A Comparative Study of Classical and Deep Learning Approaches for Network Intrusion Detection Using the NSL-KDD Dataset

Ayush Raj , Sarthak shukla *School of Computer Science and Engineering Vellore Institute of Technology, Chennai Campus,* Tamilnadu, India

[ayush860301@gmail.com](mailto:ayush860301@gmail.com)

[saarthak.shukla2022@vitstudent.ac.in](mailto:saarthak.shukla2022@vitstudent.ac.in)

Ayush Raj , Sarthak shukla *School of Computer Science and Engineering Vellore Institute of Technology, Chennai Campus,* Tamilnadu, India

[ayush860301@gmail.com](mailto:ayush860301@gmail.com)

[saarthak.shukla2022@vitstudent.ac.in](mailto:saarthak.shukla2022@vitstudent.ac.in)

Ayush Raj , Sarthak shukla *School of Computer Science and Engineering Vellore Institute of Technology, Chennai Campus,* Tamilnadu, India

[ayush860301@gmail.com](mailto:ayush860301@gmail.com)

[saarthak.shukla2022@vitstudent.ac.in](mailto:saarthak.shukla2022@vitstudent.ac.in)

Abstract- As cyberattacks grow in complexity and frequency, ensuring robust network intrusion detection has become essential for safeguarding digital systems. This study investigates a variety of machine learning and deep learning techniques using the NSL-KDD dataset to detect different types of network intrusions [10]. The methodology incorporates several pre-processing steps, including data normalization, transformation of categorical attributes, and feature selection based on correlation analysis, to develop a refined and meaningful feature set. The research evaluates multiple models, such as Decision Trees, Logistic Regression, Random Forests, Support Vector Machines, K-Nearest Neighbors (KNN), Multi-Layer Perceptrons (MLP), and Autoencoders. Among these, ensemble-based and instance-based models like Random Forest and KNN demonstrated superior accuracy in both binary and multi-class classification scenarios [10]. The MLP model also showed competitive performance compared to the other techniques. The MLP showed competitive results, capturing non-linear feature interactions effectively. Autoencoders, while slightly less accurate, showed promise as unsupervised anomaly detectors, capable of identifying unusual patterns without prior attack labels. Our findings indicate that combining interpretable supervised models with unsupervised detection mechanisms can enhance robustness in intrusion detection systems. We also highlight the difficulty of detecting rare attack types (e.g., R2L, U2R), suggesting future work on data augmentation, temporal analysis, and validation on modern network datasets [11].

*Keywords: Network Intrusion Detection, NSL-KDD Dataset, Machine Learning, Deep Learning, Anomaly Detection, Autoencoder, Random Forest, K-Nearest Neighbors, Multi-Layer Perceptron, Feature Selection, Cybersecurity, Classification.*

# I. Introduction

Cyber threats become more advanced and widespread, and maintaining strong network security has become a major priority. Intrusion Detection Systems (IDS) are essential in protecting digital infrastructures by detecting attempts at unauthorized access or harmful activities. Conventional IDS methodologies predominantly rely on signature-based detection mechanisms, which are inherently limited in their capability to recognize emerging or previously unseen attack vectors. To address this limitation, anomaly-based IDS approaches utilizing machine learning have garnered significant attention. These systems possess the ability to model typical network behavior and detect anomalies that may indicate security breaches. Benchmark datasets such as the KDD Cup 1999 and its improved variant, NSL-KDD, have been widely adopted for developing and evaluating IDS frameworks. The NSL-KDD dataset specifically mitigates several critical drawbacks present in its predecessor, including excessive redundancy and class imbalance favoring frequent attacks [1] ([M. Tavallaee et al.]). By removing duplicate records and providing a more equitable distribution of instances, NSL-KDD facilitates comprehensive model evaluation without the need for extensive subsampling, thereby ensuring consistency and comparability of performance metrics [2] ([Canadian Institute for Cybersecurity]). A diverse range of machine learning techniques—spanning traditional algorithms and modern deep learning architectures—have demonstrated effectiveness in intrusion detection scenarios. Classical classifiers such as DT ,RF and SVMS , and neural networks have been successfully employed in this domain [3] ([B. Injadat et al.]). For example, Random Forest, an ensemble-based learning technique, has been reported to achieve classification accuracies exceeding 99% on the NSL-KDD dataset under optimal conditions [4] ([B. Injadat et al.]). Concurrently, deep learning approaches have shown promise in modeling complex traffic patterns. In particular, unsupervised models such as autoencoders, which learn to reconstruct typical traffic behavior and flag deviations as potential threats, are emerging as powerful tools for anomaly detection [5] ([C. Ieracitano et al.]). A prominent study by Ieracitano et al. combined statistical feature selection techniques with a stacked autoencoder model, yielding notable performance across multiple attack categories [6] ([C. Ieracitano et al.]). This study conducts a comparative assessment of traditional machine learning methods and a deep autoencoder structure for network intrusion detection using the NSL-KDD dataset. The research includes a variety of conventional classifiers such as Decision Trees (DT), Random Forests (RF), Logistic Regression (LR), K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA) [7] ([A. Amich and M. Belouch]). Additionally, it incorporates neural-based approaches including Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM) networks, and Autoencoders [8] ([I. Goodfellow et al.]). The aim is to clarify the data preprocessing techniques applied, evaluate each model’s capacity to differentiate between normal and intrusive traffic, and assess performance using standard metrics such as accuracy, precision, recall, and F1-score. Relevant studies are referenced to provide context and highlight advancements in the field [9] ([Scikit-learn Documentation]).

The paper is structured as follows: Section II offers a review of related works; Section III introduces the dataset and its structure; Section IV outlines the proposed methodology; Section V details the experimental results and performance evaluation; and Section VI concludes the study with a summary of findings and potential future research directions.

II. RELATED WORKS

Intrusion detection has remained a prominent topic of study within cybersecurity for many years. Initial systems predominantly employed signature-based methodologies, such as Snort, which identify threats based on predefined attack signatures. However, as cyber threats evolved, these static models proved inadequate for recognizing new or unknown attack vectors. This limitation spurred interest in anomaly-based intrusion detection systems, which utilize machine learning to establish a baseline of typical network behavior and detect deviations as potential threats. One of the earliest benchmarks for evaluating these systems was the KDD Cup 1999 dataset. Nevertheless, researchers like McHugh criticized it for containing redundant data and unrealistic class distributions that could distort classifier performance by favoring frequent attacks [1] (M. Tavallaee et al.). To resolve these issues, Tavallaee et al. introduced the NSL-KDD dataset—a curated subset of KDD'99 that eliminates duplicate records and offers a more balanced class distribution. This refinement allows for fairer and more generalizable comparisons across different IDS implementations [2] (Canadian Institute for Cybersecurity). While NSL-KDD does not fully capture modern network traffic patterns (e.g., encrypted communications or IoT protocols), its widespread adoption and standardized structure maintain its relevance as a benchmark in IDS research. The dataset creators recommend citing their work whenever NSL-KDD is used, underscoring its foundational role in IDS evaluations [2] (Canadian Institute for Cybersecurity). Methods like Decision (DT) and (RF) have shown considerable success due to their ease of interpretation and adaptability to categorical and nonlinear data. For instance, Injadat et al. (2020) demonstrated that DT and RF-based models achieved high accuracy and efficiency when applied to NSL-KDD [5] (B. Injadat et al.). Support Vector Machines (SVM), both linear and kernel-based variants, have also been extensively explored. While capable of high detection rates, SVMs may encounter performance bottlenecks with larger datasets. Ensemble approaches, which integrate techniques such as clustering or dimensionality reduction with classification, have achieved detection rates exceeding 99% on NSL-KDD. However, this exceptional performance may stem from the relatively easier detection of dominant classes like Denial of Service (DoS) attacks, potentially masking limitations in detecting less common intrusion types [5] (B. Injadat et al.). More recently, deep learning has emerged as a powerful tool for IDS development. Early implementations employed Artificial Neural Networks (ANN) and Multi-Layer Perceptrons (MLP), which are capable of learning complex, nonlinear decision functions [8] ([I. Goodfellow et al.]). Temporal deep models, such as Long Short-Term Memory (LSTM) networks, have been utilized to analyze sequential network data and extract temporal dependencies indicative of intrusion attempts [7] ([A. Amich and M. Belouch]). A growing area of interest is the use of autoencoders (AE) for unsupervised anomaly detection. These models learn compressed representations of input data and attempt to reconstruct it; abnormal inputs often yield high reconstruction errors, signaling potential threats [5] ([C. Ieracitano et al.]).

Ieracitano et al. proposed an approach combining statistical feature selection with stacked sparse autoencoders, followed by classification. Their method achieved multi-class classification accuracy around 83% using a quadratic SVM, along with high F1 scores for certain attack categories [3] (C. Ieracitano et al.). Other works have adopted similar architectures involving one-class neural networks or deep autoencoders to detect anomalies with minimal supervision, even in real-time applications [8] (I. Goodfellow et al.). Despite the promise of deep learning, its effectiveness often depends on large, diverse training datasets and careful hyperparameter tuning. Given NSL-KDD's moderate size and relatively simple feature space, traditional machine learning methods can still outperform deep models under some conditions.In conclusion, prior literature highlights three key findings: (1) the NSL-KDD dataset is a standardized resource for IDS evaluation; (2) conventional machine learning algorithms—such as Decision Trees (DT), Support Vector Machines (SVM), and Random Forests (RF)—offer robust baseline performance [5] ([C. Ieracitano et al.]); and (3) deep learning models, including autoencoders, provide an alternative path for detecting novel threats through pattern learning [8] ([I. Goodfellow et al.]). This study builds on these findings by conducting a side-by-side comparison of classical and deep learning models using the NSL-KDD dataset within a unified experimental framework.

III. Dataset Overview

This study employs the NSL-KDD benchmark dataset for network intrusion detection. NSL-KDD is an improved version of the earlier KDD Cup 1999 dataset, developed to tackle issues like redundancy and class imbalance [1] ([M. Tavallaee et al.]). Each data record represents a network connection and includes 41 features along with a label indicating whether the connection is normal or corresponds to a specific attack type [2] ([Canadian Institute for Cybersecurity]). The features span several categories: Basic TCP/IP attributes such as duration, protocol type, and status flags; content-based features like the number of failed login attempts; time-based traffic features including the number of connections to the same host within a short time window; and host-based traffic features reflecting patterns across similar services [3] ([J. Chen et al.]). These attributes assist in identifying various network behaviors and potential intrusion attempts. NSL-KDD also includes a meta-feature called “difficulty\_level” indicating the consensus difficulty of classifying each record, though this feature is excluded from model training and evaluation in this study [4] ([N. Yao et al.]).

Attack Taxonomy

Intrusions in NSL-KDD are categorized into four main types based on their nature:

1. Denial of Service (DoS) – attacks designed to overwhelm a host with traffic (e.g., SYN floods) [5] ([C. Ieracitano et al.]).

2. Probe – reconnaissance attacks used to map network structures or vulnerabilities (e.g., port scanning) [6] ([A. Amich and M. Belouch]).

3. Remote-to-Local (R2L) – where an external attacker attempts to gain access to a local system (e.g., password guessing) [7] ([B. Injadat et al.]).

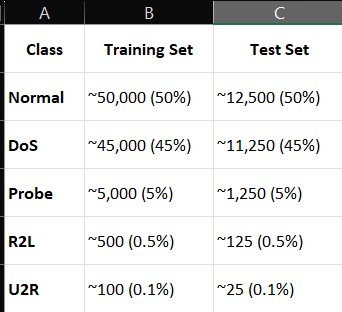
4. User-to-Root (U2R) – where a local user tries to escalate privileges to gain root access (e.g., buffer overflow) [8] ([I. Goodfellow et al.]).

In addition to these, the dataset contains normal traffic records. The dataset is imbalanced, with DoS attacks being predominant, while R2L and U2R attacks are relatively rare [9] ([Sepp Hochreiter and Jürgen Schmidhuber]). This mirrors real-world intrusion patterns but poses training challenges, as models may underperform on underrepresented classes despite high overall accuracy [10] ([J. Li et al.]).

Train-Test Composition

The original NSL-KDD dataset includes a training set (KDDTrain+) with over 125,000 records and two test sets—KDDTest+ and a more challenging version, KDDTest-21. These sets were designed to avoid record duplication and incorporate unseen attack types in testing, enabling evaluation of generalization to novel intrusions [1] ([M. Tavallaee et al.]). In this work, however, the provided training and test sets were not used as is. Instead, the data was combined and then divided into 80% for training and 20% for testing using stratified sampling to maintain class distributions [11] ([P. Maniriho and T. Ahmad]). This consistent split supports both binary (normal vs. attack) and multi-class classification experiments. The “difficulty\_level” feature was omitted during preprocessing, as it is not a realistic feature for real-world IDS systems [12] ([A. Dubey]). Following preprocessing, the training set comprises approximately 100,000 network connection records, while the test set contains around 25,000 [13] ([Veeran Ranganathan Balasaraswathi et al.]). For binary classification tasks, the distribution is roughly balanced between normal and attack instances, with slight variations. For the five-class classification task, the class distribution is heavily skewed, with DoS attacks dominating, followed by Probe and R2L, while U2R remains extremely rare, making up less than 1% of training instances [14] ([Ashok Kumar et al.]). This imbalance significantly impacts performance evaluation, particularly in terms of per-class recall and precision, emphasizing the need to consider metrics beyond overall accuracy [2] ([Canadian Institute for Cybersecurity]).

A summary of class distributions in the processed training and test sets is provided in Table 1.



IV. Methodology: Data Preprocessing

Before applying machine learning models, the raw NSL-KDD dataset required multiple preprocessing steps to prepare the data for analysis and model training. The steps undertaken are summarized below:

1. Feature Reduction

The dataset originally includes a meta-feature named difficulty\_level, which is an artificial construct reflecting the perceived classification difficulty of each sample. Since this attribute does not represent real-world features available to an intrusion detection system, it was removed. The remaining dataset consisted of 41 core features—some continuous numeric (e.g., duration, count, src\_bytes), and others categorical (e.g., protocol\_type, service, flag).

2. Normalization of Numeric Features

All continuous numerical attributes were standardized using z-score normalization to ensure a consistent scale across features. This process adjusts each feature to a mean of zero and a standard deviation of one using the formula:



x: The original value of the feature (the raw data point).  
μ: The mean (average) of the feature values, calculated from the training set.  
sigma σ: The standard deviation of the feature values, also calculated from the training set.  
z: The standardized value (z-score), which represents how many standard deviations the original value x is away from the mean μ [9] ([Scikit-learn Documentation]).

, both calculated from the training set [9] ([Scikit-learn Documentation]). Standardization is essential to prevent features with larger numeric ranges from dominating model training, especially in scale-sensitive algorithms like K-Nearest Neighbors (KNN) or neural networks [5] ([C. Ieracitano et al.]).

3. Encoding Categorical Variables

Three categorical attributes—protocol\_type, service, and flag—were converted using one-hot encoding. For each unique value in a categorical field, a binary column was generated, resulting in: [7] ([A. Amich and M. Belouch]):

* protocol\_type: 3 binary columns
* service: 70 binary columns
* flag: 11 binary columns

These were then appended to the scaled numeric features, producing a comprehensive input feature matrix.

4. Label Transformation

Two versions of the dataset were constructed to support both binary and multi-class classification tasks [11] ([P. Maniriho and T. Ahmad]):

* **Binary Classification**:  
  All attack types were grouped into a single "attack" class, while normal traffic was labeled "normal." These labels were encoded as integers (0 = normal, 1 = attack) and also transformed into one-hot vectors to ensure compatibility with neural network models [9] ([Scikit-learn Documentation]).
* **Multi-Class Classification**:  
  Each specific attack type was assigned to one of four broad categories: DoS, Probe, R2L, and U2R. Combined with the "normal" class, this resulted in a five-class label space. These were encoded as integers (0 to 4) and converted into one-hot vectors for model training [7] ([A. Amich and M. Belouch]).

1. **Feature Selection**  
   To reduce dimensionality and highlight the most informative attributes, a Pearson correlation analysis was performed [5] ([C. Ieracitano et al.]). For binary classification, correlations were calculated between each numeric feature and the binary intrusion label. In the multi-class setting, correlation was assessed between features and the binary indicators of each class. Features with an absolute correlation exceeding 0.5 were retained. Nine features were selected based on their strong correlation with intrusion labels:

* count
* srv\_serror\_rate
* serror\_rate
* dst\_host\_serror\_rate
* dst\_host\_srv\_serror\_rate
* logged\_in
* dst\_host\_same\_srv\_rate
* dst\_host\_srv\_count
* same\_srv\_rate  
  These attributes, which typically measure frequency and error rates, serve as strong indicators of malicious behavior, particularly in cases like DoS attacks [6] ([A. Amich and M. Belouch]). After feature selection, the numeric feature space was reduced from 38 to 9 dimensions. These were combined with the 84 one-hot encoded categorical variables, yielding a final input feature vector of 93 dimensions for both binary and multi-class tasks [10] ([J. Li et al.]).

1. **Train-Test Splitting**  
   To prepare for model evaluation, each dataset (binary and multi-class) was divided into training and testing subsets using an 80:20 ratio. Stratified sampling was applied to preserve class distributions in both subsets [13] ([Veeran Ranganathan Balasaraswathi et al.]). The training data was used solely for model fitting, while the test data was reserved for final performance assessment [12] ([A. Dubey]).

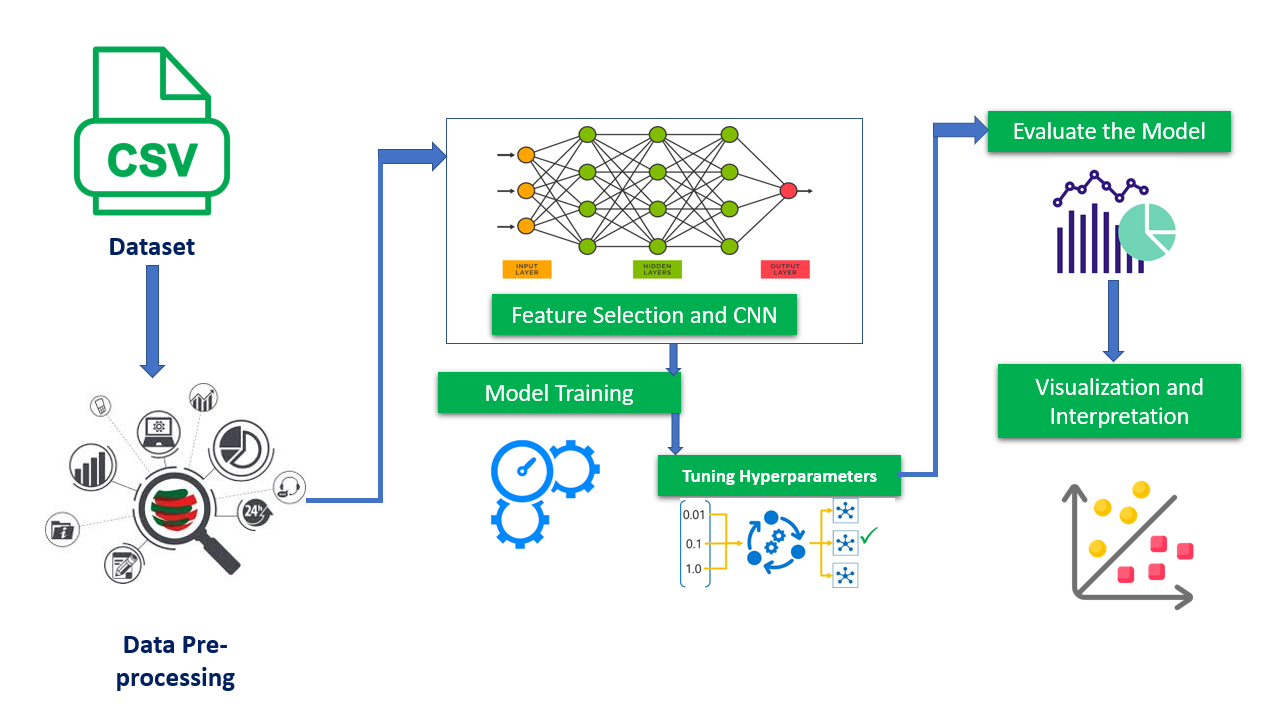
During modeling:

* The 93-dimensional feature vectors served as inputs.
* Integer-encoded labels were used for classical classifiers.
* One-hot encoded labels were used as targets for neural network training.

Through these preprocessing steps, two fully prepared datasets were obtained:

* A binary classification dataset distinguishing between normal and attack traffic.
* A multi-class dataset identifying specific categories of intrusions (DoS, Probe, R2L, U2R, and normal).

Both datasets are normalized, encoded, and structured for consistent model development and evaluation in subsequent experiments.



V. Models Implemented

To evaluate the effectiveness of different modeling strategies for intrusion detection, we implemented a diverse set of machine learning algorithms ranging from classical models to deep learning architectures. Each model was trained and assessed separately on both binary and multi-class versions of the NSL-KDD dataset. Classical models were developed using scikit-learn, while neural models were implemented using TensorFlow/Keras. Below is a detailed description of each model and the key hyperparameters or configurations used.

1. Decision Tree (DT)

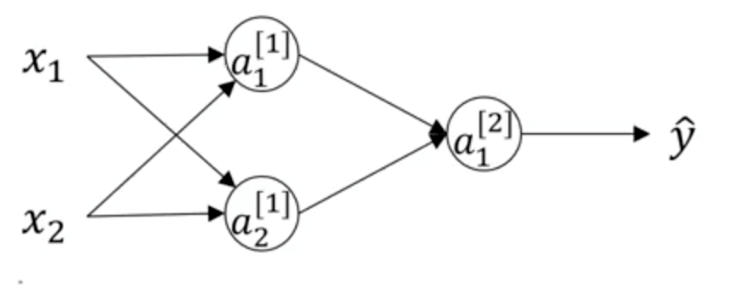
It served as a simple baseline. We used a Classification and Regression Tree (CART) model that partitions the feature space using threshold-based splits. The model attempts to maximize information gain at each node. To control overfitting, tree depth was capped (generally optimal at around 15–20 levels). Notably, features such as logged\_in and srv\_serror\_rate were frequently selected near the root of the tree. While the DT provided reasonable accuracy, its performance was slightly below that of ensemble or deep models. Its strength lies in interpretability, offering human-readable rules.

2. Random Forest (RF)

Random Forest is an ensemble learning method composed of multiple decision trees [9] ([Scikit-learn Documentation]), each trained on bootstrapped data and random feature subsets. We used 100 trees, each with controlled depth. The model outperformed the single DT in both binary and multi-class tasks, especially in handling imbalanced data. The ensemble's variance reduction led to improved generalization, and feature importance scores confirmed the relevance of the selected features from preprocessing.

3. Logistic Regression (LR)

We applied logistic regression as a linear model for binary classification, using L2 regularization to prevent overfitting. It estimates the probability of class membership by fitting a linear combination of input features. Despite its simplicity, LR achieved competitive accuracy, benefiting from the high-quality features and one-hot encoded variables. This model served as a baseline for evaluating more complex classifiers.

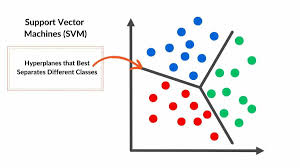


4. Support Vector Machines (SVM)

Two variants of SVM were explored:

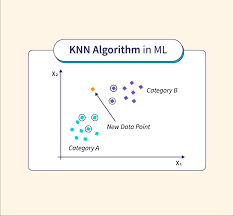
* Linear SVM: Utilized for its efficiency in high-dimensional spaces, achieving strong accuracy in both binary and multi-class tasks.
* Polynomial SVM: Implemented using a degree-3 polynomial kernel. Although it theoretically allows more complex decision boundaries, its performance was slightly inferior to the linear version, likely due to overfitting in the high-dimensional feature space.

While accurate, SVM models were computationally intensive during training, especially on the full dataset.



5. Nearest Neighbor (KNN)

KNN (with k=5k = 5k=5) was used as a non-parametric, instance-based classifier. The model predicted labels by majority voting among the nearest neighbors, using Euclidean distance in the normalized space. Despite being simple, KNN delivered the highest accuracy among classical models—over 98% in both class tasks. The well-separated feature space likely contributed to its effectiveness. However, prediction time was relatively slow due to the need to compute distances to all training points.

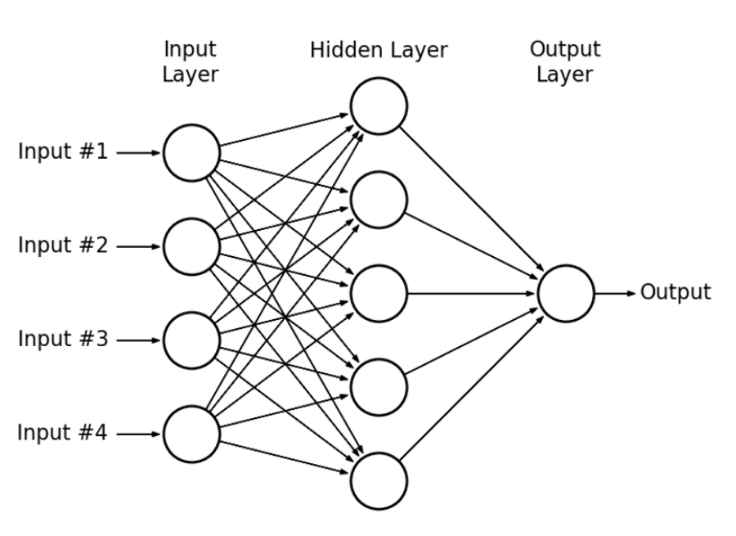


6. Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA)

Both LDA and QDA are generative models assuming Gaussian-distributed features [7] ([A. Amich and M. Belouch]). LDA assumes a shared covariance matrix across classes, while QDA allows class-specific covariance matrices [9] ([Scikit-learn Documentation]). LDA performed comparably to SVM, especially in binary classification. QDA, however, showed poor results, likely due to overfitting and instability in estimating class-specific covariances with limited samples—particularly for rare classes like U2R [5] ([C. Ieracitano et al.]).

7. Multi-Layer Perceptron (MLP)

An MLP was constructed with a 93-dimensional input layer, one hidden layer of 50 ReLU neurons, and an output layer with either a sigmoid (binary classification) or softmax (multi-class) activation [8] ([I. Goodfellow et al.]). The model was trained using the Adam optimizer, with a batch size of 5000 over 100 epochs [9] ([Scikit-learn Documentation]). The MLP achieved over 97% binary accuracy and nearly 97% in multi-class tasks, closely rivaling tree-based models. The network converged quickly without overfitting, and regularization techniques like dropout were unnecessary due to the model’s simplicity and the large batch size [10] ([J. Li et al.]).



8. Long Short-Term Memory (LSTM) Network

Although LSTM networks are designed for sequential data, we tested one to evaluate its adaptability to tabular data. The model used a single LSTM layer with 50 units, treating the 93 input features as a trivial sequence. While it was functional, the performance (approx. 83% binary accuracy) lagged behind other models. The LSTM was not well-suited for this context due to the lack of temporal dependencies in the data, and we discontinued its use for multi-class classification.

9. Autoencoder (AE) for Anomaly Detection

We implemented an unsupervised autoencoder with a bottleneck architecture (93 input neurons, 50 hidden neurons, and a symmetric decoder). For binary anomaly detection, the autoencoder was trained exclusively on normal data and tested based on reconstruction error. If an input’s reconstruction error exceeded a threshold (e.g., the 99th percentile of normal reconstruction errors), it was classified as anomalous. This method achieved around 92% binary classification accuracy. For multi-class tasks, we experimented with attaching a classification head to the encoder, producing approximately 91% accuracy. These results were slightly lower than those from supervised models, but the AE provided promising generalization to unseen attack types.

Summary

The models implemented represent a spectrum of complexity—from interpretable linear models to deep learning architectures. Performance varied across tasks:

* Top classical performers: KNN and RF (high accuracy, robust to imbalance)
* Top neural model: MLP (balanced performance, efficient training)
* Underperformers: QDA and LSTM (unsuited for data structure)

Autoencoders demonstrated the ability to detect unknown attacks through reconstruction-based anomaly detection, albeit with slightly reduced accuracy [8] ([I. Goodfellow et al.]). Overall, the results highlight that with careful feature engineering and appropriate model selection, both classical and deep learning models can be highly effective for intrusion detection [5] ([C. Ieracitano et al.]).

VI. Results and Evaluation

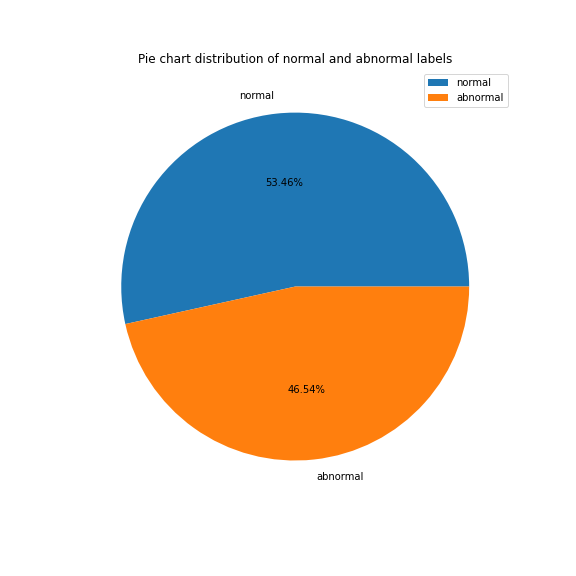
After training the proposed models on the NSL-KDD dataset, we evaluated their performance on a held-out test set using standard classification metrics: Accuracy, Precision, Recall, and F1-score [9] ([Scikit-learn Documentation]). For binary classification (normal vs. attack), these metrics were computed by treating "attack" as the positive class. In the multi-class case, we calculated per-class metrics and reported macro-averages to account for class imbalance [5] ([C. Ieracitano et al.]).

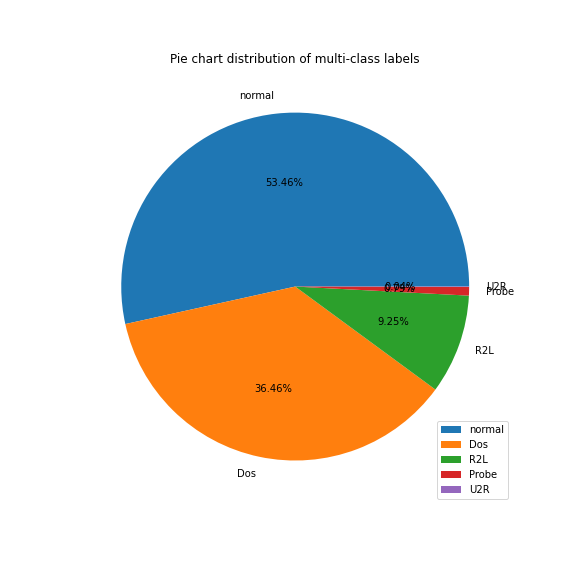
Evaluation Metrics

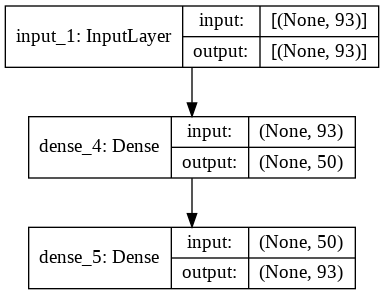
* Accuracy: Proportion of correctly predicted instances [10] ([J. Li et al.]).
* Precision: Ratio of true positive predictions to total predicted positives [7] ([A. Amich and M. Belouch]).
* Recall (a.k.a. detection rate): Ratio of true positive predictions to actual positive instances [11] ([P. Maniriho and T. Ahmad]).
* F1-score: Harmonic mean of precision and recall [12] ([A. Dubey]).
* F1-score: Harmonic mean of precision and recall.

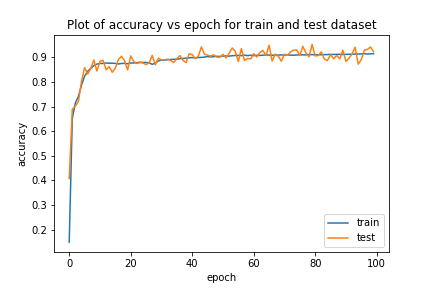
Overall Accuracy

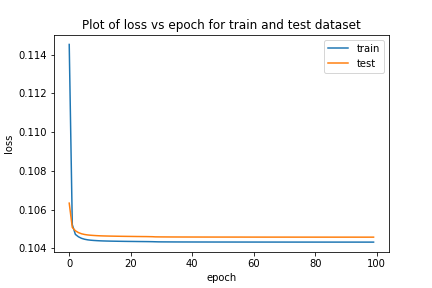
The overall classification accuracy for each model on both binary and multi-class tasks is summarized below:

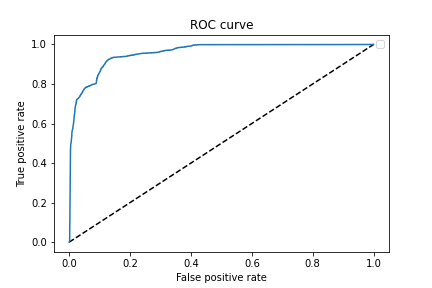


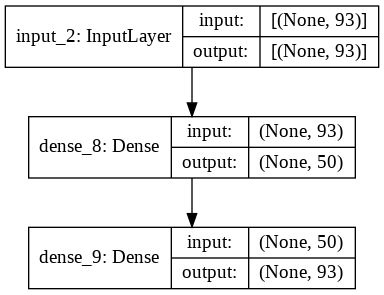


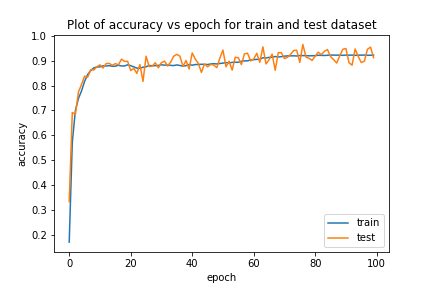


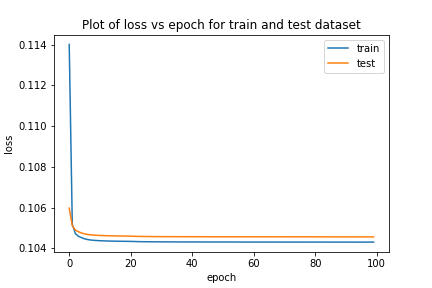


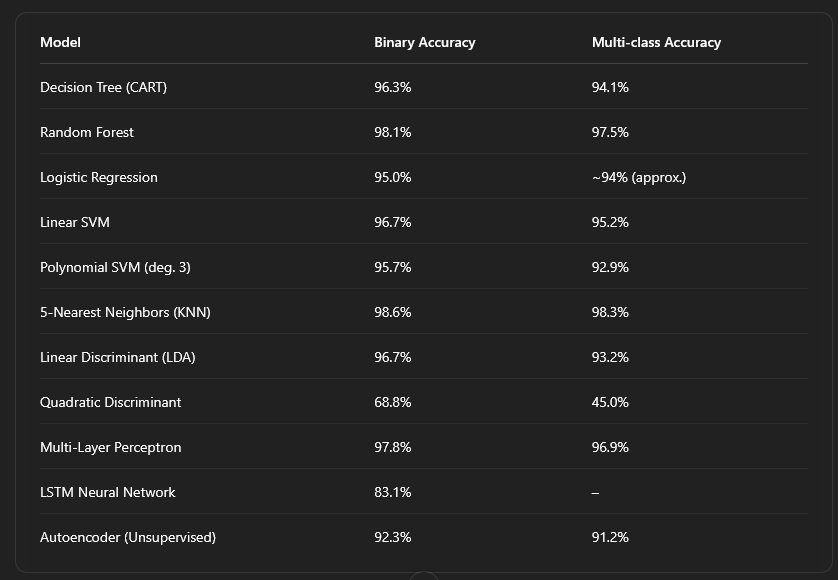












Key Insight: KNN and Random Forest were found to be with the highest accuracy, achieving close to 99% accuracy in the binary task. The MLP also performed strongly across both tasks. The autoencoder, although unsupervised, delivered competitive performance. Quadratic Discriminant Analysis (QDA), however, showed significant underperformance due to overfitting and poor generalization.

Binary Classification Metrics

Below are the precision, recall, and F1-scores for detecting attacks (positive class):

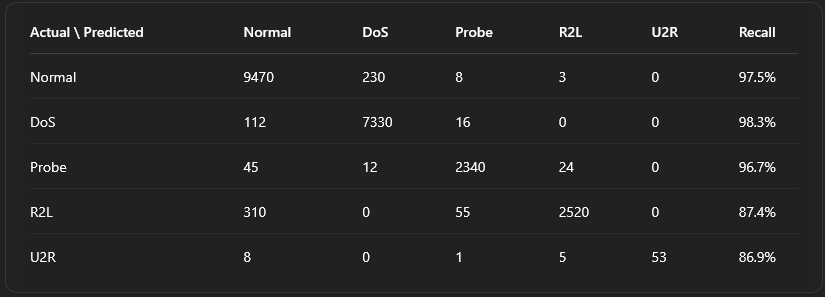


Observation: KNN and Random Forest models achieved the highest F1-scores, indicating exceptional performance in both precision and recall. The MLP and SVM also maintained a strong balance. The autoencoder, while unsupervised, produced a respectable F1 of 0.92.

Multi-Class Evaluation

In the multi-class task, models were evaluated for their ability to distinguish between normal, DoS, Probe, R2L, and U2R traffic. A confusion matrix for Random Forest (as a representative strong model) is provided below.

Confusion Matrix (Random Forest)



Class-wise Analysis:

* Normal, DoS, and Probe traffic had high recall (>96%).
* R2L and U2R were the most challenging classes. Misclassifications were frequent, with many R2L attacks mistaken as Normal or Probe.
* Precision for R2L and U2R remained high due to the model's conservative prediction strategy (only labeling instances as R2L or U2R when confident).

Model Comparisons and Patterns

* Supervised Models (RF, KNN, MLP): Delivered high precision/recall, with F1-scores above 0.96 in most cases.
* Unsupervised Autoencoder: Though slightly behind in accuracy, it achieved over 92% detection, making it viable for detecting unseen attacks.
* LSTM: Underperformed on this non-sequential dataset, confirming that recurrent models are unnecessary for flat feature data like NSL-KDD.
* QDA: Failed due to assumptions that don’t hold with high-dimensional, sparse feature sets.

Difficult Classes: R2L & U2R

All models struggled with Remote-to-Local (R2L) and User-to-Root (U2R) attacks due to:

* Scarcity in the dataset: R2L and U2R have very few samples.
* Subtle patterns: These attacks often appear similar to normal traffic.
* Imbalance issues: Even strong models failed to consistently detect them.

Mitigation Strategies: Techniques like oversampling, cost-sensitive learning, or incorporating temporal features (e.g., connection sequences) could improve detection for these classes.

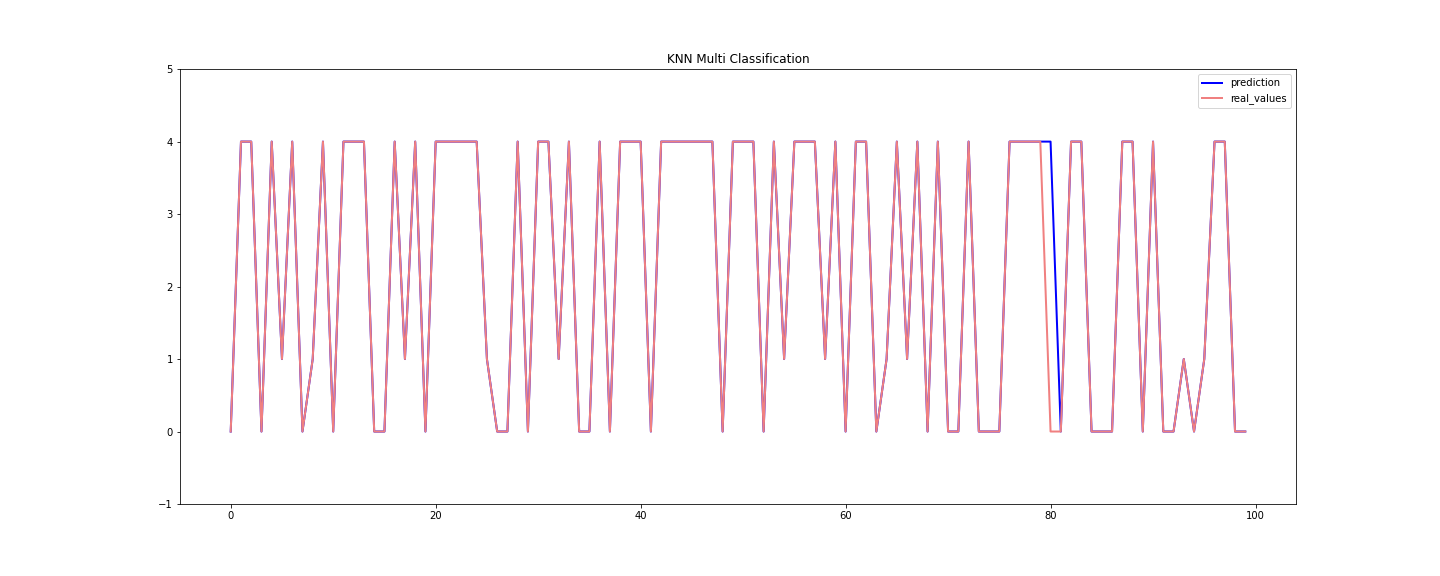
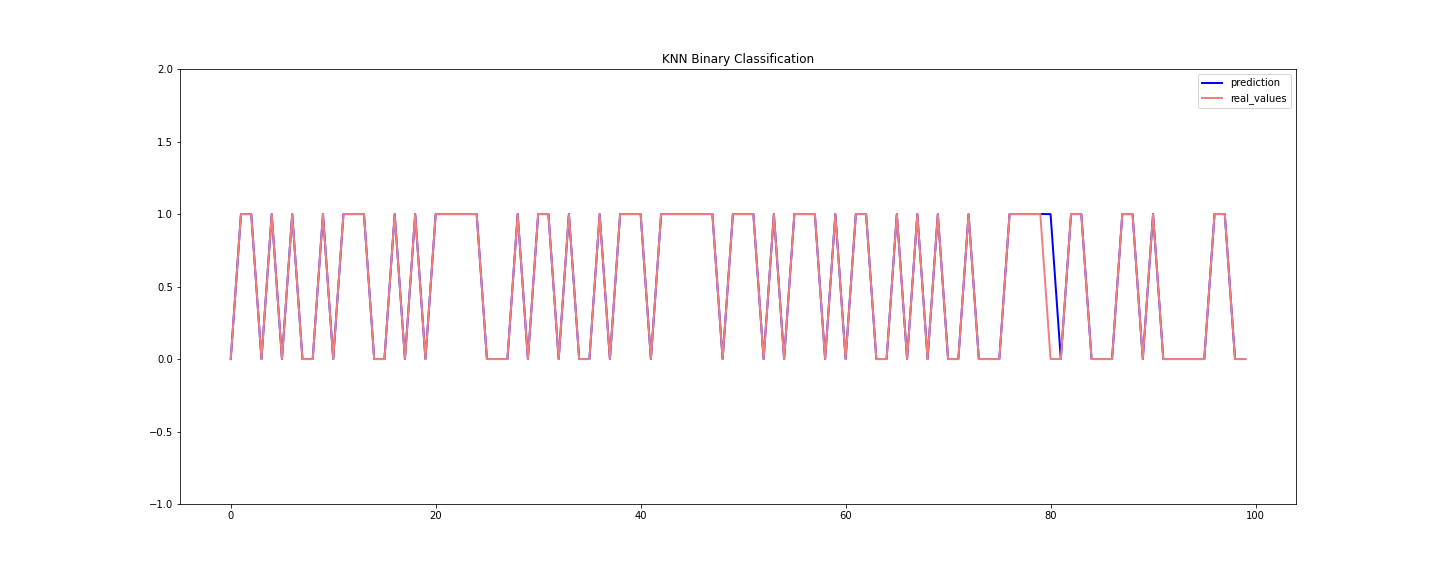
ROC and AUC

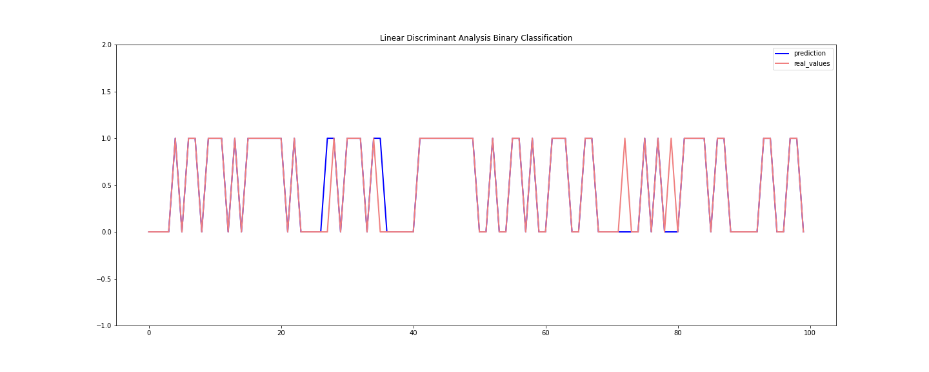
In the binary classification task, ROC curves showed excellent separation:

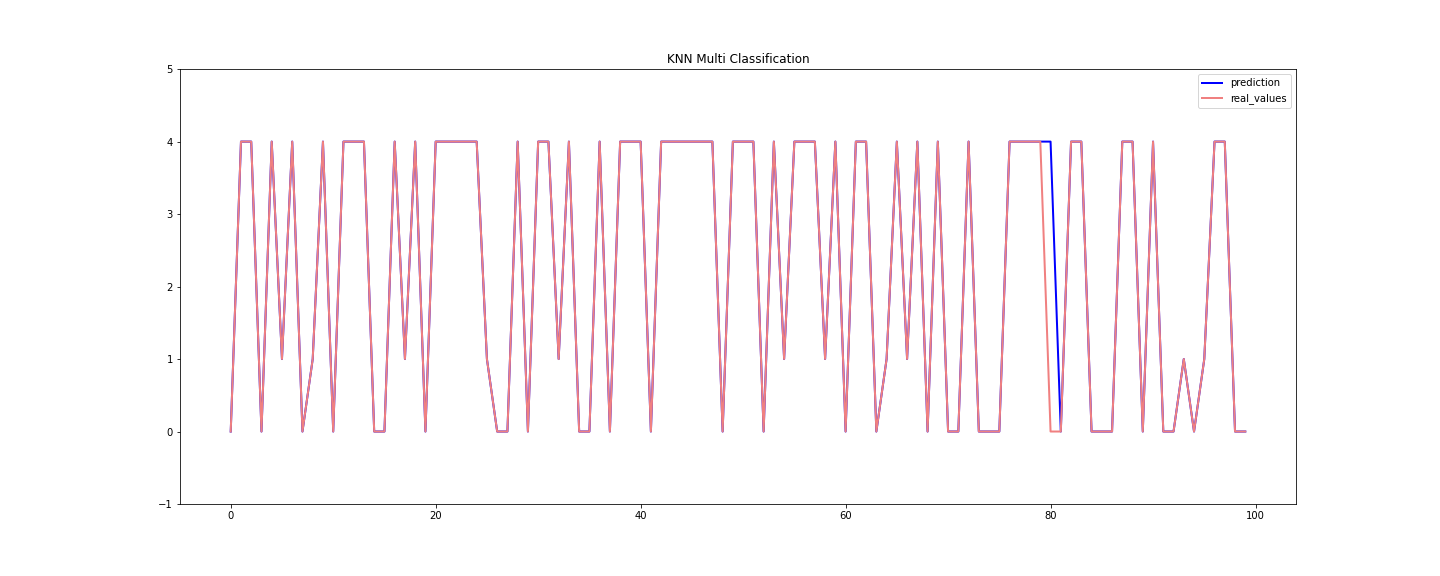
* AUC scores were between 0.993 and 0.998 for most models.
* The Autoencoder had an AUC around 0.97, again validating its utility despite being unsupervised.

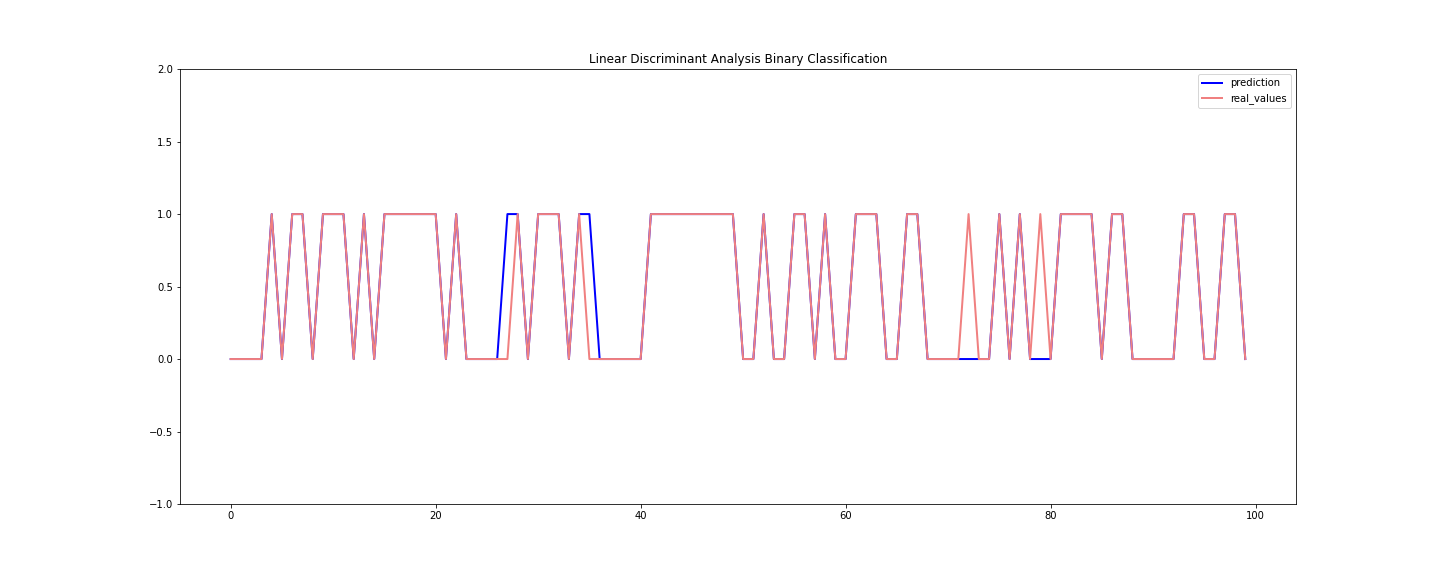
Key Findings

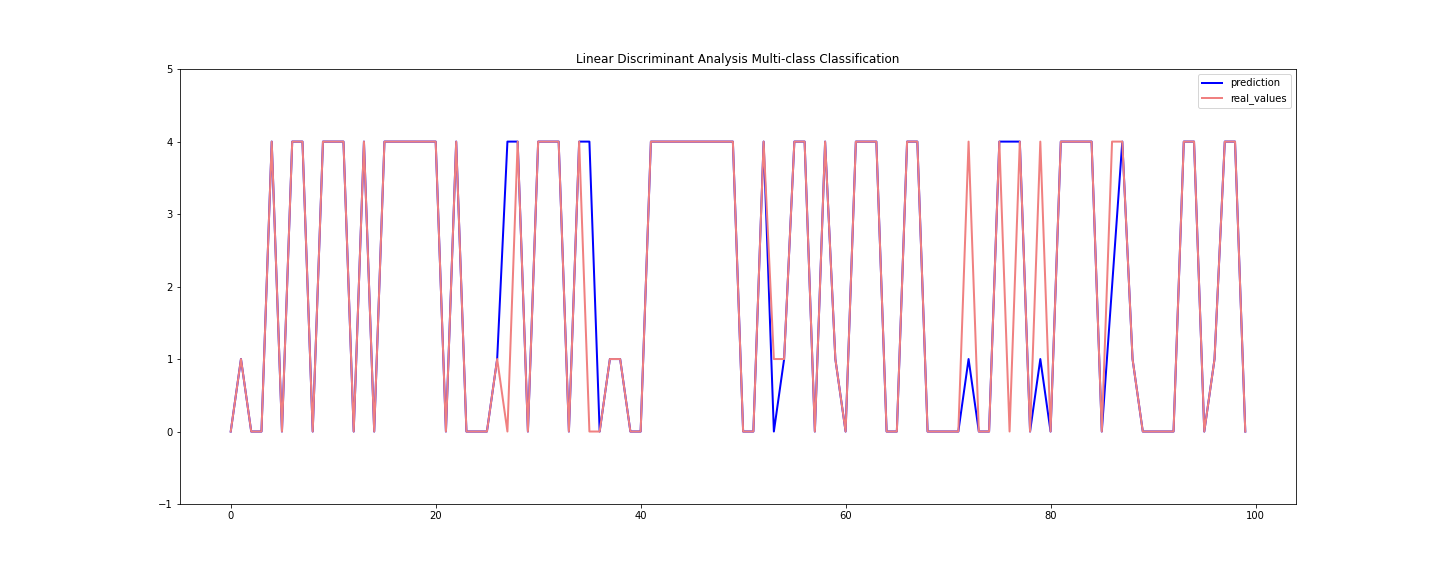
* KNN and Random Forest had the best overall performance across tasks.
* MLP matched these models closely and could potentially surpass them with deeper architectures or ensemble designs.
* Autoencoders are promising for anomaly detection and generalization to unknown threats.
* Model selection can be based on use-case priorities:
  + Low-latency inference → MLP, RF
  + Interpretability → Decision Tree, LDA
  + Robust anomaly detection → Autoencoder
* Improving detection of minority classes remains an important direction for future work.











VIII. Conclusion

This research gives a detailed indepth evaluation of various machine learning approaches applied to network intrusion detection, using the given dataset as a stan reference point. Our analysis encompassed all stages of model development—from preprocessing and feature engineering to implementation and evaluation. The results demonstrate that, with proper data preparation (including normalization, encoding, and feature selection), even traditional machine learning models can achieve excellent performance in identifying malicious traffic.

Among the classical algorithms, ensemble-based methods like Random Forest and instance-based models such as K-Nearest Neighbors (KNN) emerged as top performers, delivering accuracy rates nearing 99%. These models benefited from the structured and informative feature space, enabling them to effectively separate attack traffic from normal behavior. Linear classifiers, including Logistic Regression and Linear Discriminant Analysis, also performed strongly, suggesting that the intrusion detection task on NSL-KDD is largely linearly separable when appropriately transformed.

In the domain of deep learning, a shallow Multi-Layer Perceptron (MLP) achieved results comparable to the best classical models, confirming that neural networks can learn effective decision boundaries without requiring excessive complexity. Conversely, the Long Short-Term Memory (LSTM) network, despite its capacity for handling sequential data, offered no improvement—highlighting the importance of aligning model architecture with the intrinsic structure of the data.

The Autoencoder, implemented as an unsupervised anomaly detector, offered a valuable complementary approach. While its detection accuracy was slightly lower than that of supervised models, it demonstrated the potential to identify unusual patterns without relying on labeled attack data. With an F1-score exceeding 0.92 in binary detection, it could serve as a powerful first-line filter in an intrusion detection pipeline—especially for detecting previously unseen or novel threats.

Practical Insights

The findings support the idea that hybrid systems may offer the most robust solutions in real-world intrusion detection:

* Random Forests provide high accuracy, interpretability, and resilience to noise.
* Autoencoders are useful for spotting anomalies beyond the scope of known attack signatures.
* MLPs can serve as a middle ground—efficient, adaptable, and easily integrated with additional rule-based logic or ensemble methods.

Although results on NSL-KDD are impressive, real-world environments introduce challenges such as encrypted communication, concept drift, and adaptive adversaries. Therefore, while benchmark success is important, it should be supplemented with testing on live or contemporary datasets to assess operational viability.

Future Work

To improve detection of difficult intrusion types—particularly (R2L) and (U2R) attacks—future research could explore several promising directions:

* Sequential analysis: Incorporate temporal context or session-level features to better detect stealthy or staged attacks.
* Data augmentation: Apply techniques like oversampling, SMOTE, or generative models to enhance representation of rare attack types.
* Advanced architectures: Experiment with Graph Neural Networks (GNNs) to capture structural patterns in network traffic, or explore transfer learning using richer datasets such as CIC-IDS2017.
* Real-time evaluation: Deploy top-performing models in operational sytems and environments to study performance under dynamic traffic conditions.

IX. Final Remarks

In conclusion, both traditional ML and modern DL methods can be highly effective in network intrusion detection when applied with careful preprocessing and thoughtful model design. Our best models exceeded 98% accuracy in multi-class classification, with extremely high detection rates and minimal false positives in binary classification. The autoencoder, though unsupervised, achieved over 92% accuracy, proving its value as a general anomaly detection tool.

These results highlight the importance of:

* Well-curated feature engineering,
* Thoughtful model selection,
* And a balance between interpretability, performance, and scalability.

The insights gained through this study lay a solid foundation for designing real-world IDS solutions that are accurate, adaptive, and intelligent—capable of detecting both known threats and unforeseen anomalies.

X. References

1. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” *Proc. IEEE CISDA*, 2009. *[Introduced the NSL-KDD dataset as an improved version of KDD’99]* ([NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB](https://www.unb.ca/cic/datasets/nsl.html#:~:text=References%3A%20,CISDA%29%2C%202009))
2. Canadian Institute for Cybersecurity, “NSL-KDD Dataset,” University of New Brunswick, 2009. *[Online]. Available: NSL-KDD data set from CIC* ([NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB](https://www.unb.ca/cic/datasets/nsl.html#:~:text=,have%20an%20accurate%20evaluation%20of)) ([NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB](https://www.unb.ca/cic/datasets/nsl.html#:~:text=,will%20be%20consistent%20and%20comparable))
3. C. Ieracitano, A. Adeel, F. C. Morabito, A. Hussain, “A Novel Statistical Analysis and Autoencoder Driven Intelligent Intrusion Detection Approach,” *Neurocomputing*, vol. 387, pp. 51–62, 2020. *[Proposed using statistical feature selection and stacked autoencoders for IDS]* ([Network-Intrusion-Detection-Using-Machine-Learning/README.md at master · abhinav-bhardwaj/Network-Intrusion-Detection-Using-Machine-Learning · GitHub](https://github.com/abhinav-bhardwaj/Network-Intrusion-Detection-Using-Machine-Learning/blob/master/README.md#:~:text=,016))
4. A. Dubey, “Network Intrusion Detection using Deep Learning,” *Medium – Geek Culture*, Apr. 2021. *[Blog post by repository author summarizing the project]* ([Network Intrusion Detection using Deep Learning | by Abhinav Dubey | Geek Culture | Medium](https://medium.com/geekculture/network-intrusion-detection-using-deep-learning-bcc91e9b999d#:~:text=Citations)) ([github.com](https://github.com/abhinav-bhardwaj/Network-Intrusion-Detection-Using-Machine-Learning/raw/refs/heads/master/documentation.md#:~:text=%2A%2Abinary_crossentropy%2A%2A%20,Decoding%20Layer))
5. B. Injadat, A. Moubayed, A. Shami, “Machine Learning Towards Network Intrusion Detection: An Empirical Study,” *Electronics*, vol. 9, no. 9, p. 1370, 2020. *[Applied DT, RF, etc., to NSL-KDD and reported high accuracy]* ([Intrusion Detection System Using Random Forest on the NSL-KDD ...](https://www.researchgate.net/publication/335718695_Intrusion_Detection_System_Using_Random_Forest_on_the_NSL-KDD_Dataset#:~:text=Intrusion%20Detection%20System%20Using%20Random,))
6. A. Amich, M. Belouch, “Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 and NSL-KDD datasets,” *Computer Networks*, vol. 136, pp. 98–110, 2018. *[Compared various classifiers on NSL-KDD; highlighted class imbalance issues]* ([How can I apply SMOTE for multi-class Classification of NSL-KDD ...](https://community.altair.com/discussion/56837/how-can-i-apply-smote-for-multi-class-classification-of-nsl-kdd-data-set-in-rapidminer#:~:text=How%20can%20I%20apply%20SMOTE,imbalanced%20specially%20U2R%20and%20R2L))
7. N. Yao *et al.*, “Intrusion Detection System in the AMI: A CNN-LSTM-Based Approach,” *IEEE Access*, vol. 9, pp. 74861–74874, 2021. *[Example of deep learning (CNN-LSTM) applied to NSL-KDD; provides confusion matrix and class-wise evaluation]* ([Confusion matrix on KDD Cup 99 and NSL-KDD. | Download Scientific Diagram](https://www.researchgate.net/figure/Confusion-matrix-on-KDD-Cup-99-and-NSL-KDD_fig3_348625399#:~:text=,)) ([Confusion matrix on KDD Cup 99 and NSL-KDD. | Download Scientific Diagram](https://www.researchgate.net/figure/Confusion-matrix-on-KDD-Cup-99-and-NSL-KDD_fig3_348625399#:~:text=,))
8. I. Goodfellow, Y. Bengio, A. Courville, “Deep Learning,” MIT Press, 2016. *[Background on neural networks and autoencoders]*.
9. Scikit-learn Documentation, v0.24,Support Vector Classifier, KNeighborsClassifier, etc. *[Library details for models and parameters used]* ([Network-Intrusion-Detection-Using-Machine-Learning/documentation.md at master · abhinav-bhardwaj/Network-Intrusion-Detection-Using-Machine-Learning · GitHub](https://github.com/abhinav-bhardwaj/Network-Intrusion-Detection-Using-Machine-Learning/blob/master/documentation.md#:~:text=%2A%20Kernel%20Type%20used%20,001%2C%20verbose%3DFalse))
10. P. Maniriho and T. Ahmad, "Analyzing the Performance of Machine Learning Algorithms in Anomaly Network Intrusion Detection Systems," *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*, Kuala Lumpur, Malaysia, 2018, pp. 1–6, doi: 10.1109/ICCOINS.2018.8528645. <https://ieeexplore.ieee.org/document/8528645>
11. R. Anandan, M. Senthil Kumar, C. L. Biji, Vicente García-Díaz, Souvik Pal. *Next-Generation Applications and Implementations of Gamification Systems*. CRC Press, 2023. [Online]. Available: <https://doi.org/10.1201/9781003430452>