# Network Threat Detection as a Multi-Class Classifier based on Feature Relevance

Under the guidance of Dr. Pankaj Telang
CSC 522 Automated Learning and Data Analysis

Group 13

Sumeet Bapurao Khillare
skhilla@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Shashank Ajit Walke
swalke@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Raghunandan Ganesh Mante
rmante@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Soham Pravin Gundewar
sgundew@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

## Abstract

This project implements various machine learning techniques classify malicious network behaviors. Using the NSL-KDD dataset during training, we identified important features needed to classify such behaviors. Various different machine learning algorithms were compared, such as KNN, Autoencoders, Decision Trees, Random Forest. The NSL-KDD benchmark dataset and a set of evaluation metrics were used to evaluate the system's performance. Exploratory data analysis was utilized to improve model accuracy by pinpointing and selecting the most influential features. A successful implementation of a novel multi-label classification method, which combines various techniques, was implemented to classify various types of attacks. The results from the experiment highlighted the excellent effectiveness of the Autoencoder-Random Forest blend in precisely identifying and categorizing malicious network activities.

## CCS Concepts

• **Security and privacy → Denial-of-service attacks**; **Intrusion detection systems**.

## Keywords

Machine Learning, EDA, Intrusion Detection systems

## 1 Introduction and Background

Network security has now become one of the most crucial concerns for the modern, connected world of digitized systems. In the ever-changing dynamics and ever-growing landscape of cyber-attacks, it has considerably challenged the security teams to detect and respond against various attacks using manual intervention. The project addresses the increasing need for efficiency and effective accuracy within the network threat detection mechanisms.

Although machine learning holds out promise for identifying cyberattacks, many current models are limited in their ability to detect a wide range of threats, especially when evaluated using real-world datasets such as NSL-KDD. Though popular, NSL-KDD suffers from issues like imbalanced data and redundancy in features that affect the performance of the developed models negatively. For more reliable and practical network security, these issues need to be addressed.

### 1.1 Problem Statement

The major challenge that we will address can be stated as follows, a robust multi-class feature relevance-based classifier for Network threat detection. More specifically :

(1) Improve the classification accuracy of the NSL-KDD dataset, especially for the less frequent attack types
(2) Find and utilize the most relevant features to effectively classify threats.
(3) Overcome the inefficiency of security team's manual threat detection. Improve upon the existing machine learning approaches in network intrusion detection.

### 1.2 Related Work

Previous studies have reviewed various machine learning methods. Methods applied to network intrusion detection are of higher value. Approaches include:

- Random Forests: Unsupervised ensemble learning methodology that builds where several trees are built and then combined to make a better precise and reliable forecasting.
- Decision Trees: Schematic diagrams representing choices hierarchically and Potential consequences, used for classification and regression tasks.
- Support Vector Machines (SVM): A supervised learning model designed to analyze data for classification and regression tasks.

While all these methods have shown promise, they often face challenges in offering high accuracies and reliable operations when tested for benchmark dataset NSL-KDD. Common challenges include:

- Rare types of attacks are hard to classify.

- Overfitting to majority classes
- Failure to handle the high-dimensional feature effectively in domain of network traffic data

Our approach builds on this foundation by focusing on characteristics to improve the classification accuracy for the various attack types. Aiming to handle certain limitations of the previous approaches, we try to develop an optimized solution to detect various threats in networks.

## 2 Method

### 2.1 Novel Aspects

Our novel approach combines an autoencoder for feature extraction with a random forest for classification, hence taking advantage of both unsupervised and supervised learning techniques. The unsupervised learning of an autoencoder enhances the supervised learning of a random forest to improve overall accuracy and classify underrepresented classes better. This hybrid model has been applied to the NSL-KDD dataset for intrusion detection by considering complex network security data and unequal distribution of classes.

### 2.2 Why this choice?

While autoencoders are great for learning complex patterns and reducing input data in dimensionality, random forests represent a high-performance algorithm for classification. We combined both, where the autoencoder brings out filtered features that then improve the classification accuracy of a random forest immensely. In our case, the autoencoder managed to reduce the feature space from 124-when the categorical variables were one-hot encoded-to just 32, greatly simplifying the task for the random forest classifier. This reduction in dimensions enhances not only the efficiency but potentially the generalizing capability of the model as well, narrowing it down to the most salient features extracted by the autoencoder.

### 2.3 Approach

(1) Data Preparation: Cleaning of the data, normalize numeric features, oneHotEncoding applied to categorical variables.
(2) AutoEncoder Training: The AutoEncoder model is trained using preprocessed data.
(3) Dimensionality Reduction: The training data was processed for dimensional reduction by AutoEncoder.
(4) Random Forest Training: The algorithm trains a random forest model with reduced features generated by AutoEncoder processing.
(5) Test Data Preparation: The test data preparation was done by using the same AutoEncoder model, to reduce dimentinos of test data.
(6) Final Classification: This preprocessed dataset was used to perform the analysis using our Random Forest Model.
(7) Performance Evaluation: Finally, we evaluated the classification reports and confusion matrix of the models to see how well the model performed against the test dataset.

## 3 Plan and Experiment

We performed systematic and orderly experiments on different machine learning models. The dataset NSL-KDD is used to apply preprocessing steps such as feature encoding, normalization, and categorizing attacks in classes. This processed data will be further passed to various machine learning models. This also involves hyperparameter tuning, suitable performance metrics selection, and comparisons of the method to facilitate thorough assessment and categorization. The following sections describe the dataset, our hypothesis, and the experimental framework in greater depth.

### 3.1 Dataset Description

For this project, we are using the NSL-KDD dataset, a modified version of the original KDD Cup 1999 dataset. It is widely known in network security research, and addresses several limitations of its predecessor, such as removing redundant records and gives a more balanced distribution of attack types. This ensures more meaningful evaluations and mitigates biases that previously skewed results.

The dataset consists of 125,973 training records and 22,544 test records, each representing a network connection with 41 features. These features are grouped into four categories:

- Basic features: Characteristics like connection duration, Protocol type and service.
- Content features: Metrics such as the number of failed login attempts and file creation events
- Time-based traffic features: Indicators like the number of connections to the same host within a specific timeframe
- Host-based traffic features: Metrics, including the percentage of connections with SYN errors

By leveraging the NSL-KDD dataset, our multi-class classifier for network threat detection is built on a robust and widely accepted benchmark, ensuring that our findings can be meaningfully compared to prior work in the field.

### 3.2 Hypothesis

(1) Feature Relevance Impact: Using feature relevance will increase the accuracy in the classification of framework-especially for the less frequent attack categories such as Privilege and Access.
(2) Model Comparison: Advanced hybrid approaches such as Autoencoders combined with Random Forest will outperform standard models like Decision Trees or Random Forest alone and KNN in terms of detection accuracy and false preferential rates.

These hypotheses are expected to prove whether an emphasis on feature selection and hybrid modeling enables addressing the existing limitation within Network Intrusion Detection.

### 3.3 Exploratory Data Analysis and Preprocessing

- **Initial Analysis:**Initially, the thorough analysis of the NSL-KDD dataset with 41 columns was held. Attributes:
  - Continuous independent attributes: duration, src_bytes, dst_bytes

  - Major categorical attributes: protocol_type, service, flag
  - Dependent attribute: label (indicating normal or attack network entry)
- **Classification Categorization:**
  - Binary classification: Normal and Abnormal
  - Multi-class classification: Normal, DoS (Denial of Service), Probe, Privilege, and Access (based on Cybersecurity domain knowledge)
- **Data Visualization:** Box-plots were produced for each feature to visualize data distribution.
- **Normalization:** Min-max normalization was done on the numeric attributes with skewed distribution.
- **Feature Selection:** The correlation of numeric features to dependent attributes was assessed and nine highly correlated features were selected.
- **Categorical Feature Transformation:** Used OneHotEncoder to transform the categorical features.
- **Final Dataset:** A dataset of 93 independent columns and 1 dependent column was then formed at the end of the preprocessing.

## 3.4 Model Training and Experimentation

*3.4.1* ***Decision Tree.*** Decision trees are supervised learning algorithms, which run some classification and regression tasks within them. They are in a structure composed of nodes and branches, which may start with a root node and visit leaf nodes. We chose decision trees to analyze because:

(1) These are comparatively simple to understand and implement
(2) These can handle numeric as well as categorical data
(3) Well suited to diverse data types in the NSL-KDD dataset

In order to optimize the decision tree model, the following hyperparameters were tuned:

(1) max_depth: Controls the maximum depth of the tree.
(2) max_features: Determines the number of features to consider when looking for the best split.
(3) min_samples_split: Minimum number of samples to split an internal node.
(4) min_samples_leaf: Minimum number of samples required to be at a leaf node.
(5) criterion: Function to measure the quality of a split (gini or entropy for classification).

We tested various combinations of these hyperparameters using 5-fold cross-validation. We found best results with 75% accuracy for these parameters: 'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2

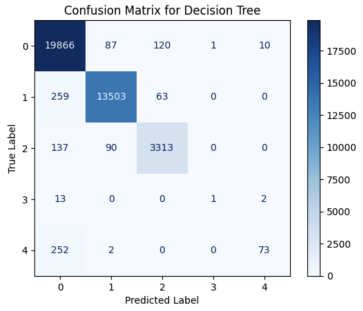


**Figure 1: Decision Tree Confusion Matrix**

*3.4.2* ***RandomForest.*** RandomForest is a method of prediction using multiple decision trees that are trained on subsets of dataset and features and then aggregating their results for classification. Here's how it works: Constructing the Forest When we train a RandomForest, we generate thousands of decision trees. Each tree is slightly different because: It has only seen part of our training data It only takes into account some of the features we've got This diversity makes our forest resilient and adaptive. Decision Making When attempting to classify something, whether network traffic is normal or an attack, for example: We share our findings with all the trees in the forest. Each tree makes its own decision We enumerate all the decisions made. The most popular choice wins We ran experiments with multiple numbers of estimators in our Random Forest model to find the best setting. Our results showed that tuning hyper-parameter n_estimators to 100 gave us the best performance, achieving an accuracy of approximately 75% on our dataset.

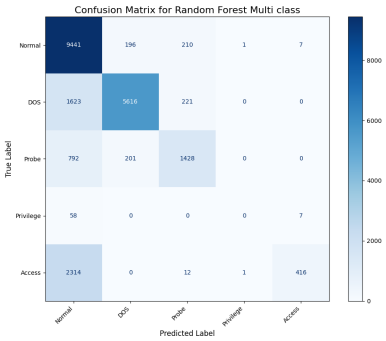


**Figure 2: RandomForest Confusion Matrix**

*3.4.3* ***K-Nearest Neighbors.*** This section outlines the implementation of KNN for network intrusion detection using multiclass classification using the NSL-KDD dataset. KNN's simple and effective algorithm adapts well to unknown attack patterns without requiring prior data distribution knowledge thus making it suitable for network intrusion detection. And the model varies its performance based on tuning parameters like the number of nearest neighbor's 'k', min-squared error, and training dataset.

The KNN classifier operates by identifying the k nearest neighbors of a data point and assigning the majority class among these

neighbors. The model's performance can be evaluated by plotting the error rate over values of 'k' by allowing us to get the full picture of its accuracy. Thus, selecting k is very important in KNN classification. This was achieved by iterating k values from 3 to 10 in our model and calculating the error rate against the 'k' value. This showed us that at k = 8, we get the highest accuracy and lowest error rate. The accuracy achieved as a result of this was 75.02% which is a bit better compared to previous results from decision tree and nearly similar to results of random forest.

(1) Input Layer: dataset that contains multiple features formatted, normalized, and pre-processed.
(2) KNN Classifier:
    - The KNN algorithm uses data points that are nearest to the data point to be classified and assigns the label of the majority of points from 'k' neighbors.
    - The number of neighbors (k) is a hyperparameter which influences the classification results. A range of k values is iterated in the range and for each value of k error rate is calculated which is also important factor in determining the model efficiency. And we get lowest error rate at k = 8, thus highest accuracy at value of k.

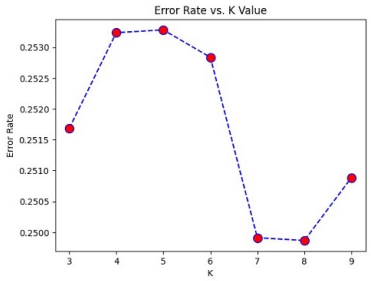


**Figure 3: Error rate vs K value**

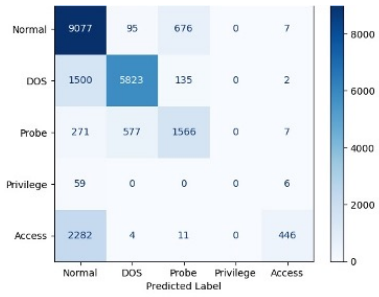


**Figure 4: KNN Confusion Matrix**

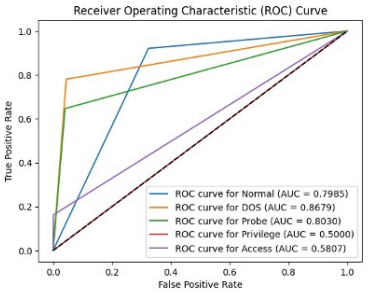


**Figure 5: KNN ROC curve**

*3.4.4* ***AutoEncoder + Random Forest.*** This section describes the development of novelty in threat detection using the NSL-KDD dataset by proposing the autoencoder along with random forests. In developing the architecture of the proposed hybrid model, it has been motivated by the fact that machine learning algorithms are unable to achieve a performance better than 75% in a multi-class classification problem in spite of features being pre-processed and reduced in dimension through correlation analysis. Our system is thus based on two key modules: feature extraction by an autoencoder that reduces their number once extracted and threat detection by a random-forest type classifier. The architecture of the autoencoder is as follows:

Encoder:

- Input layer (input_dim)
- Dense layer of 128 neurons with ReLU activation
- Dense layer of 96 neurons with ReLU activation
- Dense layer of encoding_dim neurons with ReLU activation

Decoder:

- Input layer of encoding_dim
- Dense layer of 96 neurons with ReLU activation
- Dense layer of 128 neurons with ReLU activation
- Output layer: input_dim neurons, linear activation

This is an example of a symmetric structure since it contains three layers in both the encoder and the decoder. These dimensions decrease gradually from an input size of 128 down to 96, ending in a dimensionality of encoding from which the decoder should be able to reconstruct the original input. The ReLU was applied in this work to add nonlinearity on the hidden layers, while the output layer of the decoder uses the linear activation function to allow unbounded reconstructions.

The autoencoder will be trained using the Adam optimizer and Mean Squared Error. To avoid overfitting, the following methods ensure the best training performance:

(1) Early stopping on the validation loss with a patience of 10 epochs.
(2) Learning Rate Tuning: This step will tune the Learning Rate for an optimal trade-off between convergence speed and stability.
(3) Hyperparameter Tuning: It was done pretty naively by trying different combinations of epoch vs batch size to see the performance.

(4) Trying Different Types of Activation Functions: Several types of activation functions were tried (Sigmoid, ReLU, Tanh, SoftMax) ; among them, ReLU gave the best performance on our dataset.

Hence, The features used as input for the random forest classifier are the autoencoder representations of the original data. As part of our attempt to ensure the random forest algorithm was best set to obtain optimal performance, we did a tuning of the random forest hyperparameters for the best result in this multi-class classification task using Grid Search Cross-Validation.

The proposed approach, therefore, includes an autoencoder to perform the extraction and a random forest classifier that effectively maps the complex relations within the NSL-KDD dataset. In other words, the autoencoder's capability to generate a compact representation from unrefined input provided an effective method of noise reduction in inputs and effective feature extraction; a random forest classifier will carry out predictions based on refined features.

This hybrid model has a multi-class classification among the four classes of threats, with an accuracy of about 78.14%. Quite impressive against previous approaches, this presents the effectiveness of deep learning feature extraction and traditional machine learning classification in network security applications.

Accuracy: 0.7814

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.68 | 0.97 | 0.80 | 9855 |
| 1 | 0.96 | 0.79 | 0.87 | 7460 |
| 2 | 0.89 | 0.81 | 0.85 | 2421 |
| 3 | 0.80 | 0.06 | 0.11 | 65 |
| 4 | 0.97 | 0.06 | 0.11 | 2743 |
| accuracy | | | 0.78 | 22544 |
| macro avg | 0.86 | 0.54 | 0.55 | 22544 |
| weighted avg | 0.83 | 0.78 | 0.74 | 22544 |

**Figure 6: Classification Report - Autoencoder + Random Forest**

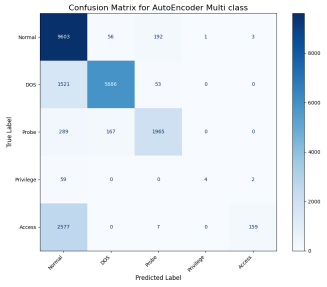


**Figure 7: Confusion Matrix - Autoencoder + Random Forest**

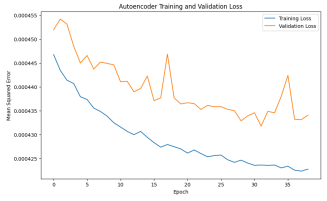


**Figure 8: Train, Val loss over epoch**

## 3.5 Experimental Design

In this section, we will give details on our experiments:

*3.5.1 Preprocessing.*

(1) Categorical Feature Encoding: We converted categorical features like protocol_type and service into numerical representations using one-hot encoding or label encoding.
(2) Normalization: Min-Max Scaler was used for normalization of numeric data.
(3) Train-Test Splits: We used the provided training and test sets in the NSL-KDD dataset without further partitioning to maintain consistency with prior research.

*3.5.2 Baseline Models.* Train and evaluate standard models like Decision Tree, Random Forest, and KNN to establish benchmark performance.

*3.5.3 Hybrid Model Implementation.* After implementing Autoencoders + Random Forest we evaluated the impact of dimensionality reduction and feature learning on classification performance.

*3.5.4 Evaluation Metrics.* We computed classification reports and confusion matrix for performance evaluation of different models.

*3.5.5 Experiment Execution.* We conducted experiments in multiple iterations, averaging results to account for random variations. Record results for different attack classes to evaluate the models' effective- ness in handling imbalanced class distributions.

## 4 Conclusion

The investigation into intrusion detection utilizing the NSL-KDD dataset has produced encouraging outcomes, especially through the innovative method that integrates autoencoders with random forests. This combined model surpassed conventional machine learning methods, illustrating the capability of unsupervised feature learning within the realm of network security applications. The combination of the autoencoder and random forest model attained an accuracy rate of 78.14% for multi-class classification, exceeding the efficacy of alternative models that were evaluated. This enhancement implies that unsupervised learning methodologies are capable of effectively deriving significant features from intricate network data, thereby improving the overall detection proficiency.

However, this wide margin in training accuracy, which stands at 99%, compared to that of the test accuracy, standing at 78.14%, raises red flags concerning overfitting issues. The bias could be because of some inherent biases in the NSL-KDD dataset, especially within the disproportionate representation of a few attack classes, including privilege escalation and unauthorized access. Though our current approach looks very promising, there is still room for improvement.

By comparison, the confusion matrix of the models yields the following observations:

(1) It shows that the combination of the AutoEncoder and RandomForest did a better job compared to others in modeling the classes of Normal, DoS, and Probe attacks.
(2) Hybrid methodology certainly performed poorly in trying to identify attacks of this class, but can be enhanced by including balanced data relevant to access attacks.

(3) Individual models like RandomForest, KNN, and Decision Tree resulted in poor performance in classifying certain types of attacks, whereas the proposed ensemble method with the inclusion of AutoEncoder with RandomForest classified different types of attack categories far more efficiently.

Future studies shall focus on:

(1) Smoothening discrepancies in data, even by using sophisticated sampling methods for the creation of synthetic data.
(2) Investigating advanced methodologies for feature extraction to identify nuanced patterns within marginalized categories of attacks.Assessing alternative deep learning frameworks that might be more effective in managing imbalanced network security datasets.
(3) Formulating strategies aimed at enhancing model generalization and minimizing the disparity between training and testing outcomes.

In summary, although our autoencoder combined with a random forest model illustrates the possibilities of integrating unsupervised and supervised learning approaches for intrusion detection, it simultaneously underscores the persistent challenges present in this domain. Ongoing investigation into dataset refinement, feature engineering, and sophisticated machine learning methodologies is essential for the creation of more resilient and precise intrusion detection systems.

## References

[1] Nabila Farnaaz and M.A. Jabbar. 2016. Random Forest Modeling for Network Intrusion Detection System. *Procedia Computer Science* 89 (2016), 213–217. https://doi.org/10.1016/j.procs.2016.06.047

[2] Bhupendra Ingre, Anamika Yadav, and Atul Soni. 2017. Decision Tree Based Intrusion Detection System for NSL-KDD Dataset. https://doi.org/10.1007/978-3-319-63645-0_23

[3] M. Komorowski, D.C. Marshall, J.D. Salciccioli, and Y. Crutain. 2016. Exploratory Data Analysis. In *Secondary Analysis of Electronic Health Records*. Springer, Cham. https://doi.org/10.1007/978-3-319-43742-2_15

[4] L. Pan, T. Zheng, and R. Fu. 2022. Research on Intrusion Detection Model Based on PCA-SVM. In *2022 IEEE 4th International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*. 1270–1275. https://doi.org/10.1109/ICCASIT55263.2022.9986847

| Meeting | Team Members Attended | Discussions | Deliverable for phase 1 |
|---|---|---|---|
| Meeting 1: 20 Oct 2024 Before Phase 1 Week 1 (ending 27 Oct) | Raghunandan Mante, Shashank Walke, Sumeet Khillare, Soham Gundewar | Idea discussion, dataset selection and what preprocessing to be done as per model to be evaluated. Novel approach finalization | The group will aim to finalize the idea for the project, select a dataset and produce a cleaned dataset along with a comprehensive EDA report. |
| Meeting 2: Before Phase 2 Week 2 (ending 3 Nov) | Soham Gundewar, Raghunandan Mante, Shashank Walke, Sumeet Khillare | Discussions involved creating collaboration environment, and documenting progress. Relevant feature identification using EDA and model selection discussion for classification, starting novel approach implementation. | This meeting we had discussions around various models to be implement for comparison around novel approaches and identifying relevant features for training models. |
| Meeting 3: Before Phase 3 Week 3 (ending 10 Nov) | Shashank Walke, Raghunandan Mante, Sumeet Khillare, Soham Gundewar | Discussions on collab environment and distribution of model training where each one selects different model for implement allowing parallel working and higher efficiency | As result we had finalized various models that we planned to implement and used google drive and google Colab to share preprocessed data and code. |
| Meeting 4 Phase 4 Week 4 (ending 17 Nov) | Sumeet Khillare, Soham Gundewar, Raghunandan Mante, Shashank Walke | Discussions around the reports of each model and their respective results based on criteria selected for comparison. | As finalized we had models of types KNN, Random forest, Decision tree and autoencoder+random forest, their respective results and metrics were used to compare and conclusions were made |
| Meeting 5 Week 5 (ending 24 Nov) | Sumeet Khillare, Raghunandan Mante, Shashank Walke, Soham Gundewar | All the reports and results were concatenated together to form and complete report | Final report was completed and submitted by the deadline |

**Table 1: Work Distribution**