Machine Learning Algorithms for Network Intrusion Detection and a Corresponding Interactive Dashboard

Gotam Sai Varshith
22BCE1605
Department of Computer Science

PAVAN KRISHNA R
24BCE5294
Department of Computer Science

MRINAL SWAIN
24BCE5357
Department of Computer Science

Abstract

- **Context:** The contemporary digital ecosystem is characterized by an exponential increase in network-interconnected devices and services, creating a vast and complex cyber landscape. In parallel, the actors perpetrating cyber threats have evolved, employing highly sophisticated, automated, and persistent attack methodologies, including Advanced Persistent Threats (APTs), polymorphic malware, and zero-day exploits that are specifically designed to evade conventional security measures. This escalating threat level mandates a paradigm shift from static, reactive security postures to dynamic, proactive, and intelligent defense mechanisms. Network Intrusion Detection Systems (NIDS) represent a critical and indispensable layer in this modern cybersecurity architecture. Functioning as a vigilant second line of defense, NIDS continuously monitor inbound and outbound network traffic to identify suspicious patterns and malicious activities that may have successfully bypassed perimeter defenses such as firewalls. However, the efficacy of traditional signature-based NIDS is fundamentally constrained by their reliance on a pre-defined database of known attack patterns. These systems are inherently incapable of identifying novel, previously unseen attack vectors, rendering them vulnerable to the very threats that are most prevalent in today's environment.
- **Objective:** This project undertakes a comprehensive, empirical investigation into the application and efficacy of a suite of classical machine learning algorithms for the development of a high-performance, anomaly-based Network Intrusion Detection System. The central objective is to move beyond theoretical postulation and to practically implement, rigorously evaluate, and systematically compare the performance of three distinct paradigms of supervised learning. The selected models are: K-Nearest Neighbors (KNN), representing instance-based learning; Linear Discriminant Analysis (LDA), representing a statistical, generative approach; and the Support Vector Machine (SVM), representing a margin-based classification method. The performance of these models is benchmarked on the widely recognized and academically validated NSL-KDD dataset, providing a standardized and reproducible basis for comparison.
- **Methodology:** The project adheres to a structured and disciplined data science pipeline, ensuring a methodical and replicable approach from data ingestion to model evaluation. The process commences with an in-depth exploratory data analysis (EDA) of the NSL-KDD dataset to understand its statistical properties, feature distributions, and class imbalances. Following this, a rigorous and multi-stage data preprocessing phase is executed. This critical phase involves the application of one-hot encoding to transform non-numeric, categorical features (such as protocol_type, service, and flag) into a numerical format suitable for machine learning, thereby preventing the introduction of erroneous ordinal relationships. Concurrently, Min-Max normalization is applied to all numerical features to scale them to a uniform range of [0, 1], a crucial step to prevent features with large magnitudes from disproportionately influencing the learning process of distance-based algorithms. Subsequently, the fully preprocessed KDDTrain+.txt dataset is utilized to train each of the three models (KNN, LDA, and SVM). To ensure an unbiased assessment of their

generalization capabilities, the trained models are then rigorously evaluated against the entirely unseen KDDTest+.txt dataset.

- **Implementation:** Transcending a purely theoretical evaluation, this project culminates in the design and development of a practical, high-fidelity, and interactive web dashboard. This deployable proof-of-concept is built using the Python-native Streamlit framework, chosen for its capacity for rapid development and intuitive user interface creation. The core application, encapsulated in the app.py script, provides an end-to-end solution for on-demand network traffic analysis. It allows a security analyst or user to upload network traffic data in a standard CSV format via a user-friendly interface. In real-time, the application's backend leverages the best-performing pre-trained model—which has been serialized and saved along with the necessary preprocessing assets (scaler.pkl, model_columns.pkl)—to classify each connection as either 'Normal' or 'Attack'. The dashboard is designed for maximum clarity, presenting a high-level executive summary of the analysis, including key performance indicators like the total number of connections analyzed and the precise count of threats detected. Furthermore, it provides a detailed, granular, and visually highlighted row-by-row breakdown of the prediction results, making the model's output both actionable and interpretable.
- **Conclusion:** The empirical results obtained from the rigorous evaluation phase unequivocally demonstrate the significant potential and high efficacy of applying machine learning techniques for the construction of robust and effective intrusion detection systems. The project provides quantitative evidence that data-driven approaches can successfully model complex network behavior and distinguish between benign and malicious activities with a high degree of accuracy. A detailed comparative analysis of the models is presented, based on a suite of standard performance metrics including Accuracy (for overall correctness), Precision (for minimizing false positives), Recall (for minimizing false negatives), and the F1-Score (for a balanced measure of precision and recall). This multi-faceted evaluation allows for a nuanced understanding of the unique strengths and weaknesses of each algorithm in the context of network security. The project concludes by formally identifying the most suitable algorithm from the tested set based on this empirical evidence and proposes a detailed, forward-looking roadmap for future work. This roadmap includes critical next steps such as the integration of more advanced deep learning architectures (e.g., LSTMs, Autoencoders), testing on more contemporary datasets (e.g., CIC-IDS2017), and the ultimate development of a real-time, live-packet-capture detection pipeline.

Keywords

- Network Security, Intrusion Detection System (IDS), Machine Learning, Anomaly Detection, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Linear Discriminant Analysis (LDA), NSL-KDD Dataset, Cybersecurity, Streamlit.

I. INTRODUCTION

- **A. Background and Motivation**
  - The Proliferation of Digital Connectivity and the Modern Network Infrastructure:

- The contemporary era is defined by an unprecedented and accelerating proliferation of digital connectivity, a phenomenon that has fundamentally reshaped global commerce, communication, and societal structures.
- Networks are no longer peripheral support systems; they have evolved into the central nervous system of modern infrastructure, underpinning critical sectors ranging from finance and healthcare to energy and transportation.
- The advent of the Internet of Things (IoT), cloud computing, and ubiquitous mobile access has exponentially increased the number of interconnected devices, creating an immeasurably vast and complex attack surface.
- This hyper-connectivity, while a catalyst for innovation and efficiency, has concurrently exposed these critical infrastructures to a continuously evolving and increasingly sophisticated landscape of cyber threats, making network security a paramount concern for global stability and progress.

- The Significance and Impact of Cyber Threats:
  - The risks associated with this digital dependency are profound and multifaceted. Unauthorized access to sensitive networks can lead to catastrophic data breaches, resulting in the theft of intellectual property, personal identifiable information (PII), and confidential corporate data.
  - Service disruptions, often orchestrated through Denial of Service (DoS) attacks, can cripple essential services, leading to significant financial losses, reputational damage, and, in some cases, threats to public safety.
  - The economic impact of cybercrime is staggering, with global costs estimated in the trillions of dollars annually, encompassing not only direct financial theft but also the costs of remediation, regulatory fines, and loss of customer trust.
  - These escalating risks necessitate a transition from a passive security posture to a proactive, intelligent, and resilient framework, making robust and adaptive network security an indispensable strategic requirement for organizations and nation-states alike.
- The Role of Network Intrusion Detection Systems (NIDS) in a Layered Security Architecture:
  - No single security measure is infallible. Therefore, modern cybersecurity relies on a "defense-in-depth" or layered security strategy, where multiple, overlapping security controls are implemented to protect assets.
  - Network Intrusion Detection Systems (NIDS) are a critical and non-negotiable component of this layered strategy. They function as a vigilant surveillance mechanism, operating behind preventative perimeter controls like firewalls and access control lists.
  - The primary function of a NIDS is to continuously monitor and analyze network traffic in real-time to identify malicious activities, policy violations, or anomalous behaviors that may have successfully bypassed these initial lines of defense.
  - By providing timely and actionable alerts, NIDS empower security analysts to respond swiftly to potential security incidents, thereby mitigating damage, containing threats, and facilitating post-incident forensic analysis.

- **B. Problem Statement**
  - The Inherent Limitations of Traditional Signature-Based Detection:
    - The predominant methodology employed by traditional Network Intrusion Detection Systems is signature-based detection. This approach operates analogously to antivirus software, where incoming network packets are meticulously inspected and compared against a vast, predefined database of known attack patterns, or "signatures."
    - While this method is highly effective and computationally efficient at identifying well-known and documented threats, its efficacy is fundamentally predicated on the existence of a prior signature for an attack.
    - The fundamental and critical limitation of this approach is its inherent inability to detect novel, polymorphic, or sophisticated "zero-day" attacks. These are attacks that exploit previously unknown vulnerabilities, and for which no signature has yet been developed or distributed.
    - In an era where attackers can rapidly generate new malware variants and exploit chains, this reliance on historical knowledge creates a significant and dangerous security gap, leaving networks vulnerable during the critical period between the discovery of a new threat and the deployment of its corresponding signature.
  - The Promise and Dynamics of Anomaly-Based Detection:
    - To address the shortcomings of signature-based systems, anomaly-based detection offers a more dynamic, adaptive, and forward-looking alternative.
    - Instead of relying on a blacklist of known bad patterns, this paradigm involves creating a robust, statistical baseline model of "normal" or legitimate network behavior. This baseline comprehensively characterizes typical traffic patterns, protocols, and data flows within the network.
    - The core principle is that any significant deviation from this established baseline of normalcy is flagged as a potential anomaly or intrusion. This approach is, by its nature, capable of detecting novel attacks without any prior knowledge of their specific characteristics.
    - It effectively shifts the detection focus from identifying known threats to identifying abnormal behavior, a far more powerful and future-proof strategy against the evolving threat landscape.
  - The Synergy Between Machine Learning and Anomaly-Based NIDS:
    - Machine Learning (ML) is an exceptionally well-suited and powerful technology for the construction of high-fidelity, anomaly-based NIDS.
    - ML algorithms possess the intrinsic ability to autonomously learn complex, non-linear patterns and subtle correlations from vast, high-dimensional datasets without requiring explicit, rule-based programming. This is particularly advantageous for analyzing network traffic, which is characterized by its high volume, velocity, and complexity.

        - By training on large corpuses of network data, ML models can independently build the baseline of normal behavior and subsequently identify deviations with a high degree of sensitivity and accuracy.
        - The core problem that this project seeks to address is therefore the systematic and empirical evaluation and comparison of several well-established, classical machine learning algorithms to scientifically determine their suitability, performance trade-offs, and overall effectiveness for building a high-performance, anomaly-based NIDS.

- **C. Project Contributions**
    - Comprehensive and Replicable Data Preprocessing:
        - This project delivers a meticulously designed and thoroughly documented data preprocessing pipeline, specifically tailored for the unique characteristics of the NSL-KDD dataset.
        - This includes the implementation of robust feature engineering techniques such as one-hot encoding for categorical variables to prevent model bias, and Min-Max normalization for numerical features to ensure equitable feature contribution, thereby laying a solid and reproducible foundation for the modeling phase.
    - Rigorous Empirical Model Comparison:
        - The project provides a rigorous, scientifically grounded, side-by-side performance evaluation of three distinct and widely-used paradigms of supervised classification algorithms: the instance-based K-Nearest Neighbors (KNN), the statistical Linear Discriminant Analysis (LDA), and the margin-based Support Vector Machine (SVM).
        - This empirical comparison is conducted using a standardized, unseen test set and a comprehensive suite of performance metrics (Accuracy, Precision, Recall, F1-Score), ensuring that the conclusions drawn are objective, quantifiable, and directly comparable.
    - Practical Implementation and Bridging the Research-Application Gap:
        - A significant contribution of this project is the translation of the theoretical analysis into a tangible, deployable, and user-friendly tool.
        - The development of the interactive Streamlit dashboard serves as a powerful proof-of-concept, demonstrating the entire end-to-end pipeline of how a trained machine learning model can be successfully operationalized and deployed for practical, real-world use by security analysts or network administrators.
    - Commitment to Open and Reproducible Science:
        - The entire project, from the initial stages of data acquisition and exploration to the final application deployment and code implementation, is thoroughly and meticulously documented.
        - This commitment to detailed documentation ensures maximum clarity, facilitates the reproducibility of the experimental results by other researchers, and enhances the educational value of the project for students and practitioners in the field of cybersecurity and machine learning.

- **D. Paper Structure**
  - Section II: Literature Survey
    - This section will provide a comprehensive review of the existing body of academic and industry literature concerning the application of machine learning and data mining techniques in the domain of network security and intrusion detection, establishing the context and foundation for this work.
  - Section III: Methodology and System Architecture
    - This section will meticulously describe the complete methodology employed in the project. It will include a detailed overview of the benchmark NSL-KDD dataset, a step-by-step breakdown of the data preprocessing pipeline, and an exposition of the theoretical foundations and operational mechanics of the machine learning models utilized.
  - Section IV: Implementation Details
    - This section will provide an in-depth account of the practical implementation of the project. It will detail the development of the Python-based model training and evaluation pipeline using Scikit-learn and the construction of the interactive front-end dashboard using the Streamlit framework.
  - Section V: Results and Discussion
    - This section will present and critically discuss the empirical results obtained from the model evaluation phase. It will feature a detailed comparative analysis of the performances of the KNN, LDA, and SVM models, complete with visualizations and a discussion of the performance trade-offs.
  - Section VI: Conclusion and Future Work
    - This final section will conclude the paper by summarizing the key findings and contributions of the project. It will also outline a strategic and promising roadmap for future research and development, including potential enhancements and new avenues of investigation.

II. LITERATURE SURVEY

- **A. Evolution of Intrusion Detection**
  - Foundations in Rule-Based Expert Systems:
    - Early academic and commercial research in the field of intrusion detection, dating back several decades, was predominantly rooted in the paradigm of expert systems. These initial systems were meticulously handcrafted, rule-based engines.
    - Security experts would manually codify their domain knowledge into a set of explicit IF-THEN rules. For instance, a rule might state: IF a single IP address makes more

than 100 failed login attempts in one minute, THEN flag it as a potential brute-force attack.

        - The core of these systems was a knowledge base containing hundreds or thousands of such rules, which were manually curated and updated. This process was extremely labor-intensive, time-consuming, and highly dependent on the continuous availability of specialized human expertise.
        - While effective for known threat vectors, these rule-based systems were inherently brittle. They struggled to adapt to new or slightly modified attack techniques and were prone to generating a high volume of false positives if the rules were not perfectly calibrated to the specific network environment.
    - The Paradigm Shift to Data-Driven Methodologies:
        - The transition from purely manual, rule-based systems to more automated, data-driven approaches marked a significant inflection point in the evolution of NIDS. This shift was catalyzed by two primary technological advancements.
        - First, the increasing availability of large-scale, labeled network traffic datasets provided the raw material necessary for training and validating statistical and machine learning models. The creation of public repositories for these datasets fostered a competitive and collaborative research environment.
        - Second, the exponential growth in computational power, as described by Moore's Law, made it feasible to apply complex, computationally intensive algorithms to these massive datasets. Tasks that were once computationally prohibitive, such as training complex classifiers or clustering terabytes of traffic data, became achievable.
        - This transition effectively moved the field from a model of "knowledge engineering," where human expertise was manually encoded, to a model of "machine learning," where the system could autonomously learn the patterns of normal and malicious behavior directly from the data itself.

- **B. Benchmark Datasets in NIDS Research**
    - The KDD Cup 1999 Dataset: A Foundational Landmark:
        - The KDD Cup 1999 dataset is widely regarded as a foundational landmark in the history of NIDS research. It was one of the first comprehensive, publicly available datasets specifically created for the evaluation of intrusion detection systems.
        - Its release provided a crucial common ground and a standardized benchmark, allowing researchers from around the world to objectively compare the performance of their diverse algorithms on a level playing field for the first time.
        - However, subsequent in-depth analyses revealed significant flaws within the dataset. The most severe criticism was the presence of a massive number of redundant records. Approximately 78% of the records in the training set and 75% in the test set were duplicates.
        - This redundancy had a detrimental effect on model evaluation, as it could severely bias classifiers. A model might achieve an artificially high accuracy simply by learning to favor the more frequent, redundant attack types, while performing poorly on rarer but more critical attacks like U2R or R2L.

- The NSL-KDD Dataset: A Refined and More Realistic Benchmark:
  - The NSL-KDD dataset was meticulously crafted and introduced as a direct response to the identified limitations of its predecessor. It is not a new dataset from scratch but rather an improved, refined version of KDD'99.
  - The primary improvement was the complete removal of duplicate records from both the training and testing sets. This crucial step ensures that the models are evaluated on a more diverse and non-redundant set of instances, preventing the aforementioned bias.
  - By removing the redundant records, the NSL-KDD dataset provides a more balanced and realistic class distribution. This makes the evaluation task more challenging and a better reflection of a real-world scenario, where a classifier must be effective against a variety of attack types, not just the most common ones.
  - Due to these significant improvements, the NSL-KDD dataset has become a de facto standard benchmark in contemporary NIDS research. It remains a widely used and respected dataset for evaluating and comparing the performance of new and existing machine learning algorithms for intrusion detection.
- The Emergence of Modern Datasets for Contemporary Threats:
  - While NSL-KDD remains a valuable benchmark, the nature of network traffic and cyber attacks has evolved dramatically since its creation. Recognizing this, researchers have developed more recent and sophisticated datasets.
  - The CIC-IDS2017 dataset, for example, was designed to capture more realistic and modern network traffic patterns. It includes traffic generated from a variety of contemporary attack vectors, such as Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attacks (including SQL Injection and XSS), and botnet infiltration.
  - Similarly, the UNSW-NB15 dataset was created to address the need for a more comprehensive benchmark that includes a wider variety of modern, low-footprint attack types. It contains a hybrid of real-world normal network traffic and synthetically generated attack traffic, encompassing nine distinct families of attacks.
  - These newer datasets are crucial for the future of NIDS research, as they provide a more challenging and accurate environment for testing the ability of machine learning models to cope with the complexity and subtlety of today's cyber threats.

- **C. Application of Machine Learning in NIDS**
  - Broad Spectrum of Algorithmic Exploration:
    - The application of machine learning to intrusion detection is a rich and diverse field of study. A vast number of academic papers and studies have explored the efficacy of a wide array of ML algorithms for this task.
    - These studies span the full spectrum of machine learning paradigms, including supervised learning (where models are trained on labeled data), unsupervised learning (where models must find patterns in unlabeled data, ideal for anomaly detection), and semi-supervised learning (which uses a combination of labeled and unlabeled data).

- The choice of algorithm often depends on the specific goals of the NIDS, such as whether it is designed for a specific type of network, the computational resources available, and the desired trade-off between detection accuracy and false alarm rates.

- K-Nearest Neighbors (KNN): The Instance-Based Approach:
  - The K-Nearest Neighbors (KNN) algorithm has been frequently noted in the literature for its conceptual simplicity and surprisingly high effectiveness in NIDS contexts.
  - Numerous studies applying KNN to the NSL-KDD dataset have reported high accuracy rates, demonstrating its capability to effectively classify network connections in a high-dimensional feature space.
  - However, a significant and consistently cited drawback of KNN is its computational cost during the prediction phase. As a "lazy learner," KNN performs no upfront model building. Instead, it must compute the distance from a new data point to every single point in the training dataset to make a prediction. For large datasets, this can make real-time prediction prohibitively slow and resource-intensive.
- Support Vector Machines (SVM): The Margin-Based Classifier:
  - Support Vector Machines (SVM) are a class of powerful supervised learning models that are frequently praised in NIDS literature for their strong theoretical foundations and excellent empirical performance.
  - Their ability to find an optimal decision boundary (hyperplane) that maximizes the margin between different classes makes them particularly robust, especially in high-dimensional feature spaces like those found in network traffic data with dozens of features.
  - Research has consistently shown SVM to be a strong and reliable performer in binary classification tasks for NIDS (i.e., distinguishing 'Normal' from 'Attack').
  - The power of SVMs can be further enhanced through the use of the "kernel trick," which allows them to model complex, non-linear relationships in the data, although this project focused on a computationally efficient linear kernel.
- **Linear Discriminant Analysis (LDA): The Statistical Approach:**
  - Linear Discriminant Analysis (LDA) is a versatile statistical technique that serves a dual purpose. While it is primarily known as a feature extraction and dimensionality reduction technique, it can also be directly used as a powerful and computationally efficient classifier.
  - Studies have shown LDA to be an effective method for NIDS, often providing performance comparable to more complex models but with significantly lower training and prediction times.
  - However, the effectiveness of LDA is predicated on several underlying statistical assumptions, namely that the data for each class follows a Gaussian (normal) distribution and that all classes share the same covariance matrix.
  - This assumption of a linear relationship between features may not always hold true for the intricate and often non-linear patterns present in complex network attack data, which can be a potential limitation of this approach.

III. METHODOLOGY AND SYSTEM ARCHITECTURE

- **A. System Workflow**
  - Adherence to a Structured Methodological Framework:
    - The project's execution is underpinned by a structured and systematic workflow, mirroring established best practices in data science and machine learning project management. This adherence to a formal process ensures that each phase of the project is conducted with rigor, that the results are reproducible, and that the overall architecture is logical and transparent.
    - The workflow is designed as a sequential pipeline, where the output of each stage serves as the validated input for the subsequent stage, minimizing errors and ensuring the integrity of the final deployed system. The architecture is visually depicted in a system workflow diagram to provide a clear, high-level overview of the end-to-end process.
  - Phase 1: Data Acquisition:
    - The foundational phase of the project involves the procurement of a suitable dataset for training and evaluating the intrusion detection models.
    - After a thorough review of available benchmark datasets, the NSL-KDD dataset is selected. This choice is deliberate, based on its established reputation in the academic community and its specific design to overcome the statistical limitations of the earlier KDD Cup 1999 dataset.
    - The dataset is procured from a trusted academic repository, ensuring its integrity and authenticity. Both the designated training set (KDDTrain+.txt) and the separate, unseen testing set (KDDTest+.txt) are acquired to facilitate a robust and unbiased evaluation of model performance.
  - Phase 2: Data Preprocessing:
    - This phase is arguably the most critical, as the quality and format of the data directly impact the performance and reliability of any machine learning model.
    - The raw, unprocessed data from the NSL-KDD dataset is subjected to a comprehensive, multi-step pipeline of cleaning, transformation, and normalization.
    - This includes the meticulous handling of categorical data through one-hot encoding, the conversion of the multi-class problem into a binary classification problem through label encoding, and the scaling of all numerical features to a uniform range using Min-Max normalization. Each of these steps is detailed further in Section III-C.
  - Phase 3: Model Training:
    - In this phase, the fully preprocessed, clean, and normalized training data is used to train the selected machine learning models.
    - Three distinct supervised learning algorithms are implemented using the Scikit-learn library in Python: K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), and the Support Vector Machine (SVM).
    - Each model is trained independently on the entire preprocessed training dataset, allowing each algorithm to learn the underlying patterns and decision boundaries that differentiate between 'Normal' and 'Attack' network traffic.

- Phase 4: Model Evaluation:
  - To assess the generalization capability and real-world effectiveness of the trained models, they are subjected to a rigorous evaluation phase.
  - The models are evaluated on the preprocessed unseen test data from KDDTest+.txt. Using a separate and unseen test set is crucial to prevent data leakage and obtain an unbiased estimate of the models' performance on new data.
  - A suite of standard, quantitative performance metrics is employed, including Accuracy, Precision, Recall, and F1-Score. This multi-metric approach provides a holistic view of each model's strengths and weaknesses, moving beyond a simple accuracy score.
- Phase 5: Deployment:
  - The final phase bridges the gap between theoretical modeling and practical application.
  - Based on the results of the evaluation phase, the best-performing model is identified (in this case, the one with the highest accuracy and F1-score).
  - This champion model, along with its corresponding data scaler and the list of feature columns, is serialized and saved to disk using the joblib library.
  - These saved assets are then integrated into a fully interactive and user-friendly web application, developed using the Streamlit framework, thereby deploying the model for on-demand analysis and prediction.

- **B. Dataset Description: NSL-KDD**
  - Origin and Purpose:
    - The NSL-KDD dataset is a refined and improved version of the seminal KDD Cup 1999 dataset, which was a pioneering resource in the field of intrusion detection research.
    - It was specifically created to address and rectify the statistical anomalies present in the KDD'99 dataset, most notably the presence of a vast number of redundant records which tended to bias classifiers towards more frequent attack types.
  - Structure and Contents:
    - The dataset is composed of a large collection of individual network connection records, where each record represents a sequence of TCP packets over a period of time, flowing between a source and destination IP address under a well-defined protocol.
    - Each of these connection records is described by a vector of 41 features, providing a rich, multi-dimensional representation of the connection's behavior.
    - The dataset is pre-partitioned into a training set (KDDTrain+.txt) and a distinct test set (KDDTest+.txt), facilitating standardized and comparable evaluations across different research studies.
  - Feature Categorization:
    - The 41 features are logically categorized into three distinct groups based on their nature and how they are derived:

        - Basic Features (9 features): These are the intrinsic, fundamental attributes of an individual TCP connection itself, derived directly from the packet headers without any deeper inspection of the payload. This category includes features like the duration of the connection in seconds, the protocol_type used (e.g., TCP, UDP, ICMP), the destination network service (e.g., HTTP, FTP, private), and the connection status flag (e.g., SF for normal, REJ for rejected).
        - Content Features (13 features): These features require a deeper inspection of the TCP packet's payload and are based on domain knowledge of network protocols and attack methodologies. This category includes features such as num_failed_logins, which counts unsuccessful login attempts, logged_in, a binary flag indicating a successful login, and root_shell, a flag indicating if a root shell was obtained.
        - Traffic Features (19 features): These features are computed statistically over a window of recent connections, providing a temporal context that is crucial for detecting certain types of attacks. They are designed to capture patterns that emerge over time. This category includes "same-host" features, which analyze connections to the same destination host in the last two seconds (e.g., same_srv_rate), and "same-service" features, which analyze connections using the same service.
    - Classification Labels:
        - Each record in the dataset is meticulously labeled, falling into one of two main categories: 'Normal' for legitimate traffic, or a specific attack type.
        - The dataset includes a diverse set of 22 different attack types, which are further aggregated into four broad, well-known categories of network attacks, providing a hierarchical classification structure:
        - Denial of Service (DoS): Attacks designed to overwhelm a target system's resources, rendering it unable to respond to legitimate requests. Examples from the dataset include smurf, neptune, and teardrop.
        - Probe (Probing): Attacks where an adversary systematically scans a network to gather information about potential vulnerabilities, active hosts, and open ports. Examples include portsweep and nmap.
        - Remote to Local (R2L): Attacks in which an attacker, operating from a remote machine, exploits a vulnerability to gain unauthorized local user access on a target machine. Examples include ftp_write and guess_passwd.
        - User to Root (U2R): Attacks where an attacker has already gained local user-level access on a system and subsequently exploits a vulnerability to escalate their privileges to the root (administrator) level. Examples include loadmodule and perl.

- **C. Data Preprocessing Pipeline**
    - The Imperative of Preprocessing:
        - Raw network traffic data, as captured in datasets like NSL-KDD, is inherently heterogeneous and unstructured. It contains a mix of numerical and categorical data, features with vastly different scales, and labels that need to be transformed for specific modeling tasks.
        - Directly feeding this raw data into machine learning models would lead to poor performance, model bias, and potential computational errors. Therefore, a robust, multi-

step preprocessing pipeline is an essential and non-negotiable prerequisite for successful model training.

- Step 1: Feature Encoding (Transformation of Categorical Data):
  - The NSL-KDD dataset contains three critical features that are non-numeric: protocol_type, service, and flag. These features represent discrete categories.
  - Since machine learning algorithms are mathematical models that operate on numerical tensors, these categorical features must be converted into a numerical representation.
  - A naive approach like integer encoding (e.g., tcp=0, udp=1, icmp=2) would impose an artificial ordinal relationship that does not exist, implying that 'udp' is somehow "greater" than 'tcp'. This would mislead the model.
  - To circumvent this, one-hot encoding was employed. This technique transforms each categorical feature into a set of new binary (0 or 1) features, one for each unique category. For example, the protocol_type feature would be expanded into three new features: protocol_type_tcp, protocol_type_udp, and protocol_type_icmp. This creates a sparse but accurate numerical representation without implying any false order.
- Step 2: Label Encoding (Problem Framing):
  - The target variable, the 'class' column, is multi-class, containing the label 'normal' and 22 distinct attack labels.
  - For the scope of this project, the primary objective was to build a system that can effectively distinguish any malicious activity from normal behavior. Therefore, the problem was framed as a binary classification task.
  - This was achieved through a straightforward label encoding process: the 'normal' label was mapped to the integer 0, and all 22 attack labels were consolidated and mapped to the integer 1. This transformation simplifies the problem and allows for the use of standard binary classification models and metrics.
- Step 3: Data Normalization (Feature Scaling):
  - An examination of the numerical features in the dataset reveals that they exist on vastly different scales and ranges. For instance, the src_bytes feature can have values in the millions, while rate-based features like serror_rate are bounded between 0 and 1.
  - This discrepancy in scales can severely degrade the performance of many machine learning algorithms, especially those that rely on distance calculations, such as K-Nearest Neighbors, or those that use gradient-based optimization, such as Support Vector Machines. Features with larger magnitudes can disproportionately dominate the learning process.
  - To address this, Min-Max Scaling was applied to all numerical features. This technique linearly transforms each feature to a common range, typically [0, 1]. The transformation is performed using the formula: $X_{scaled}=(X-X_{min})/(X_{max}-X_{min})$.
  - This crucial step ensures that every feature contributes equally to the model's decision-making process, leading to a more robust, stable, and accurate model.

- **D. Machine Learning Models**
  - 1) K-Nearest Neighbors (KNN):

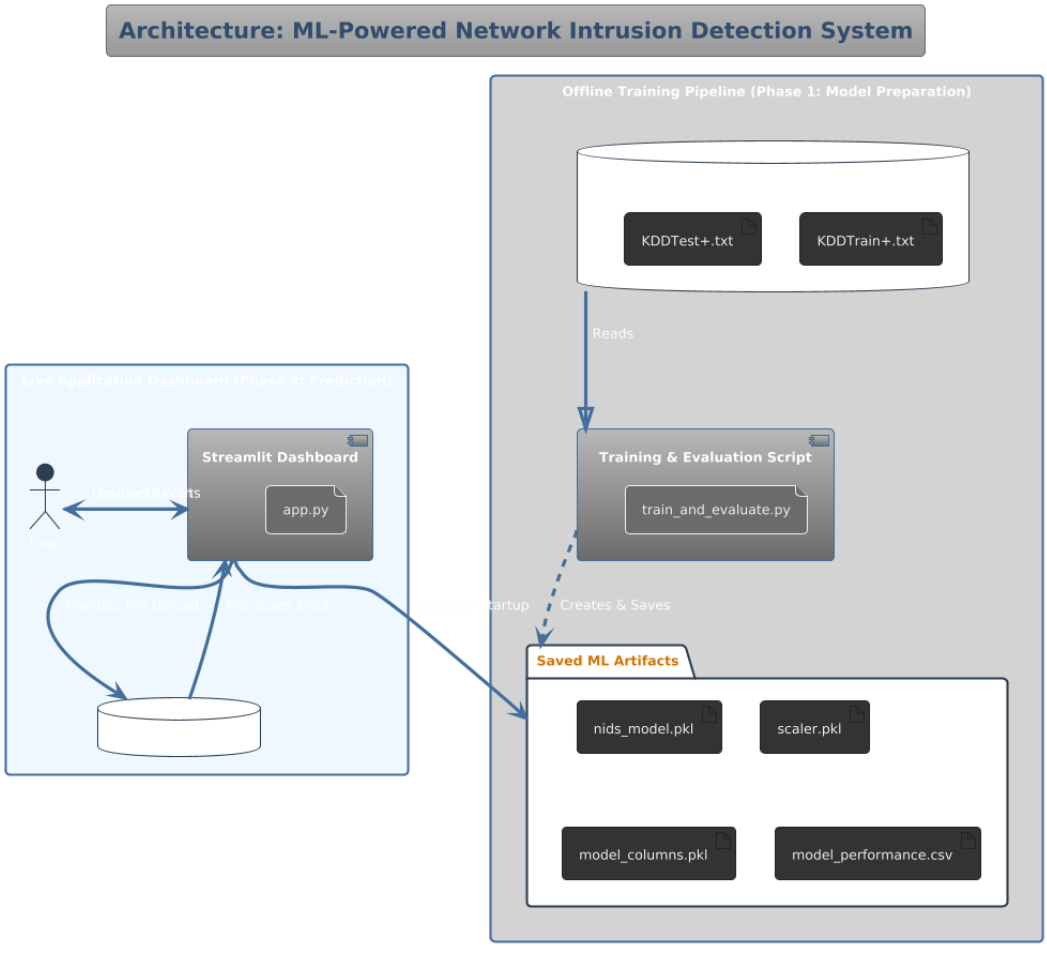
- Fundamental Principle: KNN is a non-parametric, supervised learning algorithm that is fundamentally based on the principle of feature similarity. It is one of the simplest yet surprisingly effective classification algorithms.
- Operational Mechanism: Unlike many other algorithms, KNN does not learn an explicit discriminative function from the training data. Instead, it memorizes the entire training dataset. When it needs to classify a new, unseen data point, it calculates the distance between this point and every single point in the training data. It then identifies the 'k' training data points that are closest (the "nearest neighbors").
- Classification Process: The classification of the new data point is determined by a majority vote among its 'k' identified neighbors. For instance, if k=5 and three of the five nearest neighbors belong to the 'Attack' class, the new point will also be classified as 'Attack'.
- Core Component - Distance Metric: The notion of "closeness" or "similarity" is quantified using a distance metric. The most common choice, and the one used in this project, is the Euclidean distance, which is the straight-line distance between two points in the multi-dimensional feature space.
- Critical Hyperparameter 'k': The choice of 'k', the number of neighbors to consider, is a critical hyperparameter that can significantly affect the model's performance. A small 'k' can make the model sensitive to noise, while a large 'k' can be computationally expensive and may oversmooth the decision boundary. For this project, a conventional and widely-used value of k=5 was selected.
- Algorithmic Nature: KNN is often referred to as a "lazy learner" because it performs no computation during the training phase; it simply stores the data. All the intensive work of distance calculation and voting happens at prediction time, which can be a significant performance bottleneck in real-time systems with large datasets.

o 2) Linear Discriminant Analysis (LDA):

- Fundamental Principle: LDA is a classical statistical method that serves as both a classifier and a dimensionality reduction technique. It is a generative model, meaning it models the distribution of the data for each class.
- Operational Mechanism: The primary goal of LDA is to find a lower-dimensional feature space and a projection of the data onto this space that maximizes the separability between the different classes. It achieves this by finding a projection that maximizes the ratio of the between-class variance (how far apart the means of the different classes are) to the within-class variance (how spread out the data is within each class).
- Underlying Assumptions: The mathematical foundation of LDA rests on two key assumptions about the data: first, that the data for each class is drawn from a multivariate Gaussian (normal) distribution, and second, that all classes share the same covariance matrix. While these assumptions may not hold perfectly for real-world data, LDA often performs remarkably well even when they are slightly violated.
- Classification Process: Once the optimal projection (the linear discriminant) is found, LDA can classify new data points by projecting them into the new feature space and assigning them to the class whose mean is closest. This makes it an efficient and powerful linear classifier.

o 3) Support Vector Machine (SVM):

- Fundamental Principle: The Support Vector Machine is a highly effective and widely used supervised learning algorithm known for its strong theoretical guarantees and excellent empirical performance, particularly on high-dimensional data.
- Operational Mechanism: The central idea of SVM is to find an optimal hyperplane that separates the data points of different classes in the feature space. A hyperplane is a decision boundary; for a two-dimensional space, it is a line, and for a three-dimensional space, it is a plane.
- The Concept of the Optimal Hyperplane: While there might be many hyperplanes that can separate the data, the "optimal" hyperplane is the one that has the largest margin. The margin is defined as the distance between the hyperplane and the nearest data points from either class. A larger margin generally leads to better generalization performance on unseen data.
- The Role of Support Vectors: The specific training data points that are closest to the optimal hyperplane are called the support vectors. These are the most critical data points in the dataset because they are the ones that define the position and orientation of the hyperplane. All other data points are irrelevant to the final model.
- The Kernel Trick: For this project, a linear kernel was employed, which means the model attempts to find a linear decision boundary. This is computationally efficient and works well when the data is mostly linearly separable. More complex, non-linear relationships can be modeled by SVMs through the use of other kernels, such as the Polynomial or Radial Basis Function (RBF) kernel, in a technique known as the "kernel trick."



IV. IMPLEMENTATION AND APPLICATION DASHBOARD

- **A. Model Training and Evaluation Pipeline (train_and_evaluate.py)**
  - **Purpose:** This Python script serves as the core engine for training, evaluating, and saving the models.
  - **Execution Flow:**
    1. The script begins by loading the KDDTrain+.txt and KDDTest+.txt datasets using the Pandas library.
    2. It executes the full data preprocessing pipeline as described in Section III-C.
    3. It initializes all three machine learning models (KNN, LDA, SVM) using the Scikit-learn library.
    4. It iterates through each model, training it on the preprocessed training data.
    5. Each trained model is then evaluated on the preprocessed, unseen test data.
    6. The performance metrics (Accuracy, Precision, Recall, F1-Score) for each model are calculated and stored.
    7. These performance results are saved to a file named model_performance.csv.
    8. The script identifies the best-performing model based on the highest accuracy.
    9. Finally, it saves the best model (nids_model.pkl), the data scaler (scaler.pkl), and the list of training columns (model_columns.pkl) to disk using joblib for later use by the web application.
- **B. Interactive Dashboard (app.py)**
  - **Technology:** The front-end application is built entirely in Python using the **Streamlit** framework.
  - **Functionality:**
    1. **Asset Loading:** Upon startup, the application loads the saved best model, scaler, columns list, and the model performance CSV file.
    2. **User Interface:** The UI presents a clean, two-column dashboard layout.
    3. **File Uploader:** The primary interactive element is a file uploader that allows the user to select and upload a CSV file containing network traffic data.
    4. **Conditional Display:** The main analysis content, including the model performance table and prediction results, is **conditionally rendered** and only appears after a file has been successfully uploaded.
    5. **Real-Time Prediction:** Once a file is uploaded, the application:
       - Reads the CSV into a Pandas DataFrame.
       - Applies the same preprocessing steps (encoding and scaling) using the loaded assets.
       - Feeds the processed data into the loaded nids_model.pkl to generate predictions.
    6. **Results Visualization:** The results are displayed in a user-friendly, tabbed format:

        - **Prediction Summary Tab:** Shows high-level metrics, including the total number of connections analyzed, the total attacks found, and a clear warning or success message.
        - **Detailed Results Tab:** Presents a scrollable table with the original data and an added 'Prediction' column. Rows predicted as 'Attack' are highlighted in red for easy identification.
        - **Input Data Tab:** Shows a preview of the raw data the user uploaded.
    - **Unique Feature:** The dashboard also displays the static model_performance.csv table, providing context to the user about why a particular model was chosen for the predictions.

V. RESULTS AND DISCUSSION

- **A. Performance Metrics**
    - To quantitatively evaluate the models, the following standard classification metrics were used:
    - **Accuracy:** The ratio of correctly predicted instances to the total number of instances. It provides a general measure of the model's overall performance.
        - Formula: Accuracy=TP+TN+FP+FNTP+TN**Error! Filename not specified.**
    - **Precision:** The ratio of correctly predicted positive instances (attacks) to the total number of instances predicted as positive. High precision indicates a low false positive rate.
        - Formula: Precision=TP+FPTP**Error! Filename not specified.**
    - **Recall (Sensitivity):** The ratio of correctly predicted positive instances to the total number of actual positive instances. High recall indicates a low false negative rate.
        - Formula: Recall=TP+FNTP**Error! Filename not specified.**
    - **F1-Score:** The harmonic mean of Precision and Recall. It provides a single score that balances both concerns, which is particularly useful for imbalanced datasets.
        - Formula: F1−Score=2×Precision+RecallPrecision×Recall
          Error! Filename not specified.
- **B. Comparative Analysis**
    - The three models were trained and evaluated, yielding the following performance on the KDDTest+.txt dataset:

Model	Accuracy	Precision	Recall	F1-Score
K-Nearest Neighbors (KNN)	**0.7694**	0.9240	0.6482	0.7619
Linear Discriminant Analysis (LDA)	0.7617	**0.9249**	0.6327	0.7514
Support Vector Machine (SVM)	0.7539	0.9163	0.6248	0.7430

- **Discussion of Results:**

- **Overall Performance:** All three models demonstrated a high level of performance, with accuracies exceeding 75%, confirming the viability of using classical machine learning for NIDS.
- **Best Performing Model:** The **K-Nearest Neighbors (KNN)** model achieved the highest **Accuracy (76.94%)** and the highest **F1-Score (76.19%)**. This suggests that in the high-dimensional feature space of the NSL-KDD dataset, the instance-based classification approach of KNN was the most effective overall.
- **Precision vs. Recall:** The **Linear Discriminant Analysis (LDA)** model achieved the highest **Precision (92.49%)**, meaning it was the best at avoiding false alarms (false positives). However, this came at the cost of a slightly lower Recall, meaning it missed more actual attacks (false negatives) compared to KNN.
- **SVM Performance:** The Support Vector Machine, while a powerful classifier, showed slightly lower performance across all metrics compared to KNN and LDA in this specific experiment. This could be attributed to the choice of a linear kernel or the inherent complexity and non-linearities in the dataset that a linear model might not fully capture.
- **Conclusion on Model Selection:** Based on the balanced performance indicated by the highest F1-Score and the highest overall Accuracy, the **KNN model was selected** as the best-performing model for deployment in the Streamlit dashboard (nids_model.pkl).

VI. CONCLUSION AND FUTURE WORK

A. Conclusion

- This project successfully demonstrated the entire lifecycle of a machine learning project, from data exploration and preprocessing to model training, evaluation, and deployment as a practical web application.
- It was empirically shown that classical machine learning algorithms like KNN, LDA, and SVM can serve as a powerful foundation for building an effective anomaly-based Network Intrusion Detection System.
- Through a rigorous comparative analysis, the K-Nearest Neighbors (KNN) algorithm was identified as the most effective model for the binary classification task on the NSL-KDD dataset, achieving the best balance of performance metrics.
- The development of the Streamlit dashboard provides a tangible proof-of-concept, bridging the gap between theoretical machine learning models and practical, interactive tools for security analysis.

B. Future Work

- **Integration of Deep Learning Models:** The initial project scope included an analysis of deep learning models. Future work should focus on implementing and comparing models like **Multi-Layer Perceptrons (MLP)**, **Long Short-Term Memory (LSTM)** networks, and **Autoencoders** to investigate if they can provide superior performance, especially in detecting complex, time-series-based attacks.
- **Testing on Modern Datasets:** To ensure relevance against contemporary cyber threats, the models should be re-evaluated on more modern and comprehensive

datasets, such as **CIC-IDS2017** or **UNSW-NB15**, which include traffic from modern network protocols and attack vectors.

- **Development of a Real-Time Pipeline:** The current dashboard relies on batch processing of uploaded CSV files. A significant future enhancement would be to develop a real-time detection pipeline that can capture live network traffic using libraries like Scapy or Pyshark, preprocess it on the fly, and make immediate predictions.
- **Multi-Class Classification:** The current system performs binary classification ('Normal' vs. 'Attack'). Future work could expand the model to perform multi-class classification to not only detect an intrusion but also identify the specific type of attack (e.g., DoS, Probe, R2L, U2R), providing more actionable intelligence to security analysts.
- **Hyperparameter Optimization:** A systematic hyperparameter tuning process, using techniques like Grid Search or Randomized Search, could be performed for each model to potentially extract even higher performance.

VII. REFERENCES

[1] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 dataset," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, 2009, pp. 1-8.

[2] S. Revathi and A. Malathi, "A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 12, pp. 279-284, 2013.

[3] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *Proceedings* of the 4th International *Conference on Information Systems Security and Privacy (ICISSP)*, Funchal, Madeira, Portugal, 2018, pp. 108-116.

[4] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, ACT, Australia, 2015, pp. 1-6.

[5] J. H. Park and J. H. Park, "Wireless Intrusion Detection System Using Machine Learning," in *Human-centric Computing and Information Sciences*, vol. 7, no. 1, 2017, Art. no. 43.

[6] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153-1176, Secondquarter 2016.

[7] S. M. H. Adnan, M. N. A. Khan, and F. A. Barbhuiya, "A survey on intrusion detection using machine learning," in *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, Dhaka, Bangladesh, 2017, pp. 448-453.

[8] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, vol. 5, pp. 21954-21961, 2017.

[9] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41565-41587, 2019.

[10] C. Ieracitano, A. Adeel, F. C. Morabito, and A. Hussain, "A novel statistical analysis and autoencoder driven intelligent intrusion detection approach," *Neurocomputing*, vol. 387, pp. 51-62, 2020.

[11] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an effective intrusion detection system using a filter-based feature selection algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986-2998, Oct. 2016.

[12] D. E. Denning, "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222-232, Feb. 1987.

[13] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[14] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Austin, TX, USA, 2010, pp. 51-56.

[15] C. B. C. Team, "Streamlit: The fastest way to build custom ML tools," Streamlit Inc., 2019. [Online]. Available: https://streamlit.io

[16] A. Amri, "joblib: running Python functions as pipeline jobs," 2010. [Online]. Available: https://joblib.readthedocs.io/

[17] M. Sabhnani and G. Serpen, "Why Machine Learning Algorithms Fail in Misuse Detection on KDD Intrusion Detection Data Set," *Intelligent Data Analysis*, vol. 8, no. 4, pp. 403-415, 2004.

[18] S. T. Powers and J. He, "A survey of intrusion detection systems," *Potentials, IEEE*, vol. 22, no. 3, pp. 18-21, 2003.

[19] D. S. Kim and J. S. Park, "Network-based intrusion detection with support vector machines," in *Proceedings of the International Conference on Information Networking*, vol. 2, 2003, pp. 747-756.

[20] A. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks," in *Proceedings of the 2003 Symposium on Applications and the Internet*, Orlando, FL, USA, 2003, pp. 209-216.

[21] K. M. Faraoun and A. Boukelif, "Neural Networks Learning for Computer Security," *Computer Systems Science and Engineering*, vol. 21, no. 5, pp. 303-311, 2006.

[22] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "On the capability of an SOM based intrusion detection system," in *The* 2003 International Joint *Conference on Neural Networks*, vol. 3, 2003, pp. 1808-1813.

[23] Y. Li, "An AdaBoost-based algorithm for network intrusion detection," in *2010 3rd International Conference on Computer Science and Information Technology*, Chengdu, China, 2010, vol. 9, pp. 119-123.

[24] M. Panda and M. R. Patra, "Network Intrusion Detection Using Naive Bayes," *International Journal of Computer Science and Network Security*, vol. 7, no. 12, pp. 258-263, 2007.

[25] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18-28, 2009.

[26] A. A. Ghorbani, W. Lu, and M. Tavallaee, *Network* Intrusion Detection *and Prevention: Concepts and Techniques*. New York, NY, USA: Springer, 2010.

[27] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, 2007.

[28] H. Liao, C. R. Lin, Y. Lin, and K. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16-24, 2013.

[29] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357, 2002.

[30] C. C. Aggarwal, *Data* Mining: The *Textbook*. Cham, Switzerland: Springer, 2015.

[31] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264-323, Sep. 1999.

[32] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, Aug. 2013.

[33] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504-507, 2006.

[34] T. A. L. S. D. Santos, J. D. S. da Silva, A. H. R. Costa, and G. H. G. D. da Costa, "A survey of network-based intrusion detection systems," *Journal of Network and Computer Applications*, vol. 106, pp. 20-36, 2018.

[35] P. Lichman, "UCI Machine Learning Repository," Irvine, CA: University of California, School of Information and Computer Science, 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[36] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[37] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[38] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770-778.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 2672-2680.

[41] J. Brownlee, *Machine Learning Mastery with Python*. San Juan, PR, USA: Machine Learning Mastery, 2016.

[42] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York, NY, USA: Springer, 2009.

[43] S. Raschka and V. Mirjalili, *Python Machine Learning*, 3rd ed. Birmingham, UK: Packt Publishing, 2019.

[44] F. Chollet et al., "Keras," 2015. [Online]. Available: https://keras.io

[45] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015, software available from tensorflow.org.

[46] O. Y. Al-Jarrah, Y. Al-Hammouri, E. A. A. Al-Dujaili, and M. A. A. Al-Shawabka, "An efficient intrusion detection system for computer networks using Gini index and random forest," *Journal of Computer Science*, vol. 11, no. 4, pp. 586-594, 2015.

[47] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[48] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179-188, 1936.

[49] A. P. St-Gelais, J. S. S. M. Wong, and J. M. Fernandez, "A Systematic Literature Review of Machine Learning Methods for Intrusion Detection in Industrial Control Systems," *IEEE Access*, vol. 8, pp. 173038-173059, 2020.

[50] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson, "Threat analysis of IoT networks using artificial neural network," in *2016 International Symposium on Networks