

Multi-Threat Detection: A Well-Rounded Approach for

Common Attacks Found in Cyber Space

M.T.D.



## Anthony Martinez. Jalen Williams, John Cavazos, Leo Martinez III

## Department of Electrical Engineering and Computer Science

## Texas A&M University

## Kingsville, Texas, USA

## May 1, 2024

Email: {anthony.martinez02, jalen.williams, john.cavazos, leo.martinez}@students.tamuk.edu

Table of Contents:

[Title Page i](#_Toc165459793)

[Table of Contents ii](#_Toc165459793)

[List of Figures i](#_Toc165459794)v

[List of Tables i](#_Toc165459794)v

[Abstract: 1](#_Toc165459795)

[Introduction: 1](#_Toc165459796)

[Background: 1](#_Toc165459797)

[Objectives: 2](#_Toc165459798)

[Related Work: 2](#_Toc165459799)

[Intrusion Detection: 3](#_Toc165459800)

[Intrusion Classification: 3](#_Toc165459802)

[Malware Detection: 3](#_Toc165459803)

[Malware Classification: 3](#_Toc165459804)

[Design: 3](#_Toc165459805)

[Datasets: 4](#_Toc165459806)

[Malware Classification Dataset: 4](#_Toc165459807)

[Malware Detection Dataset: 5](#_Toc165459808)

[Network Intrusion Detection & Classification Dataset: 5](#_Toc165459809)

[Feature Extraction & Utilization: 5](#_Toc165459810)

[Malware Classification Feature Extraction: 5](#_Toc165459811)

[Malware Detection Feature Utilization: 6](#_Toc165459812)

[Network Intrusion Feature Utilization: 6](#_Toc165459813)

[Feature Preprocessing & Selection: 6](#_Toc165459814)

[Categorical Data Transformation: 6](#_Toc165459815)

[Numerical Data Transformation: 6](#_Toc165459816)

[Feature Learning/Selection: 7](#_Toc165459817)

[Methodology: 7](#_Toc165459818)

[Network Intrusion Detection: 8](#_Toc165459819)

[Network Intrusion Classification: 9](#_Toc165459820)

[Malware Detection: 11](#_Toc165459821)

[Malware Classification: 12](#_Toc165459822)

[Evaluation: 13](#_Toc165459823)

[Accuracy 13](#_Toc165459824)

[Precision 13](#_Toc165459825)

[Recall 13](#_Toc165459826)

[F1-Score 13](#_Toc165459827)

[Area Under the Curve (AUC) 13](#_Toc165459828)

[Confusion Matrix 13](#_Toc165459829)

[Cross-validation(k-folds) 13](#_Toc165459830)

[The Importance of Evaluation: 13](#_Toc165459831)

[Results Analysis: 14](#_Toc165459832)

[Network Intrusion Detection Comparison with Contemporary Studies: 15](#_Toc165459833)

[Network Intrusion Classification Comparison with Contemporary Studies: 16](#_Toc165459834)

[Malware Detection Comparison with Contemporary Studies: 16](#_Toc165459835)

[Malware Classification Comparison with Contemporary Studies: 17](#_Toc165459836)

[Website Integration: 17](#_Toc165459837)

[Android App Integration: 19](#_Toc165459838)

[Conclusion: 20](#_Toc165459839)

[Acknowledgement: 20](#_Toc165459840)

[Appendix: 21](#_Toc165459841)

[References: 27](#_Toc165459842)

# List of Figures:

[Figure 1 High-Level view of MTD 4](#_Toc165461743)

[Figure 2 General methodology of ML models 7](#_Toc165461744)

[Figure 3 Network intrusion detection flowchart 9](#_Toc165461745)

[Figure 4 DNN training plot of loss and accuracy 10](#_Toc165461746)

[Figure 5 DNN architecture 10](#_Toc165461747)

[Figure 6 Network intrusion classification flowchart 11](#_Toc165461748)

[Figure 7 Malware detection flowchart 12](#_Toc165461749)

[Figure 8 Malware classification flowchart 13](#_Toc165461750)

[Figure 9 MTD website homepage 18](#_Toc165461751)

[Figure 10 MTD website about page 18](#_Toc165461752)

[Figure 11 MTD android app performance page 19](#_Toc165461753)

[Figure 12 MTD android app homepage 19](#_Toc165461754)

[Figure 13 MTD android app recent scans page 21](#_Toc165461755)

[Figure 14 MTD homepage (different view) 21](#_Toc165461756)

[Figure 15 Stacking model flowchart 21](#_Toc165461757)

[Figure 16 Intrusion detection heatmap 22](#_Toc165461758)

[Figure 17 Intrusion classification heatmap 22](#_Toc165461759)

[Figure 18 Malware classification heatmap 22](#_Toc165461760)

[Figure 19 Malware detection heatmap 22](#_Toc165461761)

[Figure 20 Intrusion classification pseudocode 23](#_Toc165461762)

[Figure 21 Intrusion detection pseudocode 24](#_Toc165461763)

[Figure 22 Malware classification pseudocode 25](#_Toc165461764)

[Figure 23 Malware detection pseudocode 26](#_Toc165461765)

# List of Tables:

[Table 1 Datasets utilized 4](#_Toc165461766)

[Table 2 Model methodologies 8](#_Toc165461767)

[Table 3 Performance metrics of all four models 15](#_Toc165461768)

[Table 4 Network intrusion detection comparisons 16](#_Toc165461769)

[Table 5 Network intrusion classification comparisons 16](#_Toc165461770)

[Table 6 Malware detection comparisons 17](#_Toc165461771)

[Table 7 Malware classification comparisons 17](#_Toc165461772)

# Abstract:

The rapid evolution of cyber threats, from early malware attacks to sophisticated network intrusions poses significant risks to many industries worldwide. This senior design project, titled "Multi-Threat Detection (M.T.D.),” addresses these challenges by employing a well-rounded approach to a multitude of different cyber-attacks. Focusing on intrusion and malware threats, the project utilizes Machine Learning (ML), Deep Learning (DL), and Artificial Intelligence (A.I.) to anticipate and mitigate potential threats. The project's foundation lies in the development of automated malware detection and classification, as well as network intrusion detection and classification techniques. Leveraging specialized Python libraries such as Scikit-learn, TensorFlow, and others, the team aims to create a robust cybersecurity solution capable of effectively addressing a wide range of threats. The project's strategy includes creating an Android app capable of scanning for a multitude of cyber threats and a public website, making the cybersecurity solution accessible to a broader audience. Educational resources, tutorials, installation, and walkthrough guides will further empower users in understanding and utilizing the application effectively. The comprehensive approach, coupled with educational resources, supports the project's goal of improving consistency, accuracy, and reliability in cybersecurity, especially in smaller Internet of Things (IoT) devices such as mobile smartphones.

# Introduction:

In 1986, the emergence of the "Brain" malware marked a pivotal moment in the history of cybersecurity, as it demonstrated the potential for malicious software to infiltrate and disrupt personal computers on a large scale. This early attack, infecting floppy disks in IBM computers, foreshadowed the evolving landscape of cyber threats that would come to pose significant risks to industries worldwide. Since then, the growth of wireless network attacks has further escalated the online landscape. Attackers now employ sophisticated techniques to disguise malicious network traffic as secure, enabling them to infiltrate systems, compromise data integrity, and steal sensitive information. Industries ranging from Fortune 500 tech giants to healthcare and education providers are increasingly vulnerable to these threats, highlighting the urgent need for robust cybersecurity measures.

Recognizing the gravity of these challenges, we dedicated our project to address cybersecurity threats head-on. By leveraging automated malware detection, malware classification, and network intrusion detection techniques, we anticipated and mitigated potential threats well in advance, thereby safeguarding individuals and organizations from harm. Central to our approach is the utilization of Machine Learning (ML) and Artificial Intelligence (AI) technologies, which empower us to forecast and combat cybersecurity threats with efficiency and precision. By utilizing specialized Python libraries tailored for machine learning tasks—such as Scikit-learn, NumPy, Pandas, TensorFlow, Seaborn, Theano, Keras, and others—we possessed a formidable arsenal for developing a cutting-edge cybersecurity solution capable of addressing a diverse array of threats.

Furthermore, to maximize accessibility and usability, we deployed our solution as an Android OS-based application. By leveraging the widespread adoption of mobile devices, we were able to modify access to cybersecurity protection, empowering users across different sectors and demographics to defend against potential cyber threats effectively. In summary, our project created a proactive response to the evolving cybersecurity landscape, combining advanced technologies with user-friendly design to deliver comprehensive protection against a wide range of threats.

## Background:

After our team was formed, we engaged in discussions to determine the primary focus of our project. Following a thorough review of previous projects and extensive deliberation, we made the decision to embark on an Anomaly Detection project, with plans to later expand into Malware Classification and Detection. The inspiration for the malware aspect came from a personal experience involving a compromised laptop due to malware. Traditionally, malware detection primarily targeted known threats, but we leveraged machine learning to develop adaptive detection systems that could respond to evolving malware behaviors. While preparing to test our anomaly detection system, we decided to incorporate the malware component as well. Furthermore, we entertained the idea of creating a website that would provide technology and cybersecurity information, a link to download a mobile app, and an operational tutorial once both the anomaly and malware features were fully completed and functional.

Throughout the project, we conducted extensive research into anomaly detection algorithms and malware classification techniques. We experimented with various machine learning models and data preprocessing methods to enhance the accuracy and effectiveness of our detection systems. Our efforts culminated in the development of a robust anomaly detection system capable of identifying abnormal behavior in network traffic and a sophisticated malware classification system capable of detecting and categorizing malicious software.

# Objectives:

Throughout the entirety of the project development, we employed an incremental approach, efficiently specifying, developing, and validating our solution. Our design alternatives leveraged machine learning algorithms, particularly well-suited for pattern detection, making them more effective than historical methods based on malware comparisons. Our solution relied on these optimal machine learning algorithms for intrusion and malware detection. The key technologies involved included cloud computing via Google Colab, although our software remained compatible with multiple IDEs, all coded in Python. The solution's architecture consisted of major components that harnessed preexisting algorithms and dataset features, working seamlessly with imported libraries to achieve its objectives.

The project strategy centered on an incremental approach, with the primary emphasis placed on the development of machine learning models, forming the bedrock upon which the broader project was constructed. We had already established a foundational network intrusion detection model, which served as a cornerstone for the subsequent development of both the malware detection and malware classification machine learning models.

Following this foundational work, our project extended its scope to include the creation of an Android application in Java and a public website in HTML, CSS, and JavaScript. These platforms enabled users to seamlessly download and install the application on their mobile devices. By providing a versatile yet simple Android app and a user-friendly website, we aimed to make our cybersecurity solution accessible to a wider audience. We developed our user-friendly and easily accessible mobile app user interface. Similarly, we designed a refined and polished version of our website which will be shown later in the report.

Additionally, our project's roadmap encompassed the creation of educational resources, including tutorials and guides, to empower and educate users interested in our application. These resources not only enhanced the usability of our solution but also contributed to a broader understanding of network intrusion and malware detection in the context of cybersecurity.

# Related Work:

In our goal to enhance the credibility of our project, we recognized the importance of studying and understanding the latest advancements within the realm of machine learning research, with a focused examination into network intrusion and malware detection. Delving into a large list of scholarly articles enabled us not only to deepen our understanding of this dynamic field but also furnished us with invaluable benchmark metrics against which we could measure the effectiveness of our own machine learning models. Moreover, we meticulously elucidate the findings from each segment of our research journey, providing comprehensive insights into the predominant methodologies employed across various papers for each machine learning task.

## Intrusion Detection:

## Our exploration uncovered a cluster of seminal works [4], [5], [6], [7] with the goal of intrusion detection. These papers stand as highly respected within the research domain, providing a foundation upon which to cultivate a resilient machine learning framework. Notably, paper [6] stands out for its adept utilization of an autoencoder, a specialized deep neural network architecture leveraged for many tasks such as feature learning, dimensionality reduction, and binary classification.

## Intrusion Classification:

Venturing into intrusion classification led us to a list of noteworthy studies [8], [9], [10], [11]. These contributions serve as cornerstones in our overall knowledge of intrusion types, providing insights for constructing a robust machine learning framework. Noteworthy among these is paper [9], distinguished by its deployment of SVM SMOTE for synthetic data generation, coupled with a deep neural network architecture tailored for 4-class classification.

## Malware Detection: In our exploration of malware detection, we encountered a variety of scholarly research papers [12], [13], [14], [15]. Each of these studies serves as valuable guides, aiding us in the development of a robust machine learning model. Notably, paper [12] stands out for its innovative use of a dilated Convolutional Neural Network (CNN) to identify malicious intent within digital imagery.

## Malware Classification:

Exploring malware classification led us to a collection of influential studies [16], [17], [18], [19]. These seminal works provide the foundation for our machine learning framework. Notably, paper [18] pioneered a behavioral-based approach, utilizing the dynamic behavioral traits of malware families as the key element for classification.

# Design:

In the initial phases of our preproposal, we assigned individual tasks to team members to optimize our time management. As we progressed to the later stages, our strategy shifted towards a collaborative approach. We distributed tasks among the team to facilitate real-time feedback and suggestions, aiming to reduce errors and enhance the overall reliability of the system.

To support the seamless execution of our project, we leveraged third-party applications to assist with scheduling and ensure the punctual delivery of project components. These tools proved invaluable in keeping us on track and maintaining project timelines. Notion played a crucial role, offering excellent organization and collaborative efficiency. It enabled real-time viewing and modification of individual task lists, as well as the easy drag-and-drop functionality for files such as datasets or Python scripts for testing purposes. To gain a larger overall view of our entire project, Figure 1 provided more information.

A diagram of software development

Description automatically generated

Figure 1 High-Level view of MTD

## Datasets:

We employed a variety of datasets from distinct sources that we have carefully selected to align with our specific requirements. These datasets have established recognition within the machine learning research domain, particularly in the field of cybersecurity. Notably, they have been extensively featured in the works of various researchers, including papers authored by [4], [8], [12], and [16]. Our intention was to thoroughly scrutinize these publications to discern opportunities for enhancing their prior research efforts. We have organized the datasets we utilized along with what ML task they correspond to in Table 1 below.

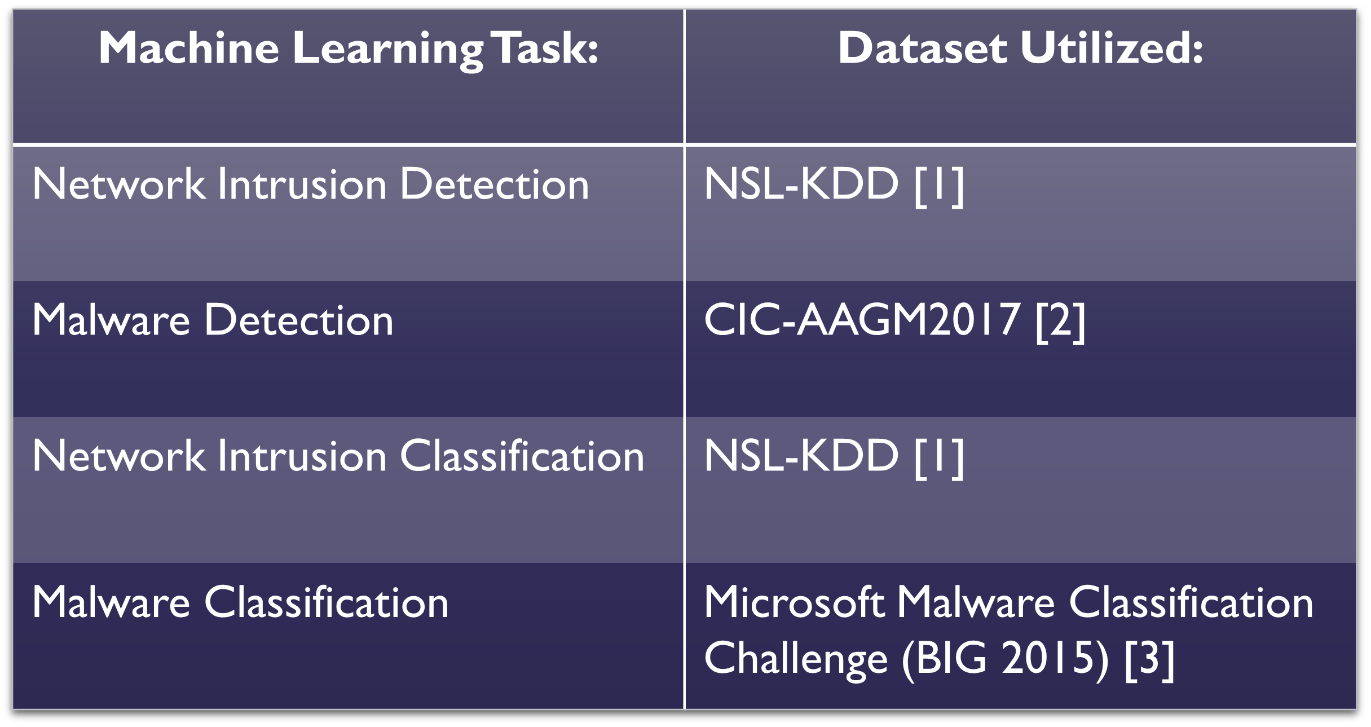


Table 1 Datasets utilized

### Malware Classification Dataset:

In the area of Malware Classification, we leveraged the extensive Microsoft Malware Classification Challenge (BIG 2015) dataset [3]. This dataset encompasses a substantial collection of malware samples in the .asm format, complemented by corresponding .BYTES files, providing an extensive foundation for extracting features. This dual-file format augments our capability to comprehensively analyze and classify malware instances.

The dataset itself is diverse, encompassing samples from nine distinct malware families: Ramnit, Lollipop, Kelihos\_ver3, Vundo, Simda, Tracur, Kelihos\_ver1, Obfuscator.ACY, and Gatak. Each family presents a unique set of characteristics, posing varying degrees of challenge in the classification task. This diversity ensures that our classification model was robust and adaptable to a wide range of potential threats.

### Malware Detection Dataset:

For the purpose of Malware Detection, we make use of the Android Adware and General Malware Dataset, denoted as CIC-AAGM2017 [2]. This dataset encompasses a vast collection of 631,956 diverse samples, encompassing both benign and malicious instances. The dataset's substantial size allows for an effective train/test split without the risk of underfitting our data.

It's noteworthy that the CIC-AAGM dataset was captured by semi-automated installation of Android apps on real smartphones, ensuring real-world relevance [2]. In other words, while certain steps, such as the installation of apps, may require human intervention, other aspects of the data capture process may be carried out automatically by some software or tools. This hybrid approach allows for a balance between human control and the efficiency provided by automation. The dataset originates from 1,900 applications, categorized into three groups: Adware (250 apps), General Malware (150 apps), and Benign (1,500 apps).

An additional advantage of the CIC-AAGM2017 dataset is its specialized concentration on Android OS malware. This specialization harmonizes seamlessly with our project's primary objective, considering that our application is explicitly designed for compatibility with the Android OS platform.

### Network Intrusion Detection & Classification Dataset:

Regarding Network Anomaly Intrusion Detection, we employed the Network Security Laboratory Knowledge Discovery in Databases (NSL-KDD) dataset [1]. A notable distinction in our approach is the utilization of a distinct dataset exclusively for testing purposes, rather than relying on a percentage split from the same dataset. This approach is adopted to avoid any potential biases in our evaluation.

The NSL-KDD dataset offers the flexibility of both binary and multiclass classification. The NSL-KDD dataset holds a distinguished reputation within the research community and has been frequently employed in various iterations of studies pertaining to network anomaly intrusion detection, as exemplified in research such as the application of stacked de-noising autoencoders for intrusion detection in [6].

## Feature Extraction & Utilization:

As we employ a variety of datasets and aim to detect multiple cyber threats, we integrated a dual approach that involves both static/dynamic feature extraction and the conventional utilization of provided features.

### Malware Classification Feature Extraction:

In the domain of Malware Classification, we initially extracted static attributes (meaning characteristics of the malware files themselves) manually from various attributes, including 'mov,' 'call,' 'push,' 'pop,' 'xor,' 'sub,' 'add,' 'jmp,' 'asm\_file\_size,' 'asm\_entropy,' 'num\_bytes,' 'proc\_count,' '.text,' '.data,' '.bss,' '.rodata,' 'comment\_count,' and 'class.' Subsequently, we wproceed to acquire dynamic attributes (meaning how the malicious files behave when released) in a controlled environment to establish a more comprehensive and resilient feature set. Some of these features may include ‘SystemFileManipulation’, ‘CodeInjectionAttempts’, or ‘MultipleProcessCreation’.

The malware instances from which we derived these attributes are drawn from the Microsoft Malware Classification Challenge (BIG 2015) dataset. To facilitate this feature extraction process, we employed a dedicated Python script designed from the ground up. This script has the ability to capture pertinent features from both the '.asm' and '.BYTES' files.

Moreover, to associate the appropriate malware family or 'class' with each malware instance within our dataset, we relied on the 'trainlabels.csv' file provided by Microsoft. The approach involves a single Python script that seamlessly manages both the extraction of features and the assignment of 'class' labels to the corresponding malware samples.

### Malware Detection Feature Utilization:

Turning our focus to Malware Detection, we leveraged the feature set provided by the CIC-AAGM2017 dataset, comprising a total of 80 features. This encompassing array of attributes includes metrics such as 'duration,' 'total\_fpackets,' 'total\_bpackets,' 'total\_fpktl,' 'total\_bpktl,' 'min\_fpktl,' 'min\_bpktl,' 'max\_fpktl,' 'max\_bpktl,' 'mean\_fpktl,' 'mean\_bpktl,' 'std\_fpktl,' 'std\_bpktl,' 'total\_fiat,' 'total\_biat,' 'min\_fiat,' and 'class.'

While this rich feature set is valuable, it's prudent to acknowledge that not every feature may contribute significantly to the classification task. Therefore, we employed established feature selection techniques to discern the most pertinent attributes for our model.

### Network Intrusion Feature Utilization:

In the field of Network Anomaly Intrusion Detection, we turn to the NSL-KDD dataset, which offers a comprehensive set of 43 features, furnishing our machine learning model with all the necessary attributes for effective intrusion detection. Among these attributes are 'duration,' 'protocol\_type,' 'service,' 'flag,' 'src\_bytes,' 'dst\_bytes,' 'land,' 'wrong\_fragment,' 'urgent,' 'hot,' 'num\_failed\_logins,' 'logged\_in,' 'num\_compromised,' 'root\_shell,' and 'su\_attempted,' to name a few.

Just as with the CIC-AAGM2017 dataset, the abundance of features at our disposal necessitates thorough feature selection and data cleaning processes. These steps are imperative to optimize the overall performance of our model, ensuring it's both accurate and efficient in identifying network anomalies and potential intrusions.

## Feature Preprocessing & Selection:

### Categorical Data Transformation:

Categorical data, such as the 'Protocol' attribute, often exist in real-world cybersecurity datasets. This data can represent various network protocols like TCP, UDP, or HTTP. To integrate this data into machine learning algorithms, it's crucial to convert it into a numerical format that algorithms can understand. This transformation prevents the algorithm from misinterpreting the categorical values as ordinal or continuous, which could lead to incorrect results.

#### One-Hot Encoding:

One-Hot Encoding is a valuable technique for categorical data transformation. It converts categorical variables into binary vectors, where each category becomes a binary column. This approach prevents the model from assuming any ordinal relationship between categories. In the context of network intrusion detection, it ensures that different network protocols are treated as distinct entities with equal significance.

#### Label Encoding:

Label encoding is employed to convert class labels into ordered numeric format using lexicographic order. In the context of malware detection, this ensures that the machine learning model can understand and predict the different malware classes accurately.

### Numerical Data Transformation:

Numerical data in cybersecurity often includes attributes such as 'Bytes Transferred,' 'Packet Count,' 'Duration,' or 'Connection Speed.' These attributes are crucial for understanding network behavior and identifying potential security threats. However, these attributes can vary significantly in their scales and units, which can pose challenges for machine learning algorithms.

#### Standardization:

Standardization involves transforming numerical attributes in such a way that they have a mean of 0 and a standard deviation of 1. This process is essential because it makes the data more comparable and easier for machine learning models to process. In cybersecurity, standardization allows various attributes with different units and scales to be directly compared. StandardScalar() from the Scikit Learn Python Library helps to standardize these attributes for overall better performance.

#### Normalization:

Normalization is vital for ensuring that numerical attributes across different scales are on a common scale. In network intrusion detection, features like 'duration' or 'bytes transferred' may vary significantly in magnitude. MinMaxScaler() from the Scikit Learn Python Library helps to normalize these attributes, making them directly comparable and preventing certain features from dominating the model due to their larger scales. After successfully applying normalization to the numeric attribute, the min and max value of a feature value is 0 and 1 respectively.

### Feature Learning/Selection:

The selection of relevant features is critical in machine learning for cybersecurity. In the case of malware and network intrusion detection, the dimensionality of the data can be high due to numerous attributes. Feature selection techniques such as Information Gain (IG), Chi-Square, Recursive Feature Elimination (RFE), and Mutual Information help identify the most informative attributes. Deep Learning (DL), a subset of machine learning, can be applied for feature learning by leveraging tools like autoencoders (a type of artificial neural network) to uncover and understand meaningful patterns amongst different features. These processes can lead to more efficient models that focus on the most relevant aspects of the data, thereby enhancing detection accuracy and reducing computational complexity.

# Methodology:

In our project, we aimed to strike a balance between performance and accuracy, acknowledging the computational demands inherent in machine learning and deep learning tasks. As we engaged in ML tasks, we encountered several anticipated challenges. These challenges included heightened computational costs resulting in prolonged processing times. Additionally, we grappled with issues related to memorization errors, also referred to as overfitting, where the algorithms leaned heavily on memorized patterns rather than accurately interpreting and placing data based on its intrinsic content. Another significant challenge was maintaining accuracy while minimizing latency in the ML domain.

To address these challenges, we conducted an assessment of various ML/DL algorithms, recognizing the variability in accuracy when applied to specific datasets. Through this assessment, we identified and selected the most effective algorithm for our framework. Among the key algorithms we considered were Random Forest, Support Vector Machines (SVM), Neural Networks, K-Nearest Neighbor (KNN), Decision Trees, and Logistic Regression, and Naïve Bayes.

To support our work, we relied on established data science libraries such as Pandas, NumPy, and Scikit-learn [22], [23]. Our coding efforts were predominantly carried out using Python and Java, particularly for app UI purposes.

Each of the models followed a generalized methodology as shown in Figure 2, for a more detailed description of each model’s methodology, refer to Table 2 further below. Additionally, pseudocodes (Python) for each model can be found in the appendix (Figures 20, 21, 22, and 23)

A screenshot of a video game

Description automatically generated

Figure 2 General methodology of ML models

A screen shot of a blue and white screen

Description automatically generated

Table 2 Model methodologies

## Network Intrusion Detection:

For the methodology of Network Intrusion Detection, the process begins with data preprocessing, encompassing techniques such as Minmax scaling, LabelEncoder encoding, and rigorous data cleaning procedures to ensure the integrity and uniformity of the dataset. Subsequently, the Mean Absolute Difference (MAD) feature selection algorithm is employed to identify the most relevant features for classification, thus mitigating the curse of dimensionality and enhancing model performance.

The core classification task is executed through the application of the Random Forest algorithm, renowned for its robustness and ability to handle high-dimensional data effectively. This ensemble learning technique harnesses the collective wisdom of multiple decision trees to discern patterns within the data and make accurate predictions regarding intrusion instances.

To evaluate the efficacy and generalization capabilities of the developed model, a rigorous 10-Fold Cross Validation strategy is employed. This involves partitioning the dataset into ten subsets, iteratively training the model on nine subsets while validating its performance on the remaining subset. Cross-validation ensures robustness against overfitting and provides a reliable estimate of the model's performance across different data partitions. Finally, the performance of the trained model is assessed on unseen data, thus simulating real-world deployment scenarios, and validating its effectiveness in detecting network intrusions beyond the confines of the training dataset. The flowchart for this task can be found in Figure 3.

A diagram of a training and training

Description automatically generated

Figure 3 Network intrusion detection flowchart

## Network Intrusion Classification:

In the Network Intrusion 4-Class Classification methodology, the process initiates with comprehensive preprocessing steps, including Minmax scaling, LabelEncoder encoding, and meticulous data cleaning to ensure dataset uniformity and integrity.

The core of the classification task is undertaken by a Deep Neural Network (DNN) architecture. This DNN is structured with three hidden layers, one input layer, and one output layer. Each layer employs Rectified Linear Unit (ReLU) activation functions, fostering nonlinear transformations essential for capturing complex relationships within the data. The final layer utilizes Softmax activation to yield probabilities across the four intrusion classes: 'Dos', 'Probe', 'R2L', and 'U2R'. The DNN serves multifaceted purposes within the classification pipeline. It acts as a tool for dimensionality reduction, extracting salient features from the input data, and facilitating intricate feature learning processes crucial for discerning nuanced patterns across the dataset. Moreover, the DNN architecture executes the 4-Class classification task, effectively categorizing network intrusions into their respective classes. The training of the DNN is optimized using the Adam Optimizer, which adaptively adjusts the learning rate to expedite convergence and mitigate the risks, early stopping mechanisms are incorporated with a patience of 6 steps, ensuring efficient training termination to prevent overfitting. Figure 4 and Figure 5 below visualize the DNN training graph plot and architecture respectively. We monitored both the training and validation loss/accuracy to indicate whether our model was effective before we even began to perform independent tests on unseen data. This was detrimental to our ability to saving valuable computing resources and time to be utilized in other aspects of our project such as the Android App.

A graph with a line and a line

Description automatically generated

Figure 4 DNN training plot of loss and accuracy

A diagram of a network

Description automatically generated

Figure 5 DNN architecture

To rigorously evaluate the model's performance and ensure its generalizability, a 10-Fold Cross Validation approach is employed. This involves partitioning the dataset into ten subsets, iteratively training and validating the model on different subsets to obtain robust performance estimates.

Finally, the trained model undergoes evaluation on unseen data, encompassing instances from the four predefined classes ('Dos', 'Probe', 'R2L', and 'U2R'). This step serves as a critical validation of the model's effectiveness in real-world intrusion classification scenarios, ensuring its reliability and applicability beyond the scope of the training dataset. The flowchart for this task can be found in figure 6.

A diagram of a training

Description automatically generated

Figure 6 Network intrusion classification flowchart

## Malware Detection:

For the methodology of Malware Detection, the process begins with rigorous preprocessing steps aimed at ensuring data consistency and integrity. This includes Minmax scaling, LabelEncoder encoding, and thorough data cleaning procedures.

Subsequently, the classification task is undertaken using the Random Forest algorithm, chosen for its robustness and ability to handle complex data structures effectively. Random Forest leverages an ensemble of decision trees to discern patterns within the data and make accurate predictions regarding the presence of malware.

To evaluate the performance of the developed model and ensure its generalizability, a 10-Fold Cross Validation strategy is employed. This involves partitioning the dataset into ten subsets, iteratively training the model on nine subsets while validating its performance on the remaining subset. Such cross-validation aids in averting overfitting and provides a reliable estimate of the model's efficacy across various data distributions. Finally, the trained model undergoes evaluation on unseen data to assess its effectiveness in detecting malware instances beyond the confines of the training dataset. This step serves as a critical validation of the model's performance in real-world scenarios, ensuring its reliability and applicability in practical deployment settings. The flow chart for this task can be found in Figure 7.

A diagram of a training and training

Description automatically generated

Figure 7 Malware detection flowchart

## Malware Classification:

In the methodology for Malware 9-Class Classification, the process begins with intricate Feature Extraction from raw .asm files and their corresponding .BYTES counterparts. This extraction process aims to capture the nuanced characteristics and behavioral patterns indicative of malware instances. Subsequently, the extracted features undergo rigorous preprocessing steps to ensure data uniformity and reliability. This includes Minmax scaling, LabelEncoder encoding, and meticulous data cleaning procedures.

To address class imbalance and enhance the model's performance, oversampling techniques are employed, specifically utilizing CTGAN (Conditional Tabular Generative Adversarial Network) synthetic data generation [25]. This approach augments the dataset with synthetic samples, ensuring more comprehensive coverage of minority classes without introducing bias.

The classification task is executed through a sophisticated Stacking classifier framework. At the base layer, multiple diverse classifiers: Naïve Bayes, Adaptive Boosting (AdaBoost), and eXtreme Gradient Boosting (XGBoost) are employed to capture different facets of the data's underlying distribution. These base classifiers provide diverse perspectives on the data, enhancing the model's ability to discern complex patterns. The predictions from the base classifiers serve as inputs to the meta layer, where a Random Forest classifier aggregates these predictions to make the final classification decision.

To rigorously assess the model's performance and ensure its generalizability, a 10-Fold Cross Validation strategy is employed. This involves partitioning the dataset into ten subsets, iteratively training the model on nine subsets while validating its performance on the remaining subset. Such cross-validation helps mitigate overfitting and provides a robust estimate of the model's efficacy across various data partitions. Finally, the trained model undergoes evaluation on unseen data to validate its effectiveness in classifying malware instances across the designated nine classes: 'Ramnit', 'Lollipop', 'Kelihos\_ver3', 'Vundo', 'Simda', 'Tracur', 'Kelihos\_ver1', 'Obfuscator.ACY', and 'Gatak'. This step serves as a crucial validation of the model's real-world applicability and reliability, ensuring its effectiveness in practical deployment scenarios, especially when utilized in the app. The flowchart for this task can be found in Figure 8. Additionally, the flowchart for the stacking model can be found in the appendix (Figure 15).

A diagram of a computer

Description automatically generated

Figure 8 Malware classification flowchart

## Evaluation:

Performance metrics included are:

* Accuracy - Accuracy measures how many predictions are correct out of all predictions made. It's a fundamental metric to assess the overall correctness of a model. High accuracy indicates that a model is making more correct predictions.
* Precision – Precision evaluates the ratio of true positives to the total number of positive predictions. It assesses how many of the positive predictions made by the model are correct. High precision indicates that the model has a low rate of false positives, which is crucial in cybersecurity to avoid false alarms.
* Recall (Sensitivity or True Positive Rate) – Recall measures the ability of the model to correctly identify all relevant instances (true positives) in the dataset. It's essential in cybersecurity to minimize false negatives, ensuring that actual intrusions or malware attacks are not missed.
* F1-Score – The F1-Score is the harmonic mean of precision and recall. It provides a balance between these two metrics. It's useful when you want to strike a balance between minimizing false positives and false negatives. A higher F1-Score indicates a model that performs well on both precision and recall.
* Area Under the Curve (AUC) – AUC is often used to evaluate the performance of binary classification models. It assesses the ability of the model to distinguish between positive and negative instances by looking at the Receiver Operating Characteristic (ROC) curve. A higher AUC indicates a better-performing model.
* Confusion Matrix – A confusion matrix is a tabular representation of the model's performance. It provides a detailed breakdown of true positives, true negatives, false positives, and false negatives. This can help in understanding where the model may be making errors, especially when dealing with imbalanced datasets.
* Cross-validation(k-folds) - Cross-validation is a technique used in machine learning to assess the performance of a model on unseen data. It involves dividing the dataset into k subsets, or "folds", and then training and testing the model k times, each time using a different fold as the test set and the remaining k-1 folds as the training set. A high cross-validation score indicates that the model can generalize well to unseen data, which is an important characteristic for any machine-learning model.

### The Importance of Evaluation:

In the context of our cybersecurity machine learning project, we employ a suite of evaluation metrics to gauge the robustness and consistency of our model's performance in detecting network intrusions and malware, ensuring the utmost security. These metrics help us scrutinize the accuracy and precision of our predictions concerning the actual values.

In assessing the effectiveness of feature selection and the overall performance of our training model, we rely on the recall and F1-Score. Recall allows us to measure our model's ability to identify true positives, essential in ensuring that actual network intrusions and malware instances are not missed. Meanwhile, the F1-Score provides a harmonious blend of precision and recall, ensuring that our model strikes a balance between minimizing false positives and false negatives.

The Area Under the Curve (AUC) is another pivotal metric in our arsenal. It offers a comprehensive summary of our model's ability to distinguish between malicious and non-malicious instances by examining the Receiver Operating Characteristic (ROC) curve. The ROC curve is a graphical representation that illustrates the trade-off between true positive rates and false positive rates at various thresholds. In simpler terms, it visually conveys how well our model separates the two classes. A higher AUC signifies superior model performance in discerning threats, indicating a larger area under the ROC curve and a more effective discrimination between positive and negative cases (the AUC/ROC metric is primarily used for binary classification, and not multi-class classification). In the context of the project, AUC score was utilized for both network intrusion detection and malware detection.

To further bolster the integrity of our results, we employ cross-validation as a technique that safeguards our training process from overfitting and ensures that our model derives only pertinent information. Cross-validation partitions our dataset into subsets, allowing us to train and test our model iteratively, thereby enhancing its generalizability.

Furthermore, as we compare our model's performance to existing research papers, we observe alignment with some studies, reinforcing the credibility of our approach, and prompting us to critically assess our model and fine-tune it to surpass current benchmarks in the field of cybersecurity.

# Results Analysis:

Following extensive testing and thorough research, our analysis reveals that each of our four models has surpassed the benchmarks set forth by contemporary literature in the field of cybersecurity research. This achievement underscores the efficacy and robustness of our approaches in addressing prevalent challenges within the cybersecurity domain. Moreover, our models have not only met but exceeded the standards established by current research papers. Through meticulous testing protocols and comprehensive evaluation metrics, we have demonstrated the superior performance and advancements achieved by our methodologies.

This accomplishment is a testament to the dedication and expertise invested in the development and refinement of our models. By surpassing established benchmarks, we have not only validated the efficacy of our approaches but have also contributed to pushing the boundaries of cybersecurity research forward.

Furthermore, this success underscores the importance of continuous innovation and exploration within the field. As cyber threats evolve and become increasingly sophisticated, it is imperative that our methodologies remain at the forefront of technological advancement. Through ongoing research and refinement, we are committed to further enhancing the effectiveness and resilience of our models to safeguard against emerging cyber threats. Table 3 provides a comprehensive overview of the performance of all four models. Confusion matrix heatmaps can further describe the model’s performance which is provided by Figures 16, 17, 18, and 19, and can be found later in the appendix.

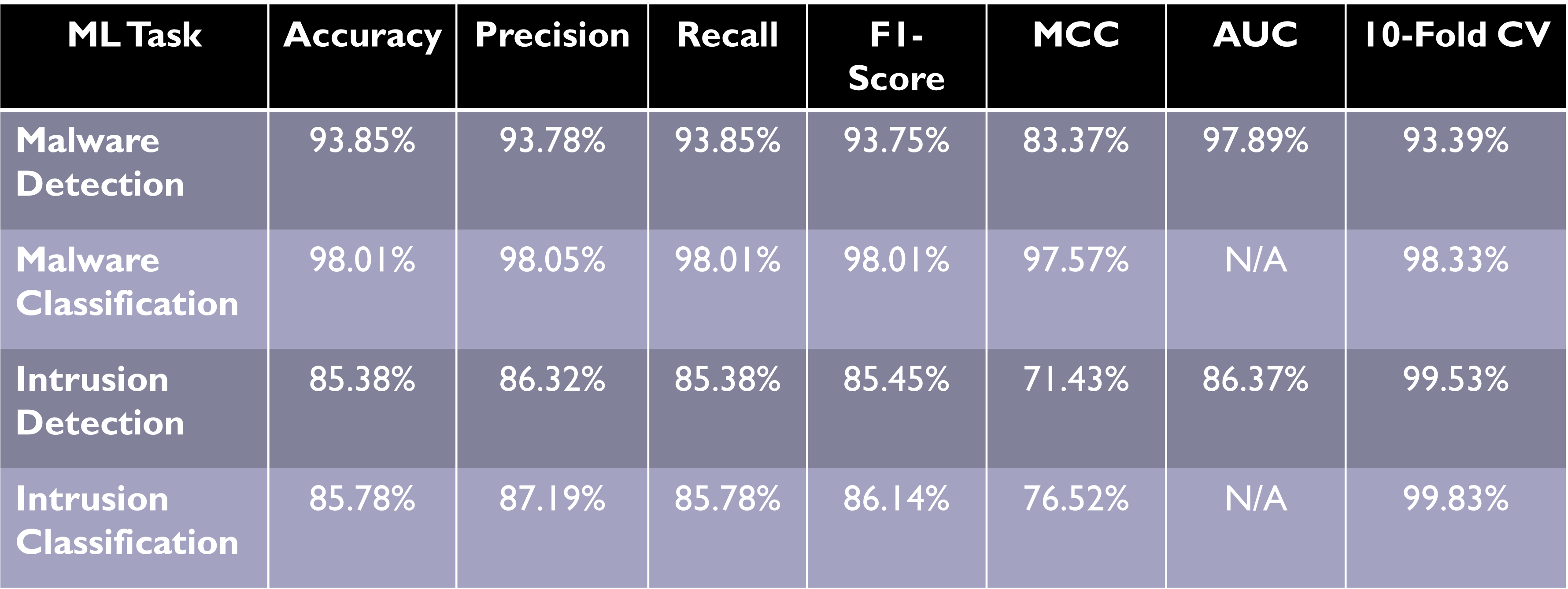


Table 3 Performance metrics of all four models

The performance evaluation across various tasks within the realm of cybersecurity presents a nuanced picture. In the malware detection task, the model showcases exceptional performance metrics, boasting accuracy, precision, recall, and F1-score all surpassing 93%. Such high scores signify the model's adeptness in accurately identifying instances of malware. Moreover, the Matthews Correlation Coefficient (MCC) and the Area Under the Curve (AUC) reinforce the model's discriminative power and its ability to differentiate between malware and benign instances. The commendable 10-fold cross-validation score of 93.39% suggests very effective generalization capabilities.

Similarly, in the malware classification task, the model excels with outstanding accuracy, precision, recall, and F1-score, all reaching an impressive 98.01%. The MCC further affirms the model's effectiveness in correctly classifying various types of malware.

However, when shifting focus to the intrusion detection task, although the model performs reasonably well with an accuracy of 85.38% and other metrics hovering around the same range, there is noticeable room for improvement compared to the malware-related tasks. The MCC and AUC scores suggest that the model's discriminative power could be enhanced further. Nonetheless, the high 10-fold cross-validation score indicates good generalization capabilities.

Intrusion classification, too, mirrors similar performance levels as intrusion detection, albeit with slightly higher metrics. The model exhibits an accuracy of 85.78%, precision of 87.19%, recall of 85.78%, and F1-score of 86.14%. Although moderately strong, the MCC suggests potential for improvement.

In summary, while the models excel in malware detection and classification, the intrusion-related tasks demonstrate competent performance but also highlight areas for refinement. Continued research and model enhancement efforts hold promise in further fortifying the intrusion detection and classification tasks, ultimately contributing to the development of more robust and accurate cybersecurity systems. Tables 4, 5, 6, and 7 will provide comparisons with other established research papers for network intrusion detection, intrusion classification, malware detection, and malware classification respectively.

## Network Intrusion Detection Comparison with Contemporary Studies:

For network intrusion detection (Table 4), our proposed approach excels with an accuracy of 85.38%, the highest among the compared methods. It outperforms techniques like Decision Tree (80.14%), Fuzzy Classifier (82.74%), and AE-Based Feature Learning (83.34%). While the proposed algorithm in Experiment-2 (84.12%) comes close, our approach demonstrates superiority in binary classification tasks.

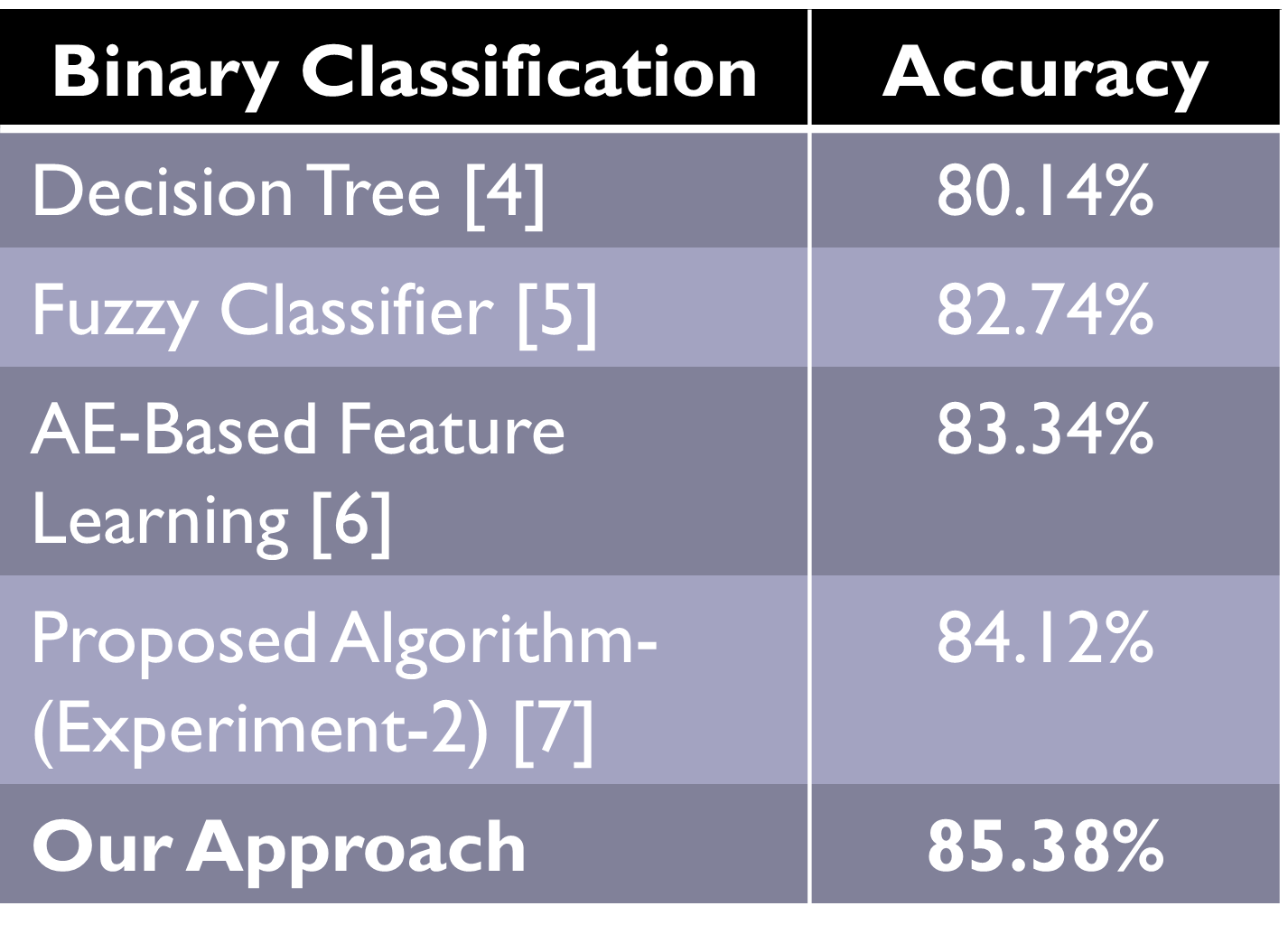


Table 4 Network intrusion detection comparisons

## Network Intrusion Classification Comparison with Contemporary Studies:

In network intrusion classification (Table 5), our proposed approach achieves an impressive accuracy of 85.78%, surpassing the majority of compared methods such as CNN + MLP (71.88%), SVM-SMOTE + DNN (80.47%), RepTree (83.59%) , and ANN + Decision Tree (85.40%), indicating its competitive edge and effectiveness in handling multiclass classification tasks.

A black and white rectangular box with white text

Description automatically generated

Table 5 Network intrusion classification comparisons

## Malware Detection Comparison with Contemporary Studies:

In the malware detection (Table 6), which is a binary classification problem, our proposed approach achieves an accuracy of 93.85%, outperforming other methods listed such as Dilated CNN (83.53%), Opcode Histogram (90.00%), Classical VGG-16 (91.31%), and Statistic-Based (92.75%). This demonstrates our approach's capability in accurately detecting malware instances, surpassing the performance of the compared techniques.

A black and white rectangular text with white text

Description automatically generated with medium confidence

Table 6 Malware detection comparisons

## Malware Classification Comparison with Contemporary Studies:

Lastly, in malware classification (Table 7), our proposed approach achieves an impressive accuracy of 98.01%, significantly outperforming other methods such as Malgazer (95.00%), Hierarchical Learning (95.78%), Behavioral Frequency (96.60%), and Multi-Layered RF (96.84%). This superior performance underscores the effectiveness of our approach in handling complex multiclass classification problems.

A black and white rectangular box with white text

Description automatically generated

Table 7 Malware classification comparisons

## Website Integration:

Following the development of our suite of machine learning models, we proceeded to establish a user-centric website that serves as the central hub for disseminating our Android application, "MTD." This intuitive app seamlessly integrates all the previously mentioned machine learning models, consolidating their functionalities into one cohesive and user-friendly interface.

Furthermore, recognizing the importance of user education and empowerment in cybersecurity, our website goes beyond merely offering our application for download. It serves as a comprehensive resource hub, providing users with access to a wealth of educational materials and pertinent information. These resources are designed to equip users with a deeper understanding of our work and to impart essential cybersecurity knowledge and best practices.

By offering a holistic approach that combines practical application with educational resources, our platform not only enhances users' cybersecurity defenses through our app but also empowers them with the knowledge and tools necessary to navigate the digital landscape safely and securely. Through this initiative, we aim to foster a community of informed and proactive users who are better equipped to protect themselves against cyber threats. Figures 9 and 10 provide some screenshots of the website layout. Another appearance of the website can be found in the appendix (Figure 14).

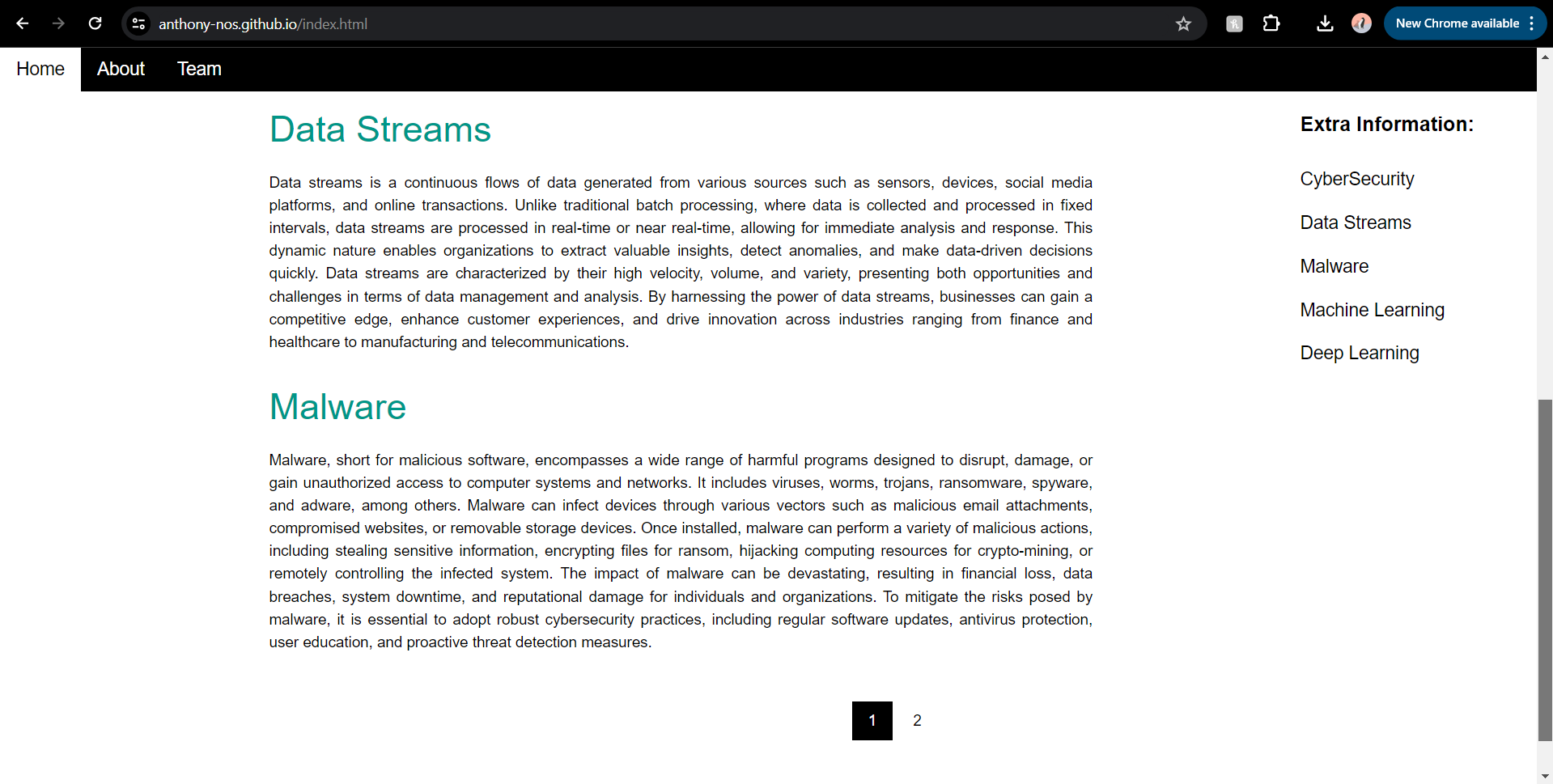


Figure 9 MTD website homepage



Figure 10 MTD website about page

## Android App Integration:

A screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedAs we reached the final stages of our development cycle, we embarked on the creation of a fully operational application that represents the culmination of our endeavors. This application serves as a sophisticated combinational use of advanced machine learning models honed to proficiently combat an array of cyber threats. Through rigorous training and optimization, these models demonstrate remarkable efficacy and reliability, instilling trust in their ability to safeguard digital environments effectively. Screenshots of the app’s user interface can be found in Figures 11 and 12, along with an additional image found in the appendix (Figure 13)

Figure 11 MTD android app performance page

Figure 12 MTD android app homepage

Our application not only embodies cutting-edge technology but also prioritizes user accessibility as a core tenet of its design philosophy. With a keen focus on user experience, we meticulously crafted an intuitive interface and incorporated user-friendly features to ensure seamless navigation and interaction. This deliberate approach ensures that individuals from all backgrounds, including those within the IoT community, can harness the full potential of our application with ease.

By placing equal emphasis on accessibility and functionality, we have succeeded in developing a versatile tool that empowers users to fortify their digital landscapes with confidence and ease. Through our commitment to excellence in both technology and user experience, we aim to redefine the standard for cybersecurity applications, ushering in a new era of digital protection and peace of mind for users worldwide.

# Conclusion:

In conclusion, our project, Multi-Threat Detection (M.T.D.), presents a comprehensive cybersecurity solution designed to combat the ever-evolving landscape of cyber threats that permeate our daily lives. By adopting a holistic approach, we address common vulnerabilities with a keen focus on real-world relevance and practicality. Central to our endeavor is the integration of diverse datasets, ensuring that our solutions resonate with real-world scenarios. Additionally, our Android app empowers users to bolster their device security, offering tangible protection against malicious cyber threats.

Machine learning and deep learning serve as the backbone of our project, with a meticulous assessment of various algorithms such as Random Forest, Support Vector Machines, and Artificial Neural Networks. Through rigorous evaluation metrics and continuous refinement, we ascertain the effectiveness and robustness of our models. This commitment to evaluation and improvement not only bolsters the credibility of our project but also ensures its sustained effectiveness in safeguarding against cybersecurity challenges.

In essence, M.T.D. represents a concerted effort to provide users with a proactive defense against cyber threats, underpinned by cutting-edge technology and a commitment to ongoing enhancement. As we navigate the complexities of the digital landscape, our project stands as a beacon of resilience and innovation in the fight for cybersecurity.

# Acknowledgement:

We extend our deepest appreciation to Dr. David Hicks, who served as our instructor for the CSEN 4202 Senior Design Course at Texas A&M University-Kingsville (TAMUK). Dr. Hicks's consistent guidance, unwavering support, and extensive expertise have been instrumental in shaping the trajectory of our senior project. His mentorship has been a beacon, providing invaluable insights and direction that have profoundly impacted the success of our endeavor.

We are profoundly grateful for Dr. Hicks's depth of knowledge and profound insights. His thorough understanding of computer science, coupled with his ability to simplify intricate concepts, has significantly enriched our comprehension of the complex challenges within our project scope.

As seniors in the computer science program here at TAMUK, this senior design journey has been pivotal in our academic growth. Under Dr. Hicks's guidance, we have immersed ourselves in the intricate details of our project, focusing on enhancing cyber security threats mitigation through artificial intelligence-based applications. This exploration has expanded our intellectual horizons and equipped us with practical insights that extend far beyond the classroom, preparing us for our future careers.

In acknowledging Dr. Hicks's contributions, we recognize the profound impact he has had on our academic journey, igniting in us a passion for exploration and a dedication to excellence. We express our sincerest gratitude for his mentorship, which has been fundamental to our academic development, and for the doors of knowledge and discovery that this senior design project has opened. Additionally, we extend our thanks to Texas A&M University-Kingsville and all other professors who have provided us with the necessary knowledge and support throughout our academic endeavors.

# Appendix:

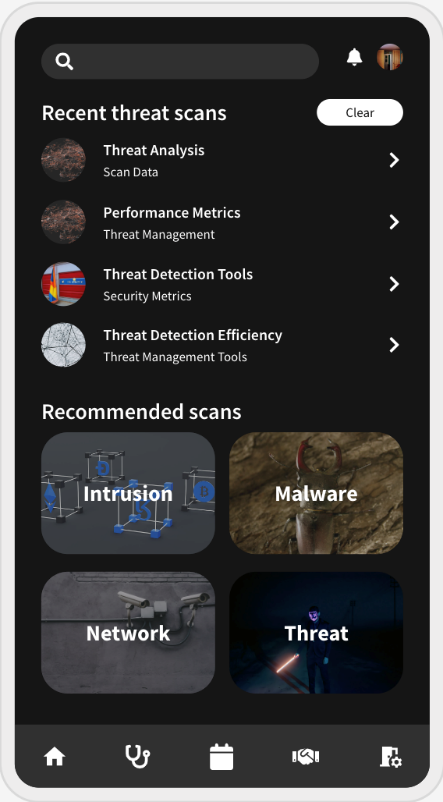


Figure 13 MTD android app recent scans page

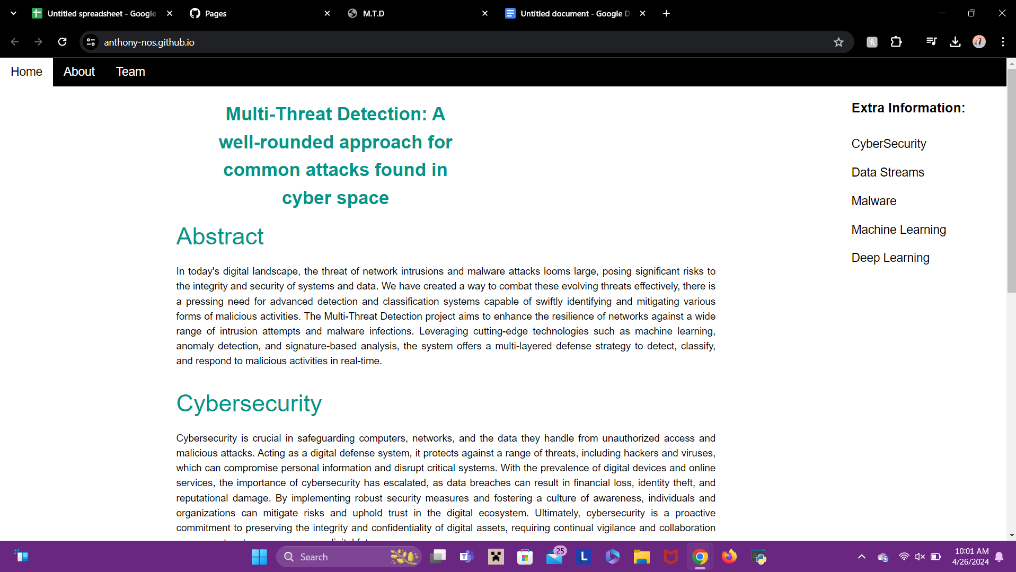


Figure 14 MTD homepage (different view)

A diagram of a software company

Description automatically generated with medium confidence

Figure 15 Stacking model flowchart

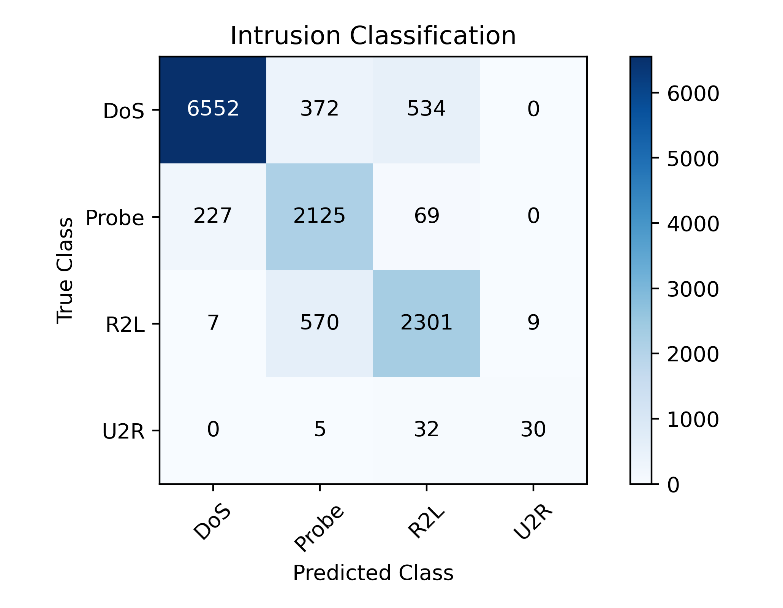
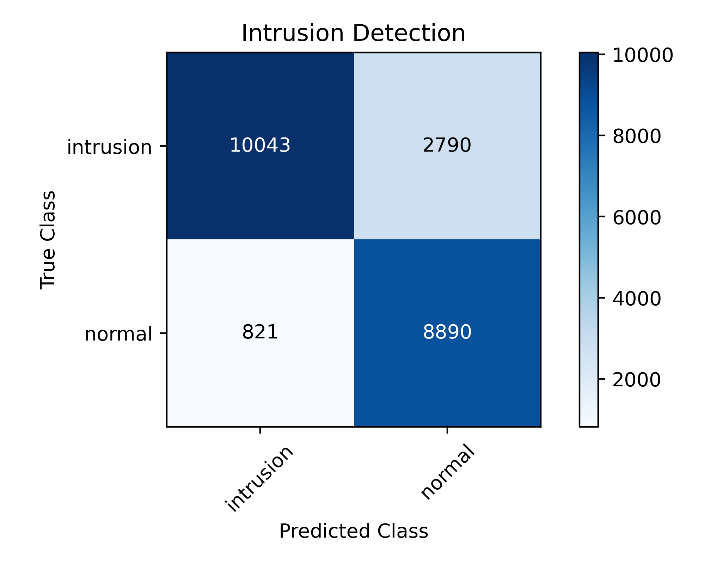


Figure 16 Intrusion detection heatmap

Figure 17 Intrusion classification heatmap

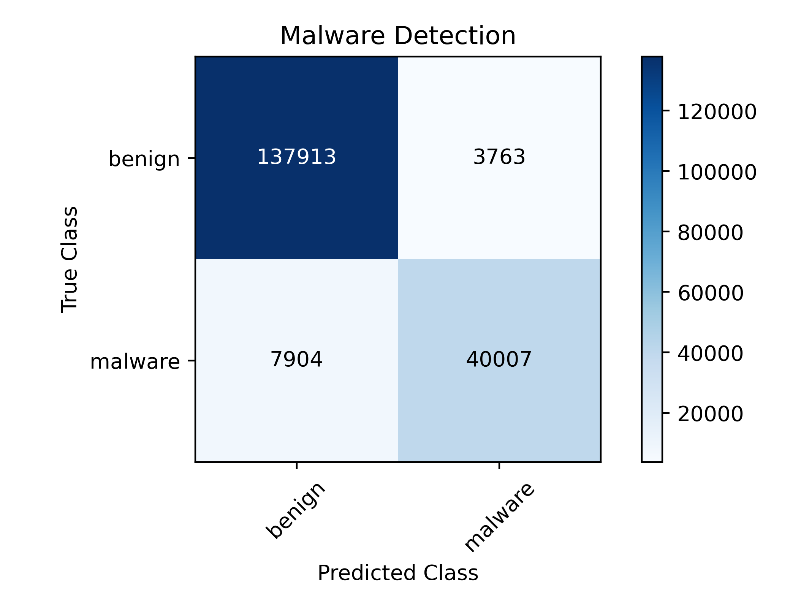
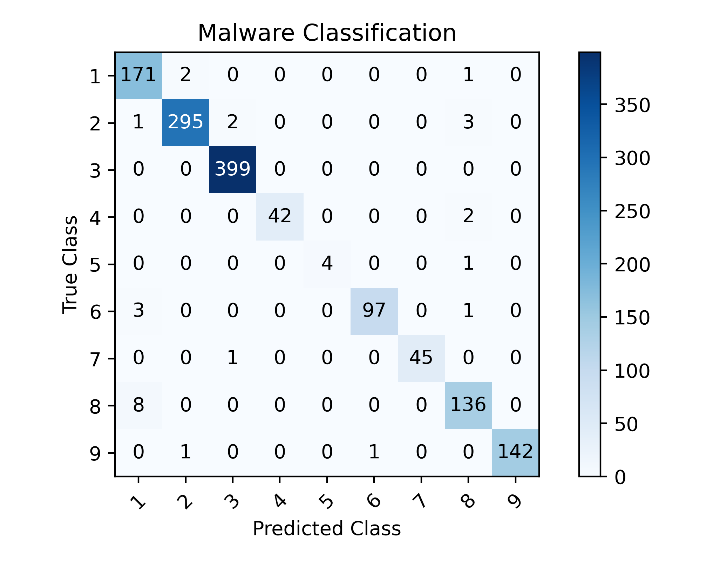


Figure 18 Malware classification heatmap

Figure 19 Malware detection heatmap

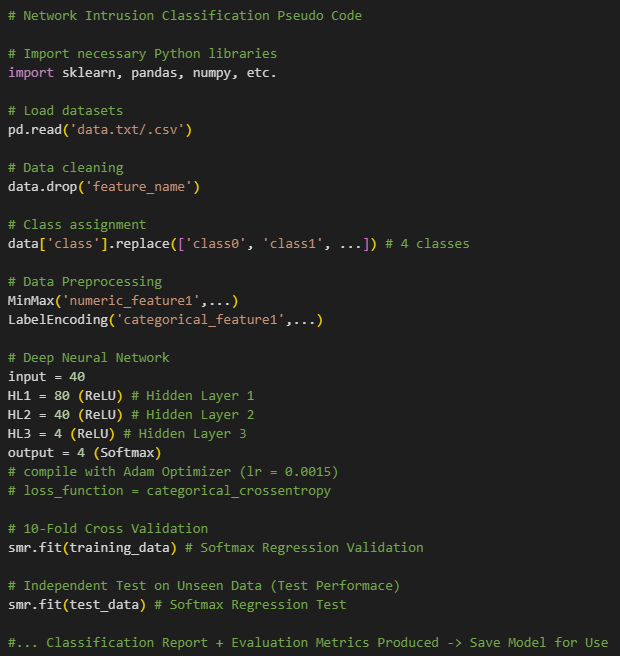


Figure 20 Intrusion classification pseudocode

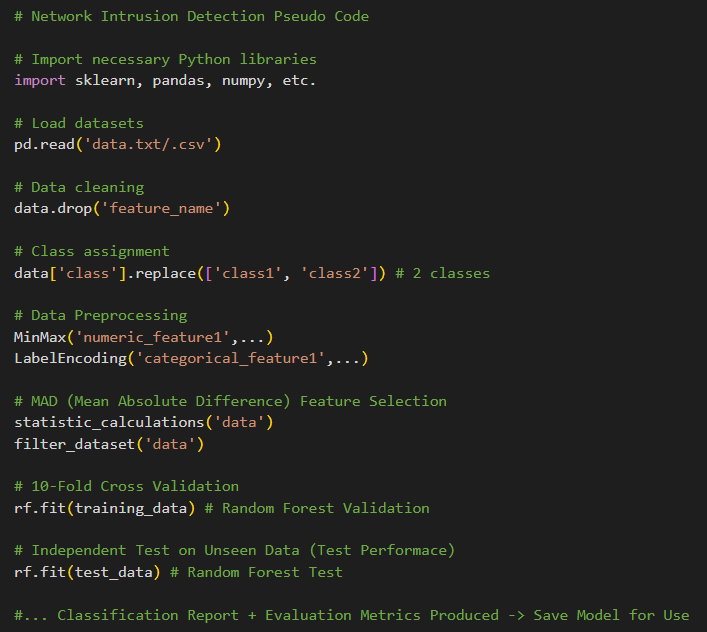


Figure 21 Intrusion detection pseudocode

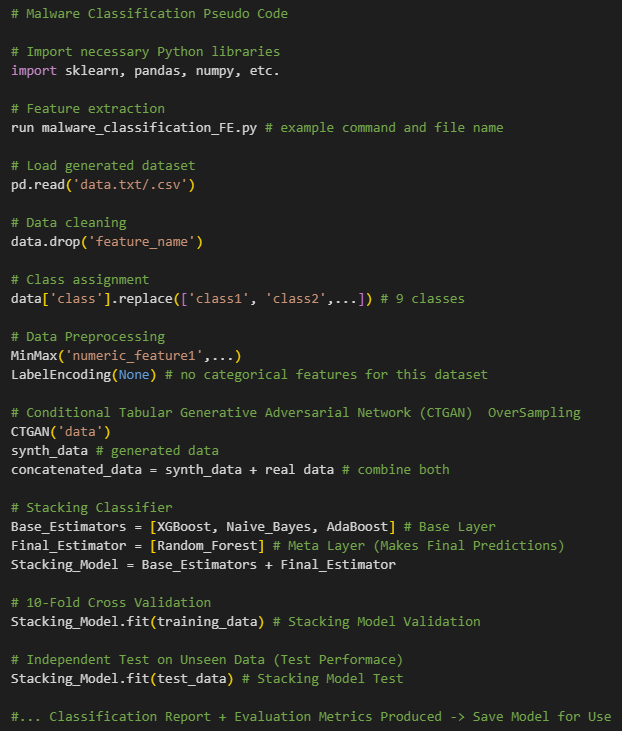


Figure 22 Malware classification pseudocode

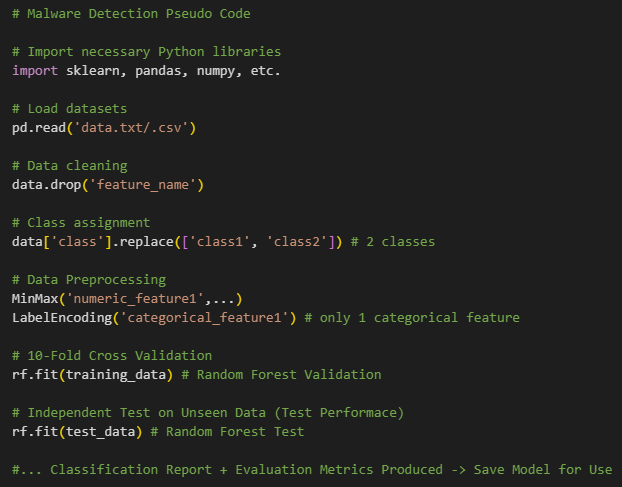


Figure 23 Malware detection pseudocode

# References:

1. M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, “A Detailed Analysis of the KDD CUP 99 Data Set,” Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.
2. Arash Habibi Lashkari, Andi Fitriah A. Kadir, Hugo Gonzalez, Kenneth Fon Mbah and Ali A. Ghorbani, “Towards a Network-Based Framework for Android Malware Detection and Characterization”, In the proceeding of the 15th International Conference on Privacy, Security and Trust, PST, Calgary, Canada, 2017.
3. Alessandro Panconesi, Marian, Will Cukierski, WWW BIG - Cup Committee. (2015). MicrosoftMalware Classification Challenge (BIG 2015). Kaggle.

https://kaggle.com/competitions/malware-classification

1. M. Mohammadi, B. Raahemi, A. Akbari, and B. Nassersharif, “Class dependent feature transformation for intrusion detection systems,” in 2011 19th Iranian Conference on Electrical Engineering. IEEE, 2011, pp. 1–6.
2. P. Kr¨omer, J. Platoˇ s, V. Sn´ aˇ sel, and A. Abraham, “Fuzzy classification by evolutionary algorithms,” in Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on. IEEE, 2011, pp. 313–318.
3. M. Yousefi-Azar, V. Varadharajan, L. Hamey and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 2017, pp. 3854-3861, doi: 10.1109/IJCNN.2017.7966342.
4. R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, “Fuzziness based semi-supervised learning approach for intrusion detection system,” Information Sciences, 2016.
5. R. Blanco, P. Malagón, J. J. Cilla and J. M. Moya, "Multiclass Network Attack Classifier Using CNN Tuned with Genetic Algorithms," 2018 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Platja d'Aro, Spain, 2018, pp. 177-182, doi: 10.1109/PATMOS.2018.8463997.
6. 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), Austin, TX, USA, 2021, pp. 1-6, doi: 10.1109/IPCCC51483.2021.9679415
7. Mustapha Belouch, Salah El Hadaj and Mohamed Idhammad. A TwoStage Classifier Approach Using RepTree Algorithm for Network Intrusion Detection. In International Journal of Advanced Computer Science and Applications, Volume 8.6, 2017
8. Y. Zhang, Y. Gandhi, Z. Li and Z. Xiao, "Improving the Classification Effectiveness of Network Intrusion Detection Using Ensemble Machine Learning Techniques and Deep Neural Networks," 2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA), San Antonio, TX, USA, 2022, pp. 117-123, doi: 10.1109/IDSTA55301.2022.9923205.
9. A. Mezina and R. Burget, "Obfuscated malware detection using dilated convolutional network," 2022 14th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Valencia, Spain, 2022, pp. 110-115, doi: 10.1109/ICUMT57764.2022.9943443.
10. S. Gülmez and I. Sogukpinar, "Graph-Based Malware Detection Using Opcode Sequences," 2021 9th International Symposium on Digital Forensics and Security (ISDFS), Elazig, Turkey, 2021, pp. 1-5, doi: 10.1109/ISDFS52919.2021.9486386.
11. S. Alvee, B. Ahn, S. Ahmad, K. Kim, T. Kim, and J. Zeng, “Devicecentric firmware malware detection for smart inverters using deep transfer learning,” in Proc. 2022 IEEE Design Methodologies Conferences, Bath U.K., Sep. 1-2, 2022, pp. 1-5
12. X. D. Hoang, B. C. Nguyen and T. T. Trang Ninh, "Detecting Malware Based on Statistics and Machine Learning Using Opcode N-Grams," 2023 RIVF International Conference on Computing and Communication Technologies (RIVF), Hanoi, Vietnam, 2023, pp. 118-123, doi: 10.1109/RIVF60135.2023.10471824.
13. K. J. Jones and Y. Wang, "Malgazer: An Automated Malware Classifier With Running Window Entropy And Machine Learning," 2020 Sixth International Conference on Mobile And Secure Services (MobiSecServ), Miami Beach, FL, USA, 2020, pp. 1-6, doi: 10.1109/MobiSecServ48690.2020.9042957.
14. S. Chakraborty, J. W. Stokes, L. Xiao, D. Zhou, M. Marinescu and A. Thomas, "Hierarchical learning for automated malware classification," MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, USA, 2017, pp. 23-28, doi: 10.1109/MILCOM.2017.8170758.
15. A. Walker and S. Sengupta, "Insights into Malware Detection via Behavioral Frequency Analysis Using Machine Learning," MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM), Norfolk, VA, USA, 2019, pp. 1-6, doi: 10.1109/MILCOM47813.2019.9021034.
16. S. A. Roseline, A. D. Sasisri, S. Geetha and C. Balasubramanian, "Towards Efficient Malware Detection and Classification using Multilayered Random Forest Ensemble Technique," 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, 2019, pp. 1-6, doi: 10.1109/CCST.2019.8888406.
17. C. -W. Chen, C. -H. Su, K. -W. Lee and P. -H. Bair, "Malware Family Classification using Active Learning by Learning," 2020 22nd International Conference on Advanced Communication Technology (ICACT), Phoenix Park, Korea (South), 2020, pp. 590-595, doi: 10.23919/ICACT48636.2020.9061419.
18. M. H. Rahman, L. Martinez III, A. Mishra, M. Nijim, A. Goyal and D. Hicks, "Enhancing Intrusion Detection through Deep Learning and Generative Adversarial Network" accepted in 4. Interdisciplinary Conference on Electrics and Computer (INTCEC 2024), 11-13 June 2024, Chicago-USA
19. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

https://scikit-learn.org/stable/about.html#citing-scikit-learn

1. TensorFlow publishes a DOI for the open-source code base using Zenodo.org: 10.5281/zenodo.4724125
2. *Notion*, Connected Workspace for Effective Collaboration: <https://www.notion.so/product>
3. CTGAN Documentation: <https://github.com/sdv-dev/CTGAN>
4. G. Zhang, X. Wang, R. Li, J. Lai, Q. Xiang and J. He, "Network Intrusion Detection Method Based on Stacked Denoising Sparse Autoencoder and Extreme Learning Machine," *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, Guangzhou, China, 2020, pp. 194-199, doi: 10.1109/ITCA52113.2020.00048.
5. W. Meng, "Intrusion Detection in the Era of IoT: Building Trust via Traffic Filtering and Sampling," in *Computer*, vol. 51, no. 7, pp. 36-43, July 2018, doi: 10.1109/MC.2018.3011034.
6. P. I. Radoglou-Grammatikis and P. G. Sarigiannidis, "Flow anomaly based intrusion detection system for Android mobile devices," *2017 6th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, Thessaloniki, Greece, 2017, pp. 1-4, doi: 10.1109/MOCAST.2017.7937625.
7. T. Wisanwanichthan and M. Thammawichai, "A Double-Layered Hybrid Approach for Network Intrusion Detection System Using Combined Naive Bayes and SVM," in *IEEE Access*, vol. 9, pp. 138432-138450, 2021, doi: 10.1109/ACCESS.2021.3118573.
8. M. Vishwakarma, and N. Kesswani, “A new two-phase intrusion detec tion system with Na¨ ıve Bayes machine learning for data classification and elliptic envelop method for anomaly detection,” Decision Analytics Journal, 7, p.100233, 2023.
9. V. Sidharth and C. R. Kavitha, "Network Intrusion Detection System Using Stacking and Boosting Ensemble Methods," *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, 2021, pp. 357-363, doi: 10.1109/ICIRCA51532.2021.9545022.
10. P. M. S. Sánchez, A. H. Celdrán, G. Bovet, G. M. Pérez and B. Stiller, "SpecForce: A Framework to Secure IoT Spectrum Sensors in the Internet of Battlefield Things," in *IEEE Communications Magazine*, vol. 61, no. 5, pp. 174-180, May 2023, doi: 10.1109/MCOM.001.2200349.
11. S. Otoum, B. Kantarci and H. T. Mouftah, "On the Feasibility of Deep Learning in Sensor Network Intrusion Detection," in *IEEE Networking Letters*, vol. 1, no. 2, pp. 68-71, June 2019, doi: 10.1109/LNET.2019.2901792.
12. G. Karatas, O. Demir and O. Koray Sahingoz, "Deep Learning in Intrusion Detection Systems," *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, Ankara, Turkey, 2018, pp. 113-116, doi: 10.1109/IBIGDELFT.2018.8625278.
13. S. Ding, Y. Wang and L. Kou, "Network intrusion detection based on BiSRU and CNN," *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, Denver, CO, USA, 2021, pp. 145-147, doi: 10.1109/MASS52906.2021.00026.
14. M. A. Siddiqi and W. Pak, "Tier-Based Optimization for Synthesized Network Intrusion Detection System," in *IEEE Access*, vol. 10, pp. 108530-108544, 2022, doi: 10.1109/ACCESS.2022.3213937.
15. M. S. Deshmukh and P. R. Bhaladhare, "Intrusion Detection System (DBN-IDS) for IoT using Optimization Enabled Deep Belief Neural Network," *2021 5th International Conference on Information Systems and Computer Networks (ISCON)*, Mathura, India, 2021, pp. 1-4, doi: 10.1109/ISCON52037.2021.9702505.