An ML Approaches on

network Intrusion Detection

**Supervised By:**

Assistant Prof. Ahmed Al Doulat

Department of Computing

East Tennessee State University

**Submitted By:**

Manoj Adhikari | E00752068

East Tennessee State University

Table of Contents

[1. Project Introduction 2](#_Toc165363669)

[2. Problem Statement 2](#_Toc165363670)

[3. Motivation 4](#_Toc165363671)

[4. Questions that the project will answer: 4](#_Toc165363672)

[5. A Brief Literature Review 5](#_Toc165363673)

[6. Dataset Detailed Description: 5](#_Toc165363674)

[1. Dataset Features: 6](#_Toc165363675)

[2. Numeric Features: 6](#_Toc165363676)

[3. Categorical Features: 8](#_Toc165363677)

[4. Target Variable: 9](#_Toc165363678)

[5. Class Type: 9](#_Toc165363679)

[6. Class Balance: 9](#_Toc165363680)

[7. Project Methodology: Detailed Approach to Solving Problem 10](#_Toc165363681)

[7.1. Data Collection: 10](#_Toc165363682)

[7.2. Data Cleaning 10](#_Toc165363683)

[7.3. Feature Selection 12](#_Toc165363684)

[7.4. Feature Extraction 14](#_Toc165363685)

[7.5. Model Building 14](#_Toc165363686)

[7.6. Results and Insights 16](#_Toc165363687)

[8. Answers to the research questions: 18](#_Toc165363688)

[9. Conclusion 19](#_Toc165363689)

[10. Future research directions 19](#_Toc165363690)

[11. Team member’s roles (if applicable): 20](#_Toc165363691)

[12. Things you learned from the project 20](#_Toc165363692)

[References: 22](#_Toc165363693)

# **Project Introduction**

In today's digital age, cybersecurity threats are pervasive and evolving. As organizations increasingly rely on networked systems, the risk of intrusion and data breaches grows exponentially. Identifying and mitigating these intrusions is paramount to ensuring the security and integrity of sensitive information[1], [2].

Intrusion detection serves as an early warning system by alerting security teams to suspicious activities or anomalies in network traffic. By monitoring network behavior in real-time, organizations can proactively identify and respond to potential security threats before they escalate into full-blown attacks[3], [4]. This project will leverage diverse machine-learning techniques to develop a robust intrusion detection system that accurately distinguishes between normal and anomalous network traffic. The project falls under the domain of cybersecurity and machine learning, specifically focusing on network intrusion detection using the dataset simulated in the military network environment[5].

The project encompasses the supervised machine learning approach to detect network intrusion. The use case of supervised learning within the project is to be able to classify and predict if the network data is either normal or intruded/anomalous based on the provided network data features.

# **Problem Statement**

Network intrusion, leading to data breaches, stands as one of the most critical challenges faced by our interconnected world. With the exponential growth of the internet and networking technologies, the threat landscape has evolved significantly. Cyber attacks, including malware, ransomware, phishing, and various forms of intrusion attempts, have become more sophisticated and frequent.

In today's digital age, organizations store vast amounts of sensitive data, including personal information, financial records, and intellectual property, on their networks. A breach in network security can lead to severe consequences, including financial losses, reputational damage, and legal liabilities. The Alarming Statistics: In the year 2023, data breaches surged globally[6]. The heat map of data breaches revealed a concerning pattern, with incidents spanning continents and industries. No organization or individual remained immune. Network Intrusion Detection System serves as a crucial defense mechanism against these threats and helps to safeguard this sensitive data by detecting potential threats beforehand. Below is the screenshot of the data breach alone in the year 2023.

A map of the world

Description automatically generated

Fig1: Data breach –World heatmap[6]

A graph of numbers and a number of people

Description automatically generated with medium confidence

Fig: Most Breached Countries[6]

# **Motivation**

The motivation to develop the project “*Network Intrusion Detection System*” comes as an action to address the problem of network intrusion, this project intends to develop robust machine learning models as a preventive approach to detect network intrusion and will serve as a preventive shield, detecting anomalies and potential threats in real-time. By doing so, we intend to save organizations and individuals from the dire consequences of security lapses

With the increasing sophistication of cyber threats, it is imperative to accurately identify and mitigate potential intrusions to safeguard critical assets and infrastructure[7]. Traditional rule-based IDS systems often struggle to keep pace with evolving threats and may generate false positives, leading to unnecessary disruptions and resource wastage. By leveraging machine learning techniques, the project aims to develop a more adaptive and effective IDS capable of distinguishing between normal network traffic and malicious attacks in real time[2], [8]. By analyzing a comprehensive dataset containing simulated intrusions in a military network environment, the project aims to develop diverse models that can proactively identify and respond to potential threats, thereby safeguarding critical assets and infrastructure.

With the development of a more advanced and adaptive IDS, the project aims to bolster the resilience of military and common networks against emerging cyber threats, ensuring the integrity and confidentiality of sensitive data. The project also intends to explore the data clusters, as well as experience the implementation of multi-layer perceptron neural networks, and other diverse traditional ML algorithms. This project would provide a learning curve and hands-on experience, enabling us to better understand the intricacies of intrusion detection and machine learning techniques in cybersecurity. Moreover, the motivation of the project is not only to build a robust network intrusion detection system but, also to perform diverse experiments with the model performance in several scenarios such as various data imputation and their relative effects on the model, conducting feature engineering for a large number of features, etc.

# **Questions that the project will answer:**

The inquiries that the project aims to address are:

1. Can machine learning algorithms effectively differentiate between normal network traffic and anomalous behavior based on the provided dataset?
2. Which features are most influential in accurately detecting intrusions within the network?
3. Which machine learning models( traditional classifiers) or multi-layer perceptron neural networks perform better for the given dataset of the intrusion detection system?
4. Can the model generalize well to unseen data and adapt to evolving cybersecurity threats?

# **A Brief Literature Review**

Network intrusion detection systems (NIDS) have been a focal point of research and development in the field of cybersecurity. Researchers and practitioners have made significant strides in enhancing the accuracy, efficiency, and robustness of these NIDS systems.

Several approaches to the development of network intrusion detection can be found in [1], [2], [7], [8], [9][10]. These works tend to find diverse approaches to intrusion detection related to traditional models, state of art classifiers, and the neural network.

The research work at [2] describes the detection of several types of attacks such as Probe, DoS, R2L, and U2R which can be put under the anomalous network. They implement classifiers such as Support Vector Machine, Random Forest, Naïve Bayes, Decision Tree, and AdaBoost, as well as used the Autoencoder method to compare with the rest of the supervised learning methods.

Ibrahim et al., at[8] , explained implementation of the classifiers such as KNN, Naïve Bayes, Random Forest, and Logistic Regression for the binary classification of the network behavior.

In [1], the researcher performs the adjustment of the weight and biases of the neural networks using the ant colony algorithms and analyzes network intrusion detection.

The work at [3] explains the strength of combining the distinct strengths of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), specifically, Bi-directional LSTM to offer high accuracy and low false positive rate. Moreover, in [4], the researcher implemented several neural network models such as CNN, RNN, LSTM, and GRU to detect the intrusion and perform the network behaviors. Additionally, Sowmya et al. [11], provide a comprehensive review of multiple papers related to network intrusion detection methodologies, highlighting the superior performance of neural networks.

This proposed project aids in connecting the dataset to the diverse approaches of intrusion detection that the papers have discussed and making evaluations to find the best model when additional work such as feature selection and different sets of parameter tuning is performed. Through this endeavor, valuable insights can be gained, contributing to the advancement of network intrusion detection systems and bolstering network security measures.

# **Dataset Detailed Description:**

The dataset used for this project is taken from kaggle[12] which has been curated from the UCI KDD [5] cutting off the instances to 25k to reduce the computational complexity of the resources during the project. The dataset comprises TCP/IP dump data collected from a simulated US Air Force LAN environment. It includes a diverse range of intrusion data instances, meticulously designed to mimic real-world attack scenarios. Each connection in the dataset is labeled as either normal or anomalous. The dataset is pretty much integrated and clean.

## **Dataset Features:**

The dataset includes 38 numerical features and 3 categorical features. The dataset offers a rich source of information for training and evaluating intrusion detection models. These 41 features encompass various aspects of network traffic, including duration, protocol type, service, flags, and numerous other parameters indicative of potentially malicious activity.

## **Numeric Features:**

Total count of numerical features: 38

Table 1: Numerical Features of IDS Dataset

|  |  |  |
| --- | --- | --- |
| Feature | Description | Data Type |
| duration | Length (number of seconds) of the connection | Continious |
| src\_bytes | Number of data bytes from source to destination | Continious |
| dst\_bytes | Number of data bytes from destination to source | Continious |
| land | 1 if connection is from/to the same host/port; 0 otherwise | Discrete |
| wrong\_fragment | Number of "wrong" fragments | Continious |
| urgent | Number of urgent packets | Continious |
| hot | Number of "hot" indicators | Continious |
| num\_failed\_logins | Number of failed login attempts | Continious |
| logged\_in | 1 if successfully logged in; 0 otherwise | Discrete |
| num\_compromised | Number of "compromised" conditions | Continious |
| root\_shell | 1 if root shell is obtained; 0 otherwise | Discrete |
| su\_attempted | 1 if "su root" command attempted; 0 otherwise | Discrete |
| num\_root | Number of "root" accesses | Continious |
| num\_file\_creations | Number of file creation operations | Continious |
| num\_shells | Number of shell prompts | Continious |
| num\_access\_files | Number of operations on access control files | Continious |
| num\_outbound\_cmds | Number of outbound commands in an FTP session | Continious |
| is\_host\_login | 1 if the login belongs to the "hot" list; 0 otherwise | Discrete |
| is\_guest\_login | 1 if the login is a "guest" login; 0 otherwise | Discrete |
| count | Number of connections to the same host as the current connection in the past two seconds | Continious |
| srv\_count | Number of connections to the same service as the current connection in the past two seconds | Continious |
| serror\_rate | % of connections that have "SYN" errors | Continious |
| srv\_serror\_rate | % of connections that have "SYN" errors | Continious |
| rerror\_rate | % of connections that have "REJ" errors | Continious |
| srv\_rerror\_rate | % of connections that have "REJ" errors | Continious |
| same\_srv\_rate | % of connections to the same service | Continious |
| diff\_srv\_rate | % of connections to different services | Continious |
| srv\_diff\_host\_rate | |  | | --- | | % of connections to different hosts | |  | | Continious |
| dst\_host\_count | the count of destination hosts | Continious |
| dst\_host\_srv\_count | the count of destination hosts using a specific service. | Continious |
| dst\_host\_same\_srv\_rate | the count of destination hosts using a specific service. | Continious |
| dst\_host\_diff\_srv\_rate | the rate of connections to different services from a destination host | Continious |
| dst\_host\_same\_src\_port\_rate | the rate of connections from the same source port to the destination host. | Continious |
| dst\_host\_srv\_diff\_host\_rate | the rate of connections to different hosts using the same service. | Continious |
| dst\_host\_serror\_rate | the rate of errors for connections to a destination host | Continious |
| dst\_host\_srv\_serror\_rate | the rate of errors for connections to a destination host | Continious |
| dst\_host\_rerror\_rate | the rate of errors for connections to a destination host | Continious |
| dst\_host\_srv\_rerror\_rate | the rate of errors for connections to a destination host using a specific service. | Continious |

## **Categorical Features:**

Total count of categorical features: 3

Table 2: Categorical Features of IDS Dataset

|  |  |  |
| --- | --- | --- |
| Feature | Description | Data-Type |
| Protocol\_type | type of the protocol, e.g. tcp, udp, etc | object |
| service | network service on the destination, e.g., http, telnet, etc. | object |
| flag | normal or error status of the connection | object |

## **Target Variable:**

The project aims to detect whether the network is anomalous or not, with the target variable or the dependent variable in the dataset being ‘**Class’**.

## **Class Type:**

1. Normal: Referes to normal network
2. Anamoly: Refers to the anomalous network

## **Class Balance:**

* 1. There is a slight difference in the number of class counts between the normal anomaly of around 7% which can be considered as the balanced class.

Total count of normal network class: 13,442

Total count of the anomalous network class: 11,737

A green and red squares

Description automatically generated

Screenshot: Count of data points by class

# **7. Project Methodology: Detailed Approach to Solving Problem**

## **7.1. Data Collection:**

The dataset of the network intrusion was obtained from the Kaggle [12] which is the subset of the KDD main dataset[5], [13]. The dataset consists of a wide variety of connection data points that include intrusions simulated in a military network environment. A connection is a sequence of TCP packets starting and ending at some time duration between which data flows to and from a source IP address to a target IP address under some well-defined protocol. The data was available in the form of a CSV, which was downloaded into the project environment for further processing.

## **7.2. Data Cleaning**

Several checkpoints were created during the data cleaning process. It is necessary to be sensitive while cleaning the anomaly detection system development-related dataset, especially during the outlier analysis. The checkpoints in data cleaning include:

#### Data Redundancy check

Row Duplicacy check: The data duplication check was performed on the rows to check if there were any redundant rows. However, no exact duplicate rows were found in the loaded dataset.

Column Sharing same information: If two columns are highly co-related to each other, and if have the same correlation pattern as the rest of the other columns, then those columns can be considered as the same info sharing or likely to be duplicate columns, as they share the same correlational values across all the features.

Such a pair of columns was discovered, and only one of the columns that can be better representative of the target feature was selected.

Examples of such pairs of features include:

• serror\_rate and srv\_serror\_rate

• rerror\_rate and srv\_rerror\_rate

#### Exploring unique values among each column:

Unique values present in each of the features were extracted. There were two features with only one/single value all across the dataset. This means these two columns were not contributing to distinguishing the target classes. Hence, these two features, namely *'num\_outbound\_cmds'*, and *'is\_host\_login'* were dropped from the dataset.

Since, both the columns were having the same value all across the dataset, this was well visualized in the correlational matrix as well.

A screenshot of a chart

Description automatically generated

Fig: Columns with a single value creating the same correlational value pattern across the matrix.

#### Dealing with Missing value

Handling missing values was addressed by scrutinizing the dataset for the existence of NaN, Null, or None values. Upon examination, it was found that the dataset did not contain any missing values. Since, no NA values were detected, dropping of NA values or imputation was not required.

#### Finding inconsistent feature values

Noise check was performed by looking at the unique values across the columns, if the data is present in some other format, or if the data type of the numerical values is stringified (in the form of a string object).

No such obeservations were found in the process. The absence of these noisy data signifies that the dataset's feature values maintain consistency and adhere to the expected formats, ensuring the integrity and reliability of the data for subsequent analyses.

#### Dealing with outliers

The outliers analysis of the numerical features was performed using the z-score method. The threshold value for the z-score was set to 3, such that the z-score of the values exceding the z-score were classified as the outlier across the normal distribution of the feature values. 2521 outlier values were detected. However, removing outliers is not always the solution, as those outliers could be exceptional behavior related to an anomaly class. Hence, they are not removed but kept to handle these exceptional data patterns.

However, the categorical feature values that appeared less than 5 times in the whole dataset were also removed, as they can reduce the model performance since the probability of getting those values included in one of the training or the testing set would be low, as could be an outlier too.

A close-up of a computer code

Description automatically generated

Fig: Low count of *protocol\_type* across the dataset

#### Data Encoding

The Label encoding has been performed to the categorical features - *'protocol\_type'*, *'service'*, and *'flag'*, as well as to the target variable ‘*class*’. One hot encoding could have been the option, but the dimension would be high after performing one hot encoding especially when there is a high number of unique values associated with each categorical feature.

## **7.3. Feature Selection**

For feature selection, the project has implemented a Pearson correlational study to observe the feature’s correlation strength between each other, and the target variable. The features with a correlation less or equal to that absolute Pearson correlation coefficient value of 0.1 were marked as insignificant features, while the rest as significant features. These correlational test for feature selection was also compared with the sklearn feature selection function called *mutual\_info\_classif*. This feature selection module involves selecting a subset of the original features based on their relevance to the target variable.

The *mutual\_info\_classif* function calculates the mutual information between each feature and the target variable. Mutual Information(MI) provides a measure of the amount of information obtained about one variable through the other variable. The top 10 significant features that contribute to the classification of normal and anomalous networks detected from the implementation of mutual information and correlational study are:

* + 1. src\_bytes
    2. dst\_bytes
    3. diff\_srv\_rate
    4. same\_srv\_rate
    5. dst\_host\_srv\_count
    6. dst\_host\_same\_srv\_rate
    7. dst\_host\_srv\_diff\_host\_rate
    8. logged\_in
    9. dst\_host\_srv\_serror\_rate
    10. serror\_rate

A graph of blue and black bars

Description automatically generated

Figure: Significant Features based on the Mutual Information Analysis method.

List of less significant features:

* duration
* land
* wrong\_fragment
* urgent
* hot
* num\_failed\_logins
* num\_compromised
* root\_shell
* su\_attempted
* num\_file\_creations
* num\_shells
* num\_access\_files
* is\_guest\_login
* srv\_count
* rerror\_rate
* srv\_diff\_host\_rate
* dst\_host\_srv\_rerror\_rate

MI method was implemented to observe the important features associated with the dataset, however, all the features were taken into account apart unless they have mutual information obtained between features value near 0 (significantly low), for example: mutual information value 0.000799 and correlation coefficient value 0.013 near to 0 (significantly low) for feature *num\_shells*, which was removed.

## **7.4. Feature Extraction**

For feature extraction, Principal Component Analysis (PCA) was performed. 80 % of the variability of the data was captured using the 8 principal components as shown in the below figure. The dataset was standardized using the StandScaler pre-processing method to standardize and scale the data points. Using significantly helped to reduce dimensions and make project computational resources effective.

A colorful squares with black text

Description automatically generated with medium confidence

Figure: Amount of variability captured in each of the principle components from each feature.

## **7.5. Model Building**

The process implemented during the model building is detailed below.

#### Train-Test Data Split:

The dimension-reduced dataset after performing the PCA was now split into the train and test set with a test\_size of 0.3. Stratification was performed during the dataset splitting using parameter *stratify=’y’*. This means that the split will be done in a way that ensures the proportion of classes of the target variable (y) is the same in both the training and testing sets, as well as the data shuffle was also set as True. This is particularly useful for imbalanced datasets where one class may be significantly underrepresented. Moreover, *random\_state* was used to ensure that the split is reproducible, meaning that when the code is run again with the same random state, the same split will be reproduced

Four traditional classifiers and a multi-layer perceptron neural network model were used in the projects. The five models used in the projects are listed below:

1. Logistic Regression Classifier
2. Random Forest Classifier
3. Support Vector Machine Classifier
4. Gaussian Naïve Bayes Classifier, and
5. Multi-layer perception
6. Logistic Regression Classifier

Logistic Regression was built using the LogisticRegression from sklearn and trained with the 5 cross-validation folds, taking accuracy as the measure. The *solver* used in the logistic regression was *'liblinear',* which was chosen as it is s suitable for small to medium-sized datasets and is based on a coordinate descent algorithm. It works well for binary classification problems.

1. Random Forest Classifier

Random Forest Classifier was built using RandomForestClassifier from sklearn ensemble model and trained using the *‘gini’* as the criterion parameter, *‘sqrt’* as the *max\_features* parameter and *max\_leaf\_nodes* was set to 10 and was trained with 5- fold CV.

The *max\_features* parameter determines the maximum number of features considered for splitting a node*. 'sqrt'* refers to the square root of the total number of features. This setting is commonly used to prevent overfitting by restricting the number of features considered at each split.

1. Support Vector Machine Classifier

Support Vector Machine was built with SVC from sklearn with two different kernels, *linear* and *sigmoid*. When trained over 5-fold CV, the *linear* kernel (mean accuracy-94%) performed better in comparison to the *sigmoid*(mean accuracy-83%), hence *linear* kernel was used to train the model moving ahead.

1. Gaussian Naïve Bayes Classifier

Gaussian Naïve Bayes Classifier was built using GaussianNB from sklearn and was trained over the 5-fold CV to observe the accuracy of the model, and was tested with the test data. The GaussianNB classifier is initialized with a *var\_smoothing* parameter, which is a smoothing value for numerical stability. Adjusting this parameter can help improve the performance of the classifier, especially when dealing with small variances in the data.

1. Multi-layer Perceptron Neural Net Model

The multi-layer perceptron neural network classifier model was built using the MLPClassifier. Hyper-parameter tunning was performed during the model training using a variation of activation functions such as *relu* and *tanh*, solver such as *adam* and *sgd*, along with different values of alpha and hidden layer sizes. These hyper-parameters were used to tune the model using the RandomizedSearchCV over CV of 10 folds with early\_stopping criteria set True. The model was found to perform best for the *relu* solver, for *alpha* value of 0.001.

## **7.6. Results and Insights**

Model Performance Table based on the classification report

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision (Anomaly)** | **Recall (Anomaly)** | **F1-score (Anomaly)** | **Precision (Normal)** | **Recall (Normal)** | **F1-score (Normal)** |
| Logistic Reg | 0.95 | 0.95 | 0.93 | 0.94 | 0.94 | 0.96 | 0.95 |
| Random Forest | 0.96 | 0.97 | 0.94 | 0.96 | 0.95 | 0.97 | 0.96 |
| SVM | 0.95 | 0.96 | 0.93 | 0.94 | 0.94 | 0.96 | 0.95 |
| Naive-Bayes | 0.92 | 0.93 | 0.90 | 0.92 | 0.92 | 0.94 | 0.93 |
| Multi-Layer Percp | 0.98 | 0.99 | 0.97 | 0.98 | 0.98 | 0.99 | 0.98 |

A screenshot of a graph

Description automatically generated

Figure: Confusion Matrix for each models

A graph of different colored bars

Description automatically generated

Figure: Precision and Recall for Anomaly Class

**Insights:**

The multi-layer perceptron (MLP) model emerges as a top performer in anomaly detection and classification of normal instances. With precision scores of 0.99 for anomalies and 0.98 for normal instances, the model demonstrates exceptional accuracy in its predictions. Moreover, with recall values of 0.97 for anomalies and 0.99 for normal instances, the MLP model exhibits outstanding capability in capturing instances of interest while maintaining a low false negative rate. With an accuracy of 98%, the MLP model stands out as a superior classifier, suitable for applications requiring highly accurate anomaly detection.

Among the models presented, the Random Forest model emerges as the second-best option after MLP. The Random Forest model demonstrates robust performance with high precision scores of 0.97 for anomalies and 0.95 for normal instances, indicating strong correctness in its predictions. Additionally, with recall values of 0.94 for anomalies and 0.97 for normal instances, the model exhibits a solid capability in capturing instances of interest while maintaining a low false negative rate. With an accuracy of 96%, the Random Forest model proves to be a powerful classifier, suitable for applications requiring accurate anomaly detection.

The Gaussian Naïve Bayes Classifier has lower performance than other models. While the Naive-Bayes model achieves respectable precision scores of 0.93 for anomalies and 0.92 for normal instances, indicating a reasonable level of correctness in its predictions, it falls slightly short in terms of recall. With recall values of 0.90 for anomalies and 0.94 for normal instances, the model exhibits a lower ability to capture instances of interest compared to other models. This indicates that the Naive-Bayes model may struggle more with correctly identifying anomalies, potentially leading to a higher false negative rate.

If features in the dataset are highly correlated, Naive-Bayes might struggle to capture these relationships effectively. Other models like Random Forest and Multi-Layer Perceptron are more capable of capturing complex feature interactions.

Naive-Bayes performs well when the data is simple and there is a clear separation between classes. However, if the data is highly complex with overlapping distributions, Naive-Bayes may not be able to capture this complexity as effectively as other models.

Overall, the Multi-Layer Perceptron neural network demonstrates the best performance for the given dataset of the intrusion detection system, achieving higher precision, recall, F1-score, and overall accuracy when compared to other models.

# **8. Answers to the research questions:**

**RQ1:** Can machine learning algorithms effectively differentiate between normal network traffic and anomalous traffic based on the provided dataset?

* The project suggests that machine learning algorithms exhibit the capability to effectively distinguish between normal network traffic and anomalous traffic using the provided dataset subset of KDD dataset[5]. However, the efficiency of classifying anomalous network activity may vary depending on factors such as the specific models employed and their corresponding training parameters.

**RQ2:** Which features are most influential in accurately detecting intrusions within the network?

* The most influential features in accurately detecting intrusion within the network, based on the mutual information feature selection method, are:

i. src\_bytes

ii. dst\_bytes

iii. diff\_srv\_rate

iv. same\_srv\_rate

v. dst\_host\_srv\_count

vi. dst\_host\_same\_srv\_rate

vii. dst\_host\_srv\_diff\_host\_rate

viii. logged\_in

**RQ3:** Which machine learning models( traditional classifiers) or multi-layer perceptron neural networks perform better for the given dataset of the intrusion detection system?

* The Multi-Layer Perceptron neural network demonstrates the best performance for the given dataset of the intrusion detection system, achieving higher precision, recall, F1-score, and overall accuracy compared to traditional classifiers such as logistic regression, random forest, SVM, and Naive-Bayes. The MLP model's ability to capture complex relationships within the data and its adaptability to various patterns contribute to its superior performance in this scenario.

**RQ4:** Can the model generalize well to unseen data and adapt to evolving cybersecurity threats?

* The model's ability to generalize well to new unseen real-case datasets and adapt to evolving cybersecurity threats depends on several factors. Given that the dataset used in this project is produced from a simulation environment under control and represents only a small portion of the larger knowledge base of the original KDD dataset, there are limitations to its generalizability. The simulated environment may not fully capture the complexity and diversity of real-world network traffic and cybersecurity threats. Therefore, the model's performance on unseen real-case data may be compromised.

However, if the unseen data follows similar features as those present in this project dataset, the model may demonstrate robustness in classifying anomalous network activity from normal activity. The model's ability to identify patterns and relationships among the features can contribute to its effectiveness in detecting intrusions, even in evolving cybersecurity landscapes. Additionally, ongoing updates and retraining of the model with new data and emerging threat patterns can enhance its adaptability over time.

# **9. Conclusion**

In conclusion, the study on Network Intrusion Detection has revealed valuable insights into the effectiveness of various machine learning models for distinguishing between normal network traffic and anomalous activity. The Multi-Layer Perceptron (MLP) model emerged as the top performer, demonstrating exceptional precision, recall, and overall accuracy in its predictions. Its ability to capture complex feature interactions makes it particularly suitable for applications requiring highly accurate anomaly detection.

Moreover, the research addressed key questions regarding the efficacy of machine learning algorithms in intrusion detection and influential features in detecting intrusions. The model's ability to generalize well to new unseen real case data and adapt to evolving cybersecurity threats may not be robust due to the model trained on the simulated dataset. However, this can be mitigated through ongoing updates and retraining with new data which can enhance its adaptability over time. In summary, the study underscores the significance of employing advanced machine learning techniques like MLP for effective network intrusion detection, while also highlighting the importance of continuous refinement and adaptation to address evolving cybersecurity challenges.

# **10. Future research directions**

Future research directions for the Network Intrusion Detection project could involve several avenues to further enhance its capabilities and applicability. Firstly, the project can explore the implementation of additional state-of-the-art models like XGBoost, LightGBM, and AdaBoost. By comparing the performance of these models, including their computational efficiency, the project can gain insights into which algorithms are best suited for detecting network intrusions.

Furthermore, the project can investigate alternative feature selection techniques, such as Recursive Feature Elimination (RFE), in addition to the Mutual Information method used in the current implementation. By building models on top of RFE-selected features and comparing the results with those obtained from Mutual Information, the project can evaluate the efficacy of different feature selection approaches and identify the most suitable method for improving model performance.

Additionally, expanding the scope of the project to address multi-class classification problems could be beneficial. By effectively classifying different types of network anomalies present in real-world breached network datasets, the project can provide more comprehensive insights into cybersecurity threats and aid in the development of robust intrusion detection systems.

Overall, by pursuing these future research directions, the Network Intrusion Detection project can advance its capabilities, contribute to the development of more effective intrusion detection systems, and better address the evolving challenges in cybersecurity.

# **11. Team member’s roles (if applicable):**

Not applicable. The project is single-handled and developed by Manoj Adhikari[14] under the supervision of the course instructor Dr. Ahmed Al Doulat[15].

# **12. Things you learned from the project**

Through this project, I have gained a comprehensive understanding of the diverse types of features and data points associated with network intrusion datasets. I've learned about the critical importance of data preprocessing and its profound impact on the performance of machine learning models. Specifically, I've developed proficiency in implementing various data cleaning checkpoints such as redundancy checks, duplicate column detection, handling missing or noisy data, data normalization, and handling outliers in anomaly detection projects such as this one where the outliers can also be important data points representing some exceptional patterns from the normal data points. Additionally, I've employed advanced techniques like mutual information to identify the most significant features for classifying the target variables, enhancing the model's predictive power.

Moreover, I've delved into the significance of Principal Component Analysis (PCA) in feature extraction and dimensionality reduction, especially when dealing with datasets containing a large number of features. By leveraging PCA, I've improved computational efficiency while preserving the essential information required for effective model training and prediction. Furthermore, I've gained hands-on experience in model building, training, and hyperparameter tuning, ensuring that the models perform optimally without overfitting to the training data.

One of the most valuable aspects of this project has been the opportunity to analyze model performance and interpret results based on precision, recall, and F1-score. This analytical approach has enabled me to evaluate the models' effectiveness in achieving the project's objectives and provided insights into areas for further refinement and optimization. Overall, this project has equipped me with a robust skill set in data preprocessing, feature selection, model development, and performance evaluation, enhancing my capabilities in tackling real-world challenges in cybersecurity and beyond.

# **References:**

[1] Y. Ren, “Analysis of network intrusion detection based on machine learning,” *2022 4th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, pp. 699–702, Oct. 2022, doi: 10.1109/AIAM57466.2022.00142.

[2] X. Y. Li, R. Tang, and W. Song, “Intrusion Detection System Using Improved Convolution Neural Network,” *2022 11th International Conference of Information and Communication Technology (ICTech)*, pp. 97–100, Feb. 2022, doi: 10.1109/ICTECH55460.2022.00028.

[3] H. Patel, “Anomaly and Threat detection in network traffic using Deep Learning”.

[4] L. Ashiku and C. Dagli, “Network Intrusion Detection System using Deep Learning,” *Procedia Comput Sci*, vol. 185, pp. 239–247, Jan. 2021, doi: 10.1016/J.PROCS.2021.05.025.

[5] “KDD Cup 1999 Data.” Accessed: Feb. 26, 2024. [Online]. Available: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[6] “2023 Was The Worst Year Yet For Data Breaches in Every Way—Except One | PCMag.” Accessed: Feb. 26, 2024. [Online]. Available: https://www.pcmag.com/articles/2023-was-the-worst-year-yet-for-data-breaches

[7] Z. Tauscher, Y. Jiang, K. Zhang, J. Wang, and H. Song, “Learning to Detect: A Data-driven Approach for Network Intrusion Detection,” *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, vol. 2021-October, pp. 1–6, Oct. 2021, doi: 10.1109/IPCCC51483.2021.9679415.

[8] I. Abobaker and A. Musa, “Machine Learning for Intrusion Detection and Network Performance,” *2021 8th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 86–91, Aug. 2021, doi: 10.1109/FICLOUD49777.2021.00020.

[9] A. Guezzaz, S. Benkirane, M. Azrour, and S. Khurram, “A Reliable Network Intrusion Detection Approach Using Decision Tree with Enhanced Data Quality,” *Security and Communication Networks*, vol. 2021, 2021, doi: 10.1155/2021/1230593.

[10] B. S. Babu, G. A. Reddy, D. K. Goud, K. Naveen, and K. S. T. Reddy, “Network Intrusion Detection using Machine Learning Algorithms,” *2023 3rd International Conference on Smart Data Intelligence (ICSMDI)*, pp. 367–371, Mar. 2023, doi: 10.1109/ICSMDI57622.2023.00071.

[11] T. Sowmya and E. A. Mary Anita, “A comprehensive review of AI based intrusion detection system,” *Measurement: Sensors*, vol. 28, p. 100827, Aug. 2023, doi: 10.1016/J.MEASEN.2023.100827.

[12] “Network Intrusion Detection.” Accessed: Apr. 28, 2024. [Online]. Available: https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection

[13] “KDD-CUP-99 Task Description.” Accessed: Feb. 26, 2024. [Online]. Available: https://kdd.ics.uci.edu/databases/kddcup99/task.html

[14] “Manoj Adhikari.” Accessed: Feb. 26, 2024. [Online]. Available: https://manojadhikary.com.np/

[15] “Faculty and Staff.” Accessed: Feb. 26, 2024. [Online]. Available: https://www.etsu.edu/cbat/computing/faculty\_and\_staff/