**NETWORK INTRUSION DETETCION SYSTEM USING MACHINE LEARNING**

Contents

[Introduction: 3](#_Toc139106473)

[Dataset Description: 3](#_Toc139106474)

[Objectives: 4](#_Toc139106475)

[Expected Outcomes: 4](#_Toc139106476)

[Project Management: 5](#_Toc139106477)

[Related Work: 5](#_Toc139106478)

[Methodology: 6](#_Toc139106479)

[Data Preprocessing: 6](#_Toc139106480)

[Data Modelling: 7](#_Toc139106481)

[Results & Discussion: 8](#_Toc139106482)

[Conclusion: 10](#_Toc139106483)

[Reflection & Future Work: 11](#_Toc139106484)

[References: 12](#_Toc139106485)

# List of Figures

[Figure 1: Project timeline 4](#_Toc139106327)

[Figure 2: Checking data for missing values 5](#_Toc139106328)

[Figure 3: Checking data for data type mismatch 5](#_Toc139106329)

[Figure 4: Removing outliers from the dataset 5](#_Toc139106330)

[Figure 5: Performing label encoding on categorical columns 6](#_Toc139106331)

[Figure 6: Checking dataset for misbalancing issue 6](#_Toc139106332)

[Figure 7: Differentiating features & target and dividing data into train and test sets 7](#_Toc139106333)

[Figure 8: Initializing model and training parameters for tuning 7](#_Toc139106334)

[Figure 9: Getting best set of parameters for the model 7](#_Toc139106335)

[Figure 10: Training the model with best parameters 7](#_Toc139106336)

[Figure 11: Performance of model trained on test set 8](#_Toc139106337)

[Figure 12: Importance of variables in model training 8](#_Toc139106338)

[Figure 13: Cross validation of the trained model 9](#_Toc139106339)

# Introduction:

Intrusion Detection Systems (IDS) are very important for keeping computer networks safe because they find and stop bad things from happening. Traditional rule-based IDS often find it hard to keep up with how quickly online threats change. Because of this, there has been a growing interest in using machine learning methods to make intrusion detection systems that are more reliable and can change over time. This study focuses on applying the Random Forest classification model to an intrusion detection dataset. This is followed by fine-tuning using grid search cross-validation, performance evaluation on an unseen test set, and validation through 5-fold cross-validation. The data set used in this study is a collection of network flow data that includes information about both the network and the hosts. Before using the Random Forest model, methods for preprocessing the data are used to make sure the data is good and can be used by machine learning algorithms. This preparation may include dealing with missing values, encoding categorical variables, and scaling numerical features. By cleaning and preparing the information, we make sure that the model gets good data to learn from. The Random Forest classification model is often used to find intrusions because it can handle complicated, high-dimensional datasets and can't be over fit. The model builds a group of decision trees, each of which is trained on a different part of the dataset and a different set of features. The Random Forest model is good at spotting intrusions because it combines the guesses of many trees. A grid search cross-validation method is used to find the best mix of hyper parameters to improve the performance of the Random Forest model. Systematically, hyper parameters like the number of trees (n\_estimators) and the highest depth of trees (max\_depth) are changed, and the performance of the model is measured by metrics like accuracy, precision, recall, and F1-score. Grid search cross-validation makes sure that the model's hyper parameters are fine-tuned for the best possible results on the given dataset. After the Random Forest model has been fine-tuned, an unknown test set is used to measure how well it works. This separate test set is made up of data from the real world that the model hasn't seen during training or when the hyper parameters were being tuned. The review gives information about how well the model can generalize and how well it can find intrusions. A 5-fold cross-validation method is also used to check the success of the model and see how stable it is. Cross-validation splits the dataset into five subsets, or "folds." The model is then trained and tested on different combinations of folds. This process helps figure out how well the model works over many versions and gives a more accurate assessment of how well it works. The Random Forest model is also easier to understand when you get changing importance measures. By looking at the feature importance scores, we can figure out how important different features are to the way the model makes decisions. This information helps find important network-level and host-level factors that play a big role in detecting intrusions. Overall, this study applies the Random Forest classification model to an intrusion detection dataset, performs fine-tuning using grid search cross-validation, evaluates the model's performance on an unseen test set, and confirms its usefulness using 5-fold cross-validation. Through data preprocessing and feature importance analysis, the model's ability to accurately identify intrusions and its readability are improved. This helps to move machine learning-based intrusion detection systems forwards.

# Dataset Description:

The auditing dataset is made up of many different intrusions that were simulated in a military network setting. The data can be found at <https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection>. The goal of this simulation was to get raw TCP/IP dump data from a network that acted like a normal US Air Force local area network (LAN). Multiple strikes were made on the simulated LAN, which was made to look like a real network. In this set of data, a "connection" is a series of TCP packets that begin and stop within a certain amount of time. During this time, data moves between a source IP address and a target IP address according to a well-defined protocol. Each link is marked as "normal" or "attack," and a certain type of attack is given to each. The dataset has information about these links. Each connection record has about 100 bytes of information. For every TCP/IP link, both normal and attack data are used to get a total of 41 features. These features contain a combination of quantitative (numerical) and qualitative (categorical) variables. Only three of the 41 features are qualitative, while the other 38 are numeric. These features record different parts and traits of the links, which is useful for finding and analyzing intrusions. The class variable in the dataset has only two possible values: "Normal" or "Anomalous." Each connection falls into one of these two groups. This classification makes it possible to figure out how a network is supposed to work and what kinds of attacks happened in the virtual environment. The dataset has 42 columns (called "features") and 25,192 rows (called "connection records"). This gives a lot of information that can be used for research and modeling. When there are both numerical and categorical variables, you can use different data analysis methods and machine learning algorithms to find trends, find outliers, and make good intrusion detection models. For accurate data analysis and making reliable models for intrusion detection, it is important to know what the features in the dataset are and how they are spread out.

# Objectives:

The following are the objectives of this study:

* To use a sample for intruder detection with the Random Forest classification model.
* To use grid search cross-validation to fine-tune the Random Forest model.
* To figure out how well the model works on a test set that hasn't been seen.
* To check how well the model works by using 5-fold cross-validation.
* To prepare the data by dealing with missing values, storing category factors, and scaling numerical features.
* To look at the numbers for how important each feature is and figure out what the most important network-level and host-level factors are for finding bugs.
* To make the machine learning-based intruder detection model work better and be easier to understand.
* To help make intruder monitoring systems that use machine learning better.

# Expected Outcomes:

The expected results of this study are a trained Random Forest classification model that can accurately find intrusions in a network, fine-tuned hyper parameters for the model that optimize its performance on the given dataset, evaluation results showing the model's high accuracy in classifying network connections as "Normal" or "Anomalous" on an unseen test set, validation of the model's effectiveness and stability through 5-fold cross-validation, and demonstrations of the model's effectiveness and stability. The study also wants to come up with data that has already been processed and is ready for analysis and modeling. This means that missing values will be filled in, classification variables will be coded, and outliers will be taken out. Using feature importance analysis to find the most important features will give us information about network-level and host-level factors that help spot intrusions. The study also hopes that data preparation and feature value analysis will make the intruder detection model more accurate and easier to understand. In the end, the goal of the study is to help machine learning-based intruder detection systems get better by showing how well the Random Forest classification model works.

# Project Management:

The project timeline that is being followed in this project is attached below from week 1 till last week which is week 16:



Figure : Project timeline

# Related Work:

Intrusion detection is very important for finding out when attackers are doing strange things on a target system [1]. Network Intrusion Detection Systems (NIDS) have two main types of detection: signature-based detection and anomaly-based detection. Garcia-Teodoro et al. [2] talked about the difficulties that come with different kinds of anomaly-based attack detection. Sinclair et al. [3] made an application that used machine learning techniques like Genetic Algorithms and decision trees to improve domain knowledge and make rules for an intrusion detection expert system. This was done to get around the problems of manually analyzing huge amounts of data to find strange network connections. Distributed Denial of Service (DDoS) attacks have also been found with the help of machine learning. Ashraf et al. [4] used machine learning methods to find DDoS attacks in software-defined networks, while Suresh et al. [5] compared different machine learning algorithms to improve the accuracy of finding DDoS attacks. The use of machine learning has led to the development of ensemble learning methods for NIDS. Both homogeneous and diverse groups have been looked at by researchers. By putting together groups of similar algorithms, the methods of bagging [6, 7] and boosting have been used to find network intrusions. On the other hand, stacked and mixed models [14] use groups of classifiers that are different from each other. Aburomman et al. [13] did a thorough review of ensemble-based NIDS, looking at homogeneous, heterogeneous, and mixed methods and comparing how well they worked on different datasets. Along with ensemble methods, feature reduction approaches [8, 9] have also been shown to improve the performance of intruder detection. When looking for DDoS attacks, both homogeneous and heterogeneous ensemble models have been used [10-12]. In this study, we want to use models from different classification families to make a new ensemble method. Also, we try to keep the number of data features that directly affect DDoS attempts to a minimum. The goal of the suggested ensemble is to get better identification accuracy and a lower rate of false positives compared to what has already been done.

# Methodology:

## Data Preprocessing:

Data preprocessing is an important part of getting a sample ready for analysis and modeling. In this part, we'll talk about what happens during the preprocessing stage:

* Missing Value Check: First, we look at the dataset to see if any records are missing. There were no missing numbers in the dataset, which is good news. This makes sure that we have full data for all the features, so we don't have to use imputation or figure out what to do with missing numbers.



Figure 2: Checking data for missing values

* Data Type Check: Next, we check to see if any of the data types in the list don't match. But no mismatches between data types were found. This means that the features in the dataset have been given the right types of data, whether they are numbers or based on categories. Using the right data types makes sure that the data is analyzed and modeled in the right way.



Figure 3: Checking data for data type mismatch

* Outlier Removal with Z-Score: We use the Z-score method to deal with possible outliers. With this method, we can find observations that are very different from the average of their traits. By figuring out the Z-score for each data point, we can see if it goes above or below a certain cutoff (which is usually set at Z-score > 3 or -3) and if it might be an outlier. These outliers can cause noise, which can affect how well machine learning models work. So, any outliers that are found are taken out of the information to make the next steps of analysis and modeling more accurate.

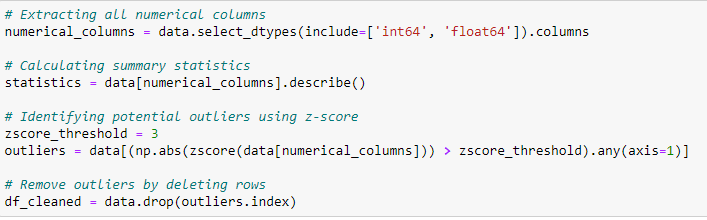


Figure 4: Removing outliers from the dataset

* Label Encoding for Categorical Columns: Since the dataset has categorical columns, we need to change them to numbers so that the machine learning programs can work with them more easily. To do this, we use label encoding, which gives each group in a categorical variable a unique numeric label. This makes it possible for the programs to understand and use the categorical data well when modeling. We make sure that the next steps of analysis and models will work with the data by turning the categorical columns into numerical ones.

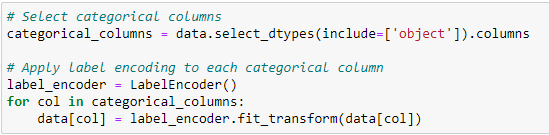


Figure 5: Performing label encoding on categorical columns

* Check for Unbalance in the Target Column: It is important to look at the balance of the target column to see if there are any problems with the data. When one class of data is much bigger than the other, it can throw off how well a model works. But when the goal column was looked at, it was found that there were no major data imbalance problems. This means that both the "Normal" and "Anomalous" classes are well represented in the dataset. This makes sure that the goal variable is well represented when it is analyzed and modeled in the future.

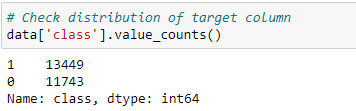


Figure 6: Checking dataset for misbalancing issue

By handling missing values, checking data types, removing outliers, converting categorical columns to numbers, and checking for target column imbalance, we make sure that the dataset is ready for further analysis and the creation of an effective intrusion detection model.

## Data Modelling:

In this part, we talk about the steps of how the Random Forest classifier is used to build the data for the intruder detection system. First, we split the dataset's features (which are independent variables) from its target values (which are dependent variables). The target numbers show whether a link is "Normal" or "Anomalous." The features tell you about the TCP/IP connections. This separation lets us train the model with the help of the features and test how well it does at guessing the goal values. To test how well the model works, we split the data into training and testing sets. A 65%-35% split is used, where 65% of the data is used to train the model and the other 35% is used to test how well the model works on data it hasn't seen before.

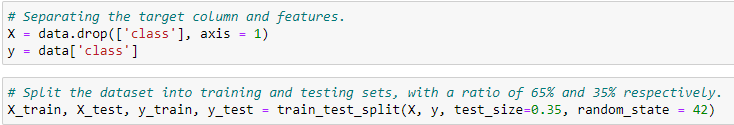


Figure 7: Differentiating features & target and dividing data into train and test sets

We used GridSearchCV to figure out the best set of parameters for the Random Forest model. GridSearchCV does a thorough search over a given parameter grid and checks the model using different parameter combos. We choose the following grid of parameters for the Random Forest model:

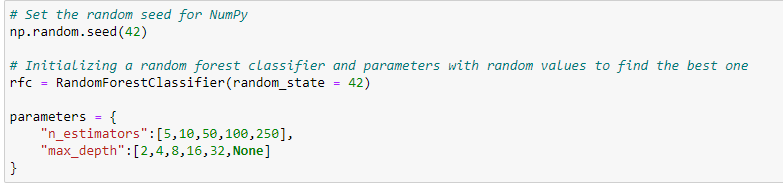


Figure 8: Initializing model and training parameters for tuning

GridSearchCV looks at every possible mix of n\_estimators and max\_depth numbers to find the best parameters. When the search was done, the best values found were:



Figure 9: Getting best set of parameters for the model

# Results & Discussion:

The best values from the GridSearchCV are then used to train the Random Forest model. The figure attached below shows the trained model process:

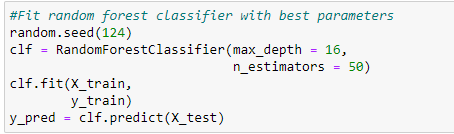


Figure 10: Training the model with best parameters

We use the trained model to see how well it works on the test set. With an accuracy of 0.9975051031980041, the model can describe links as "Normal" or "Anomalous" with a high level of accuracy. The confusion matrix is a more thorough way to measure success. The confusion matrix shows how many true negatives, fake positives, false negatives, and true positives there are. We have only 22 miss classified observations in this case which are very minimal.

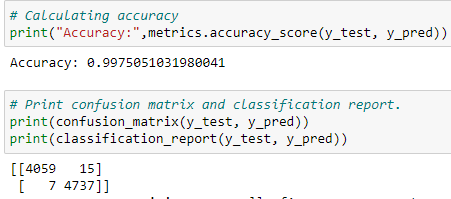


Figure 11: Performance of model trained on test set

The numbers for how important each feature is come from the Random Forest model that has been trained. The feature importance scores show how important each trait is in figuring out what the goal variable will be. The top five traits that are most important are src\_bytes, flag, dst\_bytes, same\_srv\_rate and serror\_rate. These features play a big part in how the model decides what to do and can give us information about what causes network attacks. The graph showing this information is attached below:

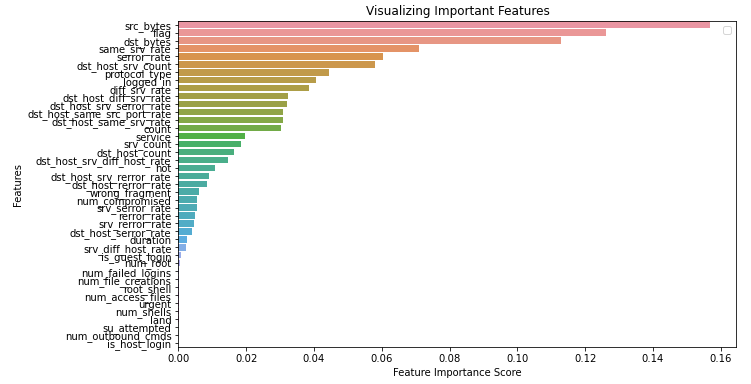


Figure 12: Importance of variables in model training

Lastly, we use cross-validation with 5 folds to check how well the Random Forest model works. Cross-validation helps figure out how well the model works on new data and how stable it is. The mean accuracy across all 5 folds is 0.9974, which shows that the model works well and consistently.

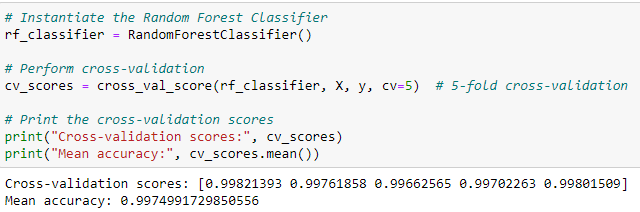


Figure 13: Cross validation of the trained model

We made a Random Forest classifier for intruder detection by following these steps for modeling: separating features, splitting data, setting hyper parameters with GridSearchCV, training and evaluating the model, figuring out how important each feature is, and cross-validating. The results show that the system is accurate, reliable, and that certain traits are important for spotting network intrusions.

# Conclusion:

In conclusion, the use of machine learning methods, especially the Random Forest classification model, in the area of intrusion detection has shown positive results. During this study, we used a dataset with simulated intrusions in a military network setting to make a good method for finding intrusions. We were able to start modeling after we cleaned up the data and made sure it was good. There were no missing numbers or mismatched data types in the dataset, which shows that it is reliable. The Z-score method was used to find and get rid of outliers, which made the model work better. Label encoding was used to turn the categorical columns into numbers, which let these features be used in the Random Forest model. Also, we made sure that the goal column didn't have any problems with data mismatch, so that training and evaluating the model would be fair. During the modeling process, the first thing that was done was to separate the features from the goal values. Then, the data was split into training and testing sets. We wanted to get a good idea of how well the model worked on data it hadn't seen before, so we set aside 65% of the data for training the model and 35% for testing. GridSearchCV was used to tune the Random Forest model's hyper parameters, which led to the best values being chosen. The model was then trained with these best values and tested on the test set. With an accuracy of 0.9975, the results of the performance review were very good. The confusion matrix showed that the model could correctly put links into two groups: "Normal" or "Anomalous." Also, the feature priority study gave useful information about which traits are most important for intrusion detection. Features like src\_bytes, flag, dst\_bytes, same\_srv\_rate, and serror\_rate were especially helpful in finding network attacks. We used 5 folds of cross-validation to make sure the model was strong and could be used in different situations. With a mean accuracy of 0.9974 across all folds, the cross-validation scores always showed high accuracy. This made the Random Forest model for finding intrusions even more reliable and stable. The results of this study show how well machine learning, and in particular the Random Forest classifier, can be used to find attacks in network settings. By using the features of the dataset and using the right preprocessing methods, we were able to create a model that can correctly spot network attacks.

# Reflection & Future Work:

The results of this study show that machine learning can be used to find network attacks using the Random Forest classification model. The program did a good job of identifying network links as "Normal" or "Anomalous," which shows that it can consistently find dangers. The preprocessing steps, such as dealing with missing values, checking the data type, removing outliers, and labeling labels, made sure that the dataset was clean and ready to be analyzed. The key factors that affect breach detection, such as src\_bytes, flag, dst\_bytes, same\_srv\_rate, and serror\_rate, could be better understood with the help of feature importance analysis. The results of the 5-fold cross-validation showed that the model is stable and accurate across different groups of the data. Overall, this study shows how useful machine learning-based intruder detection systems can be. It also helps to move the field forward by showing how well the Random Forest model works. But more study is needed to look into more models, ensemble methods, and ways to reduce the number of features to improve the accuracy of recognition and cut down on false positives when spotting attacks.

In future work, it would be helpful to look into other machine learning models and group methods for attack detection and compare how well they work with the Random Forest model. The model's speed could also be improved by looking into ways to cut down on features.

# References:

1. Mukherjee, Biswanath, L. Todd Heberlein, and Karl N. Levitt. "Network intrusion detection." IEEE network 8.3 (1994): 26-41.
2. Garcia-Teodoro, Pedro, et al. "Anomaly-based network intrusion detection: Techniques, systems and challenges." computers & security 28.1-2 (2009): 18-28.
3. Sinclair, Chris, Lyn Pierce, and Sara Matzner. "An application of machine learning to network intrusion detection." Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99). IEEE, 1999.
4. Ashraf, Javed, and Seemab Latif. "Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques." 2014 National Software Engineering Conference. IEEE, 2014.
5. Suresh, Manjula, and R. Anitha. "Evaluating machine learning algorithms for detecting DDoS attacks." International Conference on Network Security and Applications. Springer, Berlin, Heidelberg, 2011.
6. Syarif, Iwan, et al. "Application of bagging, boosting and stacking to intrusion detection." International Workshop on Machine Learning and Data Mining in Pattern Recognition. Springer, Berlin, Heidelberg, 2012.
7. Gaikwad, D. P., and Ravindra C. Thool. "Intrusion detection system using bagging ensemble method of machine learning." 2015 International Conference on Computing Communication Control and Automation. IEEE, 2015.
8. Borji, Ali. "Combining heterogeneous classifiers for network intrusion detection." Annual Asian Computing Science Conference. Springer, Berlin, Heidelberg, 2007.
9. Chan, A. P. F., et al. "Comparison of different fusion approaches for network intrusion detection using ensemble of RBFNN." Proceedings of 2005 international conference on machine learning and cybernetics. Vol. 6. 2005
10. Osanaiye, Opeyemi, et al. "Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing." EURASIP Journal on Wireless Communications and Networking 2016.1 (2016): 130.
11. Zhang, Fengli, and Dan Wang. "An effective feature selection approach for network intrusion detection." 2013 IEEE Eighth International Conference on Networking, Architecture and Storage. IEEE, 2013.
12. Jia, Bin, et al. "A DDoS attack detection method based on hybrid heterogeneous multiclassifier ensemble learning." Journal of Electrical and Computer Engineering 2017 (2017)
13. Aburomman, Abdulla Amin, and Mamun Bin Ibne Reaz. "A survey of intrusion detection systems based on ensemble and hybrid classifiers." Computers & Security 65 (2017): 135-152.