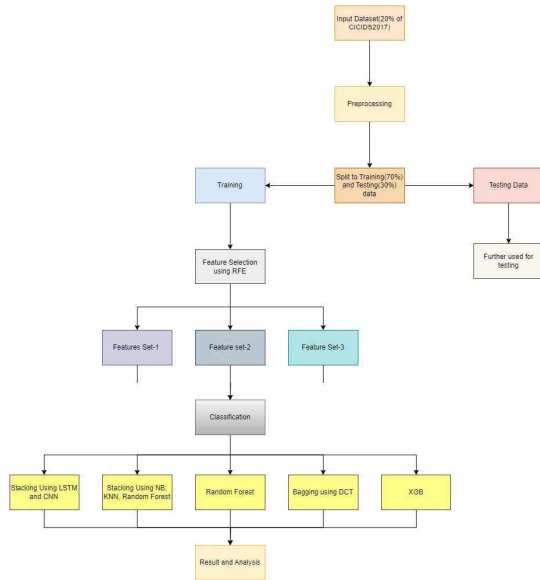


Fig. 2. Experimental Design

from the label feature—to float32 and int8, respectively. As a result, the new DataFrame used only 155.3 MB of memory.

#### F. Feature Selection

A critical phase in the machine learning process, feature selection has a significant impact on model performance. There are many different ways to choose features, including filter methods, wrapper methods, embedding methods, and dimensionality reduction techniques. To handle our dataset, we chose the wrapper strategy.

We deliberately selected the Random Forest algorithm-based Recursive Feature Elimination (RFE) model. According to the relevance scores assigned to each feature, the RFE wrapper method iteratively removes the least significant characteristics. Iteratively reducing the number of features until the target number is obtained entails training a model on the complete feature set, calculating feature importance scores, and removing the least significant features.

The chi-square test, variance threshold, graph plotting with random forest, mutual information classifier, and a variety of feature selection strategies were also tested. Of these, RFE produced the best outcomes. RFE was demonstrated to be computationally effective and suitable for processing high-dimensional data given the size of our dataset (about 500,000 values).

RFE not only chooses the most advantageous collection of traits but also rates their significance. Understanding the data and locating the most pertinent elements for the issue are made easier thanks to this rating. Greater model performance by decreasing overfitting, greater interpretability, and effective use of computational resources are only a few advantages of feature selection using RFE [4].

In order to improve model generalization, reduce dimensionality for quicker training and resource efficiency, improve

interpretability for understanding model predictions, and boost robustness by removing noise or unimportant features, feature selection was pursued.

## IV. MODELS IMPLEMENTED

### A. Random Forest

In order to improve classification and regression tasks, the ensemble learning technique Random Forest integrates many decision trees. With the introduction of randomness and diversity, each decision tree is built using a subset of the data and a subset of the characteristics. Using either a majority vote (for classification) or an average (for regression), the combined predictions of the different trees are used to produce the final prediction. By minimizing bias and variance trade-offs, Random Forest improves generalization while reducing overfitting [5].

### B. Stacking

An ensemble learning technique called stacking, or stacked generalization, combines several base models by training a meta-model on their forecasts. Predictions made for the input data by base models are then used as features in the meta-model. By combining the advantages of various models, stacking teaches the meta-model when to trust the forecasts of each base model. This method may result in more accurate and robust predictions [6].

### C. Bagging

The ensemble technique known as bagging, or bootstrap aggregating involves training several instances of the same base model on various subsets of the training data. Each model is trained separately, and the results are then combined to create the final product. Bagging is especially useful for models that are sensitive to changes in the training data since it lowers variance and improves model stability [7].

### D. Boosting

Another ensemble method called boosting combines a number of weak learners to produce a powerful predictive model. In contrast to bagging, boosting develops models one at a time, with each model concentrating on the problems that the prior model had. Forecasts from all models are merged, and the forecasts of models with higher performance are given more weight. Boosting is particularly effective in enhancing the performance of weak models because it adapts over iterations to increase accuracy [8].

## V. RESULT AND ANALYSIS

### A. Stacking set-1 (Using NB, KNN, and Random Forest)

Here, we used the stacking ensemble method to merge the three models: Naive Bayes, K-Nearest Neighbors, and Random Forest. The 24 features of the stacking model function well, with an accuracy of 99.97%. Performance is equivalent for all 24 features and for ALL features.

The false alarm rate (FAR) for 24 features is 0.00021, while the FAR for all features is 0.00023. For 24 and all feature

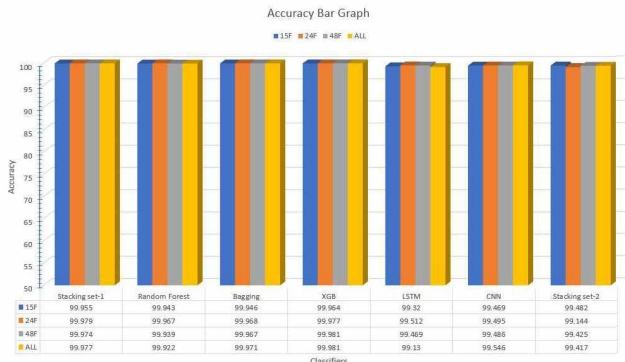


Fig. 3. Accuracy Bar Graph

sets, the FAR is low with little variation. You can choose the 24-feature set if you prefer better performance with fewer features.

### B. Stacking set-2 (Using LSTM and CNN)

Here, we are combining CNN and LSTM. When compared to other feature sets, the 24 features from stacking set 2 perform poorly with an accuracy of 99.144%. And when compared to other features, the 15-feature set is performing quite well, with an accuracy of 99.482%.

From fig 3 we can observe that stacking of LSTM and CNN improved the performance of 15 feature set when compared to individual LSTM and CNN models. Individual models of LSTM and CNN performed well compared with stacking of LSTM and CNN for 24 and 48 feature sets, respectively. For all feature set, the stacking of LSTM and CNN performed well compared to LSTM but failed to perform compared with CNN. The FAR is high(0.575%) for 24 features set and low(0.86%) for 15 features set. Overall the stacking performs well for 15 feature set [9].

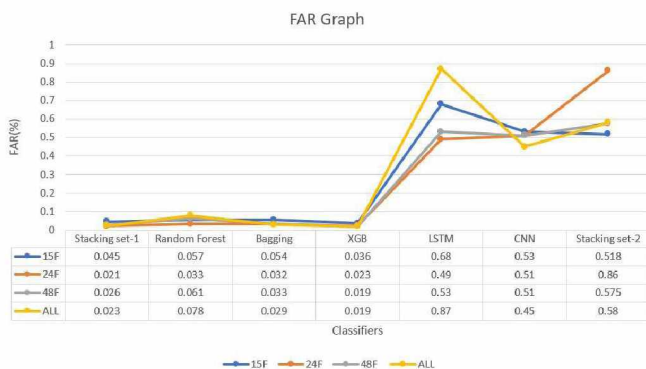


Fig. 4. FAR Graph

### C. Random Forest

With an accuracy of 99.97%, Random Forest's 24-features are performing well. The performance of 48-features (99.94%) and 15-features (99.94%) is similar, as can be seen from the

above Fig 3. Comparatively speaking to other features, the accuracy of 24-features performing well.

The FAR is high(0.078%) for all feature set and low(0.033%) for 24-feature set. Overall the 24-feature set is performing well for Random Forest.

### D. Bagging using DCT

From Fig 3, the comparable results were obtained while bagging 24 (99.946%), 48 (99.967%), and all (99.971%) features sets. If we choose fewer features, we can choose from 24 options. The overall performance of all functionalities is good compared to other features.

The FAR for 24(0.032), 48(0.033) and all(0.029) features are very similar. The 15-feature set is performing poorly. Overall, all features are performing well

### E. XGB

For the XGB 48, all features work flawlessly. You can therefore, choose from any of them. The 48 and all features are significantly different, but the overall accuracy is the same at 99.981%. You can choose 48 attributes if the quantity is more significant to you. The 24(99.977%) and 48(99.981%) features perform rather similarly. If you choose fewer features over others, we must trade off performance; however, this trade-off has little to no effect on performance. You can choose 15 features, and its performance is better than every model that uses 15 features.

The FAR is low(0.019%) the same for both 48 and all feature sets, and high (0.036) for the 15-feature set [10].

## VI. OVERALL PERFORMANCE COMPARISON

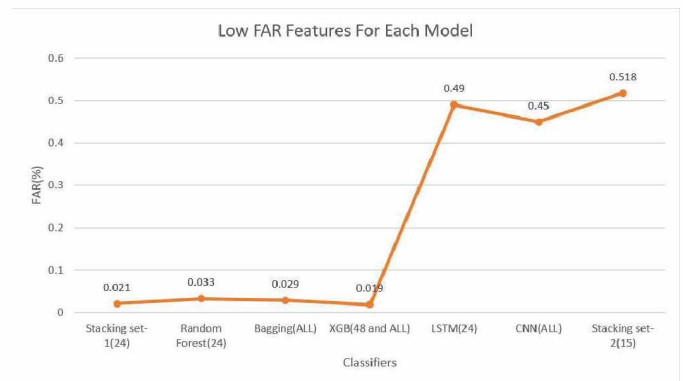


Fig. 5. Low FAR For Each Model

In this section, we present a comparison of the performance of the various models implemented in the previous section. From the result and analysis section, we can draw the conclusion that, if you compare each model with all its feature sets:

- 1) Stacking Set-1 is performing well for 24 features.
- 2) Random Forest is performing well for 24 features.
- 3) Bagging is performing well for ALL features.
- 4) XGB is performing well for 48 and all features.

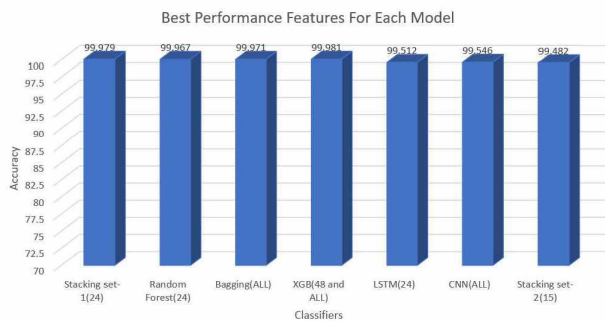


Fig. 6. Best Performance for Each Model

- 5) LSTM is performing well for 24 features.
- 6) CNN is performing well for all features.
- 7) Stacking Set-2 is performing well for 15 features.

If you compare the best-featured model, then the XGB outperforms every other model, as depicted in the figure 5 and figure 6.

## VII. CONCLUSION AND FUTURE WORK

In this study, the results of the experiments that were carried out in order to assess the efficacy of a number of different machine-learning and deep learning classifiers are presented. It was determined to use the CICIDS-2017 dataset for purposes of examination. Only pertinent information will be included to improve the training while decreasing the model's computational complexity. Through the utilization of a suitable feature selection technique, the performance of the model can be improved. To get anything straight, comprehension of the work that is being done requires encouraging findings that have been obtained for use across several different attack classes and have also been arranged in chronological order. One might arrive at the conclusion that a dependable Intrusion Detection System that is able to detect break-ins in real time is also capable of being constructed utilizing a wide number of classifiers, preventing the use of illegal network resources available for access. In addition to this, because attackers typically exploit weaknesses in both types of learning methods, this problem can be solved in the future by employing an approach known as hybridization and optimization that incorporates a number of different types. In addition to this, by taking out the relevant characteristics from the collection and making use of the proper strategies for feature selection and dimension reduction, the effectiveness of the model can be enhanced.

## REFERENCES

- [1] M. A. M. Hasan, M. Nasser, S. Ahmad, and K. I. Molla, "Feature selection for intrusion detection using random forest," *Journal of information security*, vol. 7, no. 3, pp. 129–140, 2016.
- [2] Y. Wang and L. Feng, "An adaptive boosting algorithm based on weighted feature selection and category classification confidence," *Applied Intelligence*, pp. 1–22, 2021.
- [3] A. Mishra, A. M. Cheng, and Y. Zhang, "Intrusion detection using principal component analysis and support vector machines," in *2020 IEEE 16th international conference on control & automation (ICCA)*. IEEE, 2020, pp. 907–912.
- [4] Y. Yin, J. Jang-Jaccard, W. Xu, A. Singh, J. Zhu, F. Sabrina, and J. Kwak, "Igrf-rfe: a hybrid feature selection method for mlp-based network intrusion detection on unsw-nb15 dataset," *Journal of Big Data*, vol. 10, no. 1, pp. 1–26, 2023.
- [5] N. Farnaaz and M. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.
- [6] R. Zhao, Y. Mu, L. Zou, and X. Wen, "A hybrid intrusion detection system based on feature selection and weighted stacking classifier," *IEEE Access*, vol. 10, pp. 71 414–71 426, 2022.
- [7] D. Gaikwad and R. C. Thool, "Intrusion detection system using bagging with partial decision treebase classifier," *Procedia Computer Science*, vol. 49, pp. 92–98, 2015.
- [8] N. Wang, Y. Chen, Y. Hu, W. Lou, and Y. T. Hou, "Feco: Boosting intrusion detection capability in iot networks via contrastive learning," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1409–1418.
- [9] Y. Heryadi and H. L. H. S. Warnars, "Learning temporal representation of transaction amount for fraudulent transaction recognition using cnn, stacked lstm, and cnn-lstm," in *2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*. IEEE, 2017, pp. 84–89.
- [10] Y. Feng, Z. Gao, H. Xiao, X. Yang, and Z. Song, "Predicting the tropical sea surface temperature diurnal cycle amplitude using an improved xgboost algorithm," *Journal of Marine Science and Engineering*, vol. 10, no. 11, p. 1686, 2022.