IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 4, April 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15680/IJIRSET.2024.1304045 |

Analysing Network Intrusion Detection Using Machine Learning

U. Akilesh Ragavendra, Dr. R. Nagarajan

UG Student, PG & Research Department of Computer Science, Sri Ramakrishna College of Arts and Science, Coimbatore, Tamil Nadu India

Assistant Professor, PG & Research Department of Computer Science, Sri Ramakrishna College of Arts and Science, Coimbatore, Tamil Nadu India

ABSTRACT: This study explores the effectiveness of various machine learning classifiers for intrusion detection in network security using the KDD Cup 1999 dataset. The dataset was preprocessed to handle duplicates, scale numeric features, and encode categorical variables. Feature selection was performed using a random forest-based method to identify the most relevant features for classification. Subsequently, four classifiers were trained and evaluated: Random Forest, Decision Tree, and two variations of K-Nearest Neighbors. Evaluation metrics, including accuracy scores, were computed to assess the performance of each classifier. The results indicate that Random Forest with selected features achieved the highest accuracy among the classifiers tested. This research contributes to the understanding of different machine learning techniques for intrusion detection and provides insights into their comparative performance, aiding in the selection of appropriate models for network security applications.

KEYWORDS: Machine learning algorithm, Networking, Cybersecurity, KDD cup 1999 dataset, Intrusion Detection, Data Preprocessing, Network Security, Feature Selection.

I. INTRODUCTION

Network intrusion detection is a crucial aspect of cybersecurity aimed at protecting computer networks and systems from unauthorised access, misuse, or malicious activities. With the increasing reliance on digital technologies in both personal and professional settings, the importance of effective intrusion detection has grown significantly.

The rapid expansion of the internet and interconnected devices has exposed networks to a wide range of cyber threats, including malware, phishing attacks, ransomware, and more. These threats pose serious risks to data security, privacy, and the integrity of critical systems. Data breaches resulting from successful intrusions can have severe financial and reputational consequences for organisations. The costs associated with data breaches include legal fees, regulatory fines, loss of intellectual property, damage to brand reputation, and potential lawsuits.

Many industries are subject to strict regulatory requirements regarding data protection and cybersecurity. Compliance with regulations such as the General Data Protection Regulation (GDPR), Health Insurance Portability and Accountability Act (HIPAA), and Payment Card Industry Data Security Standard (PCI DSS) necessitates robust intrusion detection mechanisms. Traditional security measures like firewalls and antivirus software are insufficient to combat sophisticated and evolving cyber threats. Network intrusion detection systems (NIDS) play a vital role in providing real-time monitoring and threat detection capabilities to identify suspicious activities and respond promptly to security incidents.

Machine learning algorithms offer promising capabilities for enhancing the effectiveness and efficiency of intrusion detection systems. By analysing vast amounts of network data, machine learning models can detect patterns, anomalies, and deviations indicative of malicious behaviour, enabling proactive threat mitigation. Leveraging advanced technologies like machine learning can strengthen intrusion detection capabilities and bolster overall cybersecurity posture in today's interconnected world.

II. OVERVIEW OF KDD CUP 1999 DATASET

The KDD Cup 1999 dataset is a benchmark dataset widely used for evaluating intrusion detection systems (IDS) and machine learning algorithms in the field of cybersecurity. It was created as part of the Third International Knowledge Discovery and Data Mining Tools Competition, known as KDD Cup 1999, which focused on the task of detecting intrusions in computer networks.

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15680/IJIRSET.2024.1304045 |

i. Dataset Source: The dataset was derived from the 1998 DARPA Intrusion Detection Evaluation Program, which collected network traffic data generated in a simulated military network environment.
ii. Data Characteristics: The dataset consists of a large collection of network connection records, each representing a TCP/IP connection. It contains a total of 4,898,431 records, with each record comprising 41 attributes.
iii. Attributes: The attributes include features such as protocol type, service, flag, duration of the connection, number of bytes transferred, and various other network traffic characteristics. Additionally, there is a target attribute labelled "label," indicating whether the connection is normal or represents a specific type of attack.
iv. Types of Attacks: The dataset covers several types of attacks, including denial of service (DoS), distributed denial of service (DDoS), probing (e.g., port scanning), user-to-root (U2R), and remote-to-local (R2L) attacks. These attacks represent different strategies employed by adversaries to compromise network security.
v. Data Preprocessing: Common preprocessing steps applied to the dataset include removing duplicate records, scaling numerical features, and encoding categorical variables for machine learning algorithms.
vi. Application in Research: The KDD Cup 1999 dataset has been extensively used in research and development of intrusion detection systems, anomaly detection algorithms, and machine learning models for cybersecurity. It serves as a benchmark for evaluating the performance and effectiveness of different approaches in detecting and mitigating network attacks.

In summary, the KDD Cup 1999 dataset is a valuable resource for studying network security and developing robust intrusion detection systems capable of identifying and responding to various types of cyber threats. Its widespread use in academic research and industry projects highlights its importance in advancing the field of cybersecurity.

III. DATA PREPROCESSING

Data preprocessing is a crucial step in any machine learning project, especially when dealing with real-world datasets like the KDD Cup 1999 dataset. In this section, we'll discuss the steps involved in preprocessing the data before building machine learning models.
Importing necessary libraries: First, we import the required libraries such as pandas, numpy, matplotlib, seaborn, and scikit-learn. These libraries provide functionalities for data manipulation, visualisation, preprocessing, and modelling.

1. **Loading and exploring the dataset:** We load the dataset using pandas' 'read_csv' function and examine its shape, columns, and data types using methods like 'shape', 'columns', and 'info'. This helps us understand the structure of the dataset and identify any missing values or inconsistencies.
2. **Handling duplicate records:** Duplicate records can skew the analysis and modelling process. We use the 'drop_duplicates' method to remove duplicate rows while keeping the first occurrence. This ensures that each observation in the dataset is unique.
3. **Data cleaning and preprocessing:** Data cleaning involves identifying and handling missing values, outliers, and inconsistencies in the dataset. Since the KDD Cup 1999 dataset has already been preprocessed to some extent, we may not encounter significant cleaning tasks. However, if necessary, we can perform additional cleaning steps based on the specific requirements of the project.

    a. **Scaling numeric features using MinMaxScaler:** Numeric features often have different scales, which can negatively impact the performance of machine learning models, particularly those sensitive to feature scales (e.g., distance-based algorithms). We use the 'MinMaxScaler' from scikit-learn to scale numeric features to a specified range, typically between 0 and 1.
    b. **Label encoding categorical features:** Machine learning models require numerical input, so we encode categorical features into numeric format using label encoding. This involves converting categorical labels into numeric integers. We use the 'LabelEncoder' from scikit-learn to perform this encoding for categorical features like protocol type, service, and flag.

By following these preprocessing steps, we ensure that the dataset is properly formatted and prepared for building machine learning models. This sets a solid foundation for training accurate and reliable intrusion detection models using the KDD Cup 1999 dataset.

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15680/IJIRSET.2024.1304045 |

IV. FEATURE SELECTION

RandomForestClassifier is a popular ensemble learning algorithm that is effective for both classification and regression tasks. It can also be utilised for feature selection, which involves identifying and selecting the most relevant features from the dataset. Here, we'll discuss how RandomForestClassifier can be used for feature selection and visualise the importance of selected features.

1. **Selecting relevant features**
   i. Importance of Feature Selection: In many real-world datasets, there may be numerous features, some of which may not contribute significantly to the predictive performance of the model. Feature selection helps in identifying and retaining only the most relevant features, leading to simpler, more interpretable models and potentially improving model performance.
   ii. Using SelectFromModel: RandomForestClassifier offers a convenient way to perform feature selection through the SelectFromModel class. This class selects features based on their importance scores computed during the training process.
   iii. Fitting SelectFromModel: We fit SelectFromModel with RandomForestClassifier using a specified number of estimators and random state. The model then computes the importance scores of each feature based on how much they contribute to reducing impurity in decision trees.
   iv. Selecting Features: After fitting the model, we can retrieve the selected features using the get_support() method, which returns a Boolean mask indicating the selected features.

2. **Visualising feature importance**
   i. Understanding Feature Importance: Feature importance represents the contribution of each feature to the predictive performance of the model. Higher importance scores indicate more influential features.
   ii. Plotting Feature Importance: We can visualise feature importance using various plotting techniques. A common approach is to create a horizontal bar plot where features are sorted based on their importance scores. This provides a clear visualisation of the most important features.
   iii. Interpreting the Plot: In the plot, features are listed on the y-axis, and their importance scores are represented by the length of the bars on the x-axis. The most important features appear at the top of the plot, while less important features are towards the bottom.
   iv. Making Informed Decisions: Feature importance plots help in making informed decisions about which features to include or exclude from the model. Features with higher importance scores are more likely to have a significant impact on the model's predictions and should be retained, while features with lower importance scores may be considered for removal if they do not contribute meaningfully to the model.

By leveraging RandomForestClassifier for feature selection and visualising feature importance, we can streamline the model-building process and create more effective predictive models.

V. MODEL TRAINING

In machine learning, model training is a crucial step where the algorithm learns patterns and relationships from the training data to make predictions on new, unseen data. Here, we'll discuss the process of model training, including splitting the dataset into training and testing sets, and training various classifiers such as RandomForestClassifier, DecisionTreeClassifier, and KNeighborsClassifier.

1. **Splitting the dataset into training and testing sets**
   i. Purpose of Splitting: Before training a model, it's essential to divide the dataset into two separate sets: training set and testing set. The training set is used to train the model, while the testing set is used to evaluate its performance on unseen data.
   ii. Train-Test Split: We use the 'train_test_split' function from the 'sklearn.model_selection' module to split the dataset. Typically, around 70-80% of the data is allocated to the training set, and the remaining 20-30% is assigned to the testing set.
   iii. Randomization and Reproducibility: It's important to set a random state to ensure that the data split is reproducible. This random state ensures that the data is shuffled before splitting, which helps in reducing bias in the model evaluation.

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15680/IJIRSET.2024.1304045 |

2. **Training various classifiers**

   i. RandomForestClassifier: RandomForestClassifier is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees. We initialise a RandomForestClassifier with specified parameters such as the number of estimators (trees) and the random state. Then, we fit the model to the training data.

   ii. DecisionTreeClassifier: DecisionTreeClassifier is a non-parametric supervised learning method used for classification tasks. It partitions the data into subsets based on feature values to create a tree-like structure of decisions. Similar to RandomForestClassifier, we initialise a DecisionTreeClassifier and fit it to the training data.

   iii. KNeighborsClassifier: KNeighborsClassifier is a simple, instance-based learning algorithm used for classification tasks. It predicts the class of a data point by looking at the 'k' closest labelled data points and taking a majority vote. We specify the number of neighbors (k) and initialise a KNeighborsClassifier. Then, we train the classifier on the training data.

By training these classifiers on the training data, we enable them to learn from the patterns present in the features and their corresponding labels. Once trained, we can evaluate the performance of each model on the testing set to assess their accuracy and generalisation capabilities.

VI. MODEL EVALUATION

After training the machine learning models, it's crucial to assess their performance to understand how well they generalise to unseen data. Model evaluation involves measuring various metrics such as accuracy scores, confusion matrices, and classification reports.

- **Accuracy scores:** Accuracy is one of the most commonly used metrics to evaluate classification models. It measures the proportion of correctly classified instances out of the total instances.In our case, we calculate accuracy scores for each trained model (RandomForestClassifier, DecisionTreeClassifier, KNeighborsClassifier) using the ‘accuracy_score’ function from the ‘sklearn.metrics’ module. The accuracy score is computed by comparing the predicted labels with the actual labels from the testing set.
- **Confusion matrices:** A confusion matrix provides a more detailed breakdown of the model's performance by showing the number of true positives, true negatives, false positives, and false negatives. We can generate confusion matrices for each model using the ‘confusion_matrix’ function from the ‘sklearn.metrics’ module. This matrix helps in identifying the types of errors made by the classifier.
- **Classification reports:** Classification reports provide a summary of various classification metrics such as precision, recall, F1-score, and support for each class in the dataset. The ‘classification_report’ function from the ‘sklearn.metrics’ module can be used to generate classification reports for each model. It computes these metrics for each class and provides an overall summary.

By evaluating the performance of the trained models using these metrics, we gain insights into their strengths and weaknesses. This evaluation process helps in selecting the best-performing model for the given task and identifying areas for improvement, such as fine-tuning hyperparameters or exploring different feature sets. Additionally, it assists in making informed decisions about deploying the model in real-world applications.

VII. COMPARISON OF MODELS

When working on a machine learning task, it's essential to compare the performance of different classifiers to identify the most suitable model for the problem at hand. In this case, we'll compare the performance of three classifiers: Random Forest, Decision Tree, and K-Nearest Neighbors (KNN).

Accuracy scores provide a quantitative measure of how well each classifier performs on the given dataset. We compute accuracy scores for each model by comparing their predicted labels with the actual labels from the testing set. By analysing the accuracy scores of Random Forest, Decision Tree, and KNN, we can determine which classifier achieves the highest level of accuracy and generalisation performance. Additionally, comparing accuracy scores helps in understanding the relative strengths and weaknesses of each classifier in terms of classification accuracy.

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 13, Issue 4, April 2024 ||

| DOI:10.15680/IJIRSET.2024.1304045 |

VIII. CONCLUSION

In this study, we explored the performance of different machine learning classifiers on the KDD Cup 1999 dataset, which contains network intrusion detection data. We conducted experiments using three classifiers: Random Forest, Decision Tree, and K-Nearest Neighbors (KNN). We began by preprocessing the dataset, which involved removing duplicate entries, scaling numeric features, and encoding categorical variables. Subsequently, we split the data into training and testing sets and trained each classifier using various subsets of features.

Our analysis revealed that Random Forest consistently outperformed the other classifiers in terms of accuracy scores. It achieved the highest accuracy among the three classifiers, indicating its effectiveness in classifying network intrusion data. Decision Tree and KNN also showed respectable performance but generally lagged behind Random Forest.

The findings of this study have several implications for network intrusion detection and cybersecurity. Random Forest, being the most accurate classifier in our experiments, can be a valuable tool for identifying malicious network activities and potential cyber threats. Its robustness and ability to handle large datasets make it suitable for real-world applications in cybersecurity operations.
However, the study also highlights the importance of considering other factors such as computational efficiency and interpretability when selecting a classifier for network intrusion detection systems. Decision Trees offer transparency and interpretability, which can be crucial for understanding the underlying patterns in network traffic. On the other hand, KNN provides a simple and intuitive approach but may suffer from computational overhead, especially with large datasets.

IX. FUTURE DIRECTIONS AND POTENTIAL IMPROVEMENTS

Moving forward, future research can explore several avenues to enhance the performance of classifiers in network intrusion detection. This includes investigating advanced ensemble methods, feature engineering techniques, and anomaly detection algorithms tailored specifically for network traffic analysis. Additionally, incorporating domain knowledge and expert insights into the model development process can further improve the accuracy and efficacy of intrusion detection systems. Furthermore, exploring techniques for handling imbalanced datasets and adapting to evolving cybersecurity threats remains a critical area for future research in this domain.
Overall, this study lays the groundwork for leveraging machine learning techniques to enhance network security and mitigate cyber threats, paving the way for more sophisticated and effective intrusion detection systems in the future.

REFERENCES

1. https://www.researchgate.net/publication/319376480_A_review_paper_on_IDS_classification_using_KDD_99_and_NSL_KDD_dataset_in_WEKA
2. https://ieeexplore.ieee.org/document/8442909
3. https://ieeexplore.ieee.org/document/10141704
4. https://jpinfotech.org/intrusion-detection-system-using-improved-convolution-neural-network/
5. https://www.researchgate.net/publication/347956951_A_Systematic_Literature_Review_of_Intrusion_Detection_System_for_Network_Security_Research_Trends_Datasets_and_Methods
6. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00448-4
7. https://www.sciencedirect.com/science/article/pii/S2590005623000310
8. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00382-x
9. https://link.springer.com/article/10.1007/s44196-021-00047-4
10. https://peerj.com/articles/cs-1648/

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details