THE DEMOCRATIC PEOPLE'S REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY MOHAMED BOUDIAF - M'SILA

FACULTY: Mathematics and Computer Science

DEPARTEMENT: Computer Science

N° ………...

Discipline: Mathematics and Computer Science
Branch: Computer Science

Specialty: Artificial Intelligence

Dissertation Presented to obtain

Master's Degree Professional

By: Mohammed Badr Eddine Laroussi Graine

SUBJECT

Performance evaluation of machine learning models for intrusion detection system

Publicly supported before the jury composed of:

Pr Hemmak Allaoua	**University of M'sila**	**President**
Pr Lamri Sayad	**University of M'sila**	**Supervised**
Dr Kamel Mohamed	**University of M'sila**	**Examiner**

Promotion: 2024 /2025

THE DEMOCRATIC PEOPLE'S REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY MOHAMED BOUDIAF - M'SILA

FACULTY: Mathematics and Computer Science

DEPARTEMENT: Computer Science

N° ………...

Discipline: Mathematics and Computer Science
Branch: Computer Science

Specialty: Artificial Intelligence

Dissertation Presented to obtain

Master's Degree Professional

By: Mohammed Badr Eddine Laroussi Graine

SUBJECT

Performance evaluation of machine learning models for intrusion detection system

Publicly supported before the jury composed of:

Pr Hemmak Allaoua	**University of M'sila**	**President**
Pr Lamri Sayad	**University of M'sila**	**Supervised**
Dr Kamel Mohamed	**University of M'sila**	**Examiner**

Promotion: 2024 /2025

TABLE OF CONTENT

List of Figures

List of Algorithms

| General introduction |

General introduction

Announcement of the Theme:

In the digital age, cybersecurity is a major concern for individuals, businesses, and governments. The rising frequency and complexity of cyber threats require improved security measures to protect sensitive data and maintain system integrity. Among these measures, Intrusion Detection Systems (IDS) are essential for identifying and reducing unauthorized access and cyber-attacks. However, traditional IDS struggle to keep up with new and changing attack methods. This has led to the investigation of Machine Learning (ML) in cybersecurity as a promising solution.

Formulation of the Subject:

The study delves into how Machine Learning can be utilized in Intrusion Detection Systems, exploring ways in which these ML methods might boost accuracy in detecting threats, cut down on false alarms, and refine our responses to cyber-attacks. This research looks into different Machine Learning methods to see how well they can spot threats in real time.

Personal and Professional Motivations:

From a personal perspective, the rapid evolution of cyber threats and the increasing reliance on digital infrastructures have sparked a deep interest in cybersecurity. Understanding how AI-driven solutions can enhance security aligns with my passion for technology and data-driven decision-making. Professionally, cybersecurity is a critical field with growing demand for intelligent security mechanisms. By exploring the role of Machine Learning in IDS, this research contributes to advancements in cybersecurity and provides valuable insights for future developments in digital protection.

The Problem:

Traditional signature-based IDS face challenges in identifying unknown and zero-day attacks due to their dependence on established rules and attack signatures. Moreover, elevated false positive rates in current IDS lead to inefficiencies in monitoring network security. The difficulty is in creating a smart, precise, and adaptive intrusion detection system that can effectively recognize cyber threats while reducing false positives.

The Hypotheses:

- Machine Learning-based Intrusion Detection Systems (ML-IDS) can achieve higher accuracy in detecting cyber threats compared to traditional signature-based IDS.
- ML-IDS can effectively detect zero-day attacks by identifying anomalies in network traffic, even when attack signatures are unknown.
- The integration of ML techniques can reduce false positive rates, improving the efficiency of cybersecurity monitoring.
- Different ML algorithms (e.g., Decision Trees, Random Forest, SVM, etc.) exhibit varying levels of performance, and some models may be better suited for IDS applications.
- Data preprocessing techniques such as feature selection and class balancing (e.g., SMOTE) significantly impact the accuracy and robustness of ML-based IDS.

Methodology:

To evaluate the effectiveness of Machine Learning in Intrusion Detection Systems, the following methodology is adopted:

1. Data Collection.
2. Data Preprocessing.
3. Machine Learning Models Implemented.
4. Model Evaluation.
5. Analysis & Interpretation.

Manuscript organization:

This thesis is structured into three main chapters:

- **Chapter 1:** Overview of Artificial Intelligence and Machine Learning, covering fundamental concepts and methodologies.
- **Chapter 2:** The role of Machine Learning in cybersecurity, focusing on IDS and their integration with AI techniques.
- **Chapter 3:** Implementation and evaluation of an ML-based IDS, including experimental results and performance analysis.

| Chapter 1 |

Introduction to Artificial Intelligence and Machine Learning

Chapter 1 | Introduction to Artificial Intelligence and Machine Learning

1.1 Introduction:

Artificial Intelligence (AI) and Machine Learning (ML) have revolutionized the way we interact with technology, offering solutions that enhance efficiency, accuracy, and decision-making capabilities across various industries. AI is a broad field of computer science dedicated to building smart machines capable of performing tasks that typically require human intelligence. Within AI, ML stands out as a powerful subset that enables systems to learn from data, recognize patterns, and make decisions with minimal human intervention. As these technologies continue to advance, their applications in cybersecurity, particularly in intrusion detection systems, have garnered significant attention. This chapter provides a foundational understanding of AI and ML, and their key principles.

1.2 Overview of Artificial Intelligence:

Artificial Intelligence (AI) is a branch of computer science that focuses on creating systems capable of performing tasks that typically require human intelligence. These tasks include problem-solving, decision-making, natural language processing, and pattern recognition. AI has evolved significantly since its inception, with applications spanning various domains such as healthcare, finance, cybersecurity, and automation.

The development of AI is primarily driven by advances in computational power, large-scale data availability, and improvements in algorithmic methodologies. AI can be broadly categorized into narrow AI, which specializes in specific tasks, and general AI, which aims to replicate human cognitive abilities across multiple domains. The field of AI encompasses various subfields, including expert systems, robotics, computer vision, and, most notably, machine learning.

1.3 Machine Learning:

Machine Learning is a scientific field, and more specifically a subcategory of artificial intelligence. It consists of letting algorithms discover “patterns”, i.e. recurring patterns, in data sets. This data can be numbers, words, images, statistics…etc.

Anything that can be stored digitally can serve as data for Machine Learning. By detecting patterns in this data, algorithms learn and improve their performance in performing a specific task. [1]

1.4 How Does Machine Learning Work?

Machine learning is a subset of artificial intelligence that enables systems to learn from data and make predictions or decisions without being explicitly programmed. It follows a structured process that transforms raw data into meaningful insights.

The process is as follows:

1.4.1 data collection: where data is gathered from various sources such as databases, files, images, and web scraping. The quality and quantity of data significantly impact model performance.

1.4.2 data preprocessing: is performed to clean, normalize, and handle missing data, ensuring the dataset is ready for analysis.

1.4.3 choosing the right model: selecting an appropriate machine learning algorithm such as linear regression, decision trees, or neural networks based on data complexity and problem requirements.

1.4.4 Training: training the model by feeding it the processed data, allowing it to adjust parameters and identify patterns while avoiding overfitting or underfitting.

1.4.5 Evaluation: where its performance is tested using metrics like accuracy, precision, and recall to ensure reliability.

1.4.6 If necessary, hyperparameter tuning is applied using techniques like grid search and cross-validation to optimize the model further.

1.4.7 prediction and deployment phase: the trained model is integrated into real-world applications, where it continuously processes new input data, refines its predictions, and improves over time. If the model's predictions do not meet expectations, it is retrained multiple times to enhance accuracy and effectiveness.

Through this iterative learning process, machine learning algorithms adapt and improve, enabling applications in various fields such as finance, healthcare, cybersecurity, and marketing.

1.5 Types of Real-World Data and Machine Learning Techniques:

Machine learning algorithms typically consume and process data to learn the related patterns about individuals, business processes, transactions, events, and so on. In the following, we discuss various types of real-world data as well as categories of machine learning algorithms.

1.5.1 Types of Real-World Data: Usually, the availability of data is considered as the key to construct a machine learning model or data-driven real-world systems. [2] Data can be of various forms, such as structured, semi-structured, or unstructured. Besides, the "metadata" is another type that typically represents data about the data. In the following, we briefly discuss these types of data.

- *Structured:* It has a well-defined structure, adheres to a data model that follows a regular order, is well-organised and easily accessible, and is used by an entity or a computer programme. Structured data is often stored in a tabular manner in well-defined schemes such as relational databases. Structured data includes things like names, dates, addresses, credit card numbers, stock information, geolocation, and so on.

- *Unstructured:* Unstructured data, on the other hand, has no pre-defined format or organization, making it far more difficult to acquire, handle, and analyse, as it primarily consists of text and multimedia information. Unstructured data includes sensor data, emails, blog entries, wikis, word processing documents, PDF files, audio files, videos, photos, presentations, web pages, and many more business documents.

- *Semi-structured:* Semi-structured data, unlike structured data, is not kept in a relational database but contains organisational qualities that make it easier to analyse. Semi-structured data includes HTML, XML, JSON documents, NoSQL databases, and so on.

  Metadata: It is not ordinary data, but "data about data." The major distinction between "data" and "metadata" is that data are merely materials that can be used to classify, measure, or even document

something in relation to an organization's data attributes. Metadata, on the other hand, explains the pertinent data information, making it more meaningful to data users. The author, file size, date generated by the document, keywords that define the document, and so on are all examples of metadata.

1.5.2 Types of Machine Learning Techniques:

As shown in Figure 1, machine learning algorithms are categorized into four types which are supervised learning, unsupervised learning, and reinforcement learning.

Additionally, there is a more specific category called semi-supervised learning, which combines elements of both supervised and unsupervised learning [3].

The following section provides a brief overview of each learning approach and its applicability in addressing real-world problems.

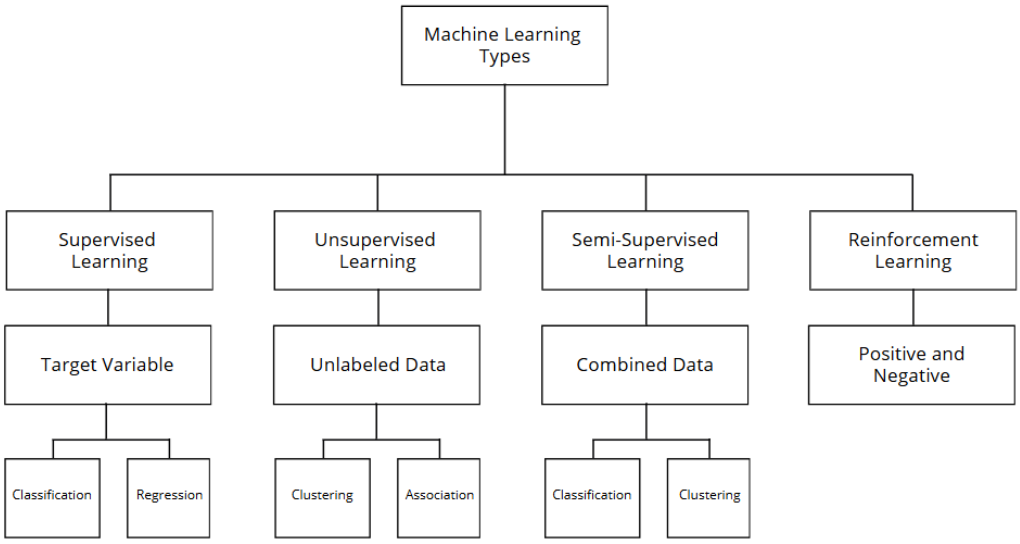


Figure 1.1 - Various types of machine learning techniques

- *Supervised:* Supervised learning is a method of training a model to map inputs to outputs using labeled data. It relies on sample input-output pairs to learn a function that generalizes to new data. This approach is goal-oriented, meaning it aims to achieve specific outcomes based on given inputs. The two primary tasks in supervised learning are "classification", which involves categorizing data, and "regression", which focuses on

predicting continuous values. For instance, determining the sentiment of a tweet or classifying a product review falls under supervised learning

- Advantages of Supervised Machine Learning:
  - Supervised learning models tend to achieve high accuracy due to their training on labeled data.
  - Their decision-making process is usually transparent and easy to interpret.
  - They can be integrated into pre-trained models, reducing the time and resources needed for developing new models from scratch.
- Disadvantages of Supervised Machine Learning:
  - Supervised learning may struggle with recognizing patterns that were not present in the training data.
  - It can be time-consuming and expensive due to its reliance on labeled data.
  - The model might generalize poorly when faced with new or unseen data.
- Applications of Supervised Learning: Supervised learning is used in a wide variety of applications, including: Image classification - Speech recognition - Recommendation systems - Predictive analytics - Medical diagnosis - Email spam detection - Quality control in manufacturing - Customer support – Gaming - Weather forecasting …etc.

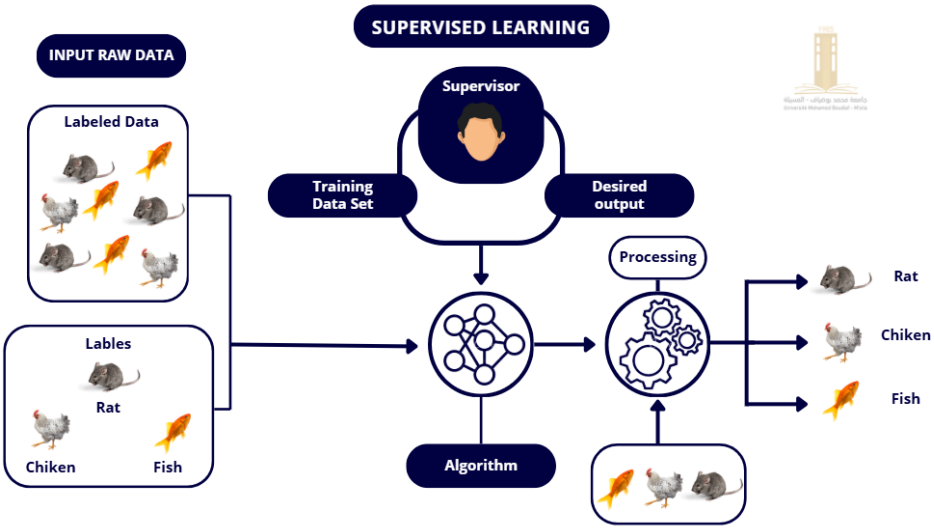


Figure1.2 - Supervised Learning Illustration

- *Unsupervised:* Unsupervised learning is a data-driven approach that analyzes unlabeled datasets without human intervention. It is often used to identify patterns, uncover structures, and extract meaningful features from data. Common applications include clustering, density estimation, feature learning, dimensionality reduction, association rule mining, and anomaly detection.
    - Advantages of Unsupervised Machine Learning:
        - Unsupervised learning uncovers hidden patterns and relationships within the data.
        - It is commonly used for tasks like customer segmentation, anomaly detection, and data exploration.
        - Since it does not require labeled data, it minimizes the effort needed for data labeling.
    - Disadvantages of Unsupervised Machine Learning:
        - Without labeled data, assessing the quality of the model's output can be challenging.
        - Cluster interpretability may be unclear, making it difficult to derive meaningful insights.
        - Techniques like autoencoders and dimensionality reduction help extract useful features from raw data.

- Applications of Unsupervised Learning: Here are some common applications of unsupervised learning: Clustering - Recommendation systems - Density estimation - Image and video compression - Data preprocessing - Genomic data analysis - Image segmentation - Community detection in social networks - Customer behavior analysis - Content recommendation …etc.

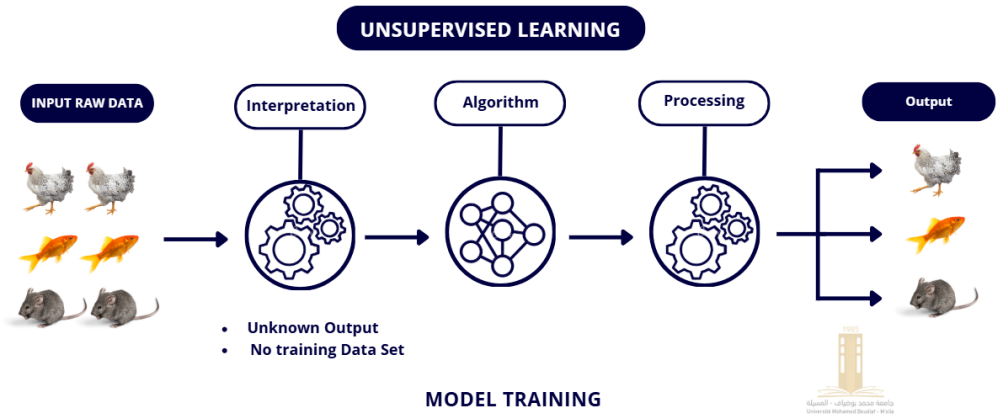


Figure 1.3 - Unsupervised Learning Illustration

- *Semi-supervised:* Semi-supervised learning blends supervised and unsupervised techniques by utilizing both labeled and unlabeled data. It acts as a middle ground between fully supervised and unsupervised learning. In real-world scenarios, labeled data is often limited, while unlabeled data is abundant, making this approach highly useful. The primary objective is to achieve better predictive performance than a model trained solely on labeled data. Common applications include machine translation, fraud detection, data labeling, and text categorization.
  - Advantages of Semi- Supervised Machine Learning:
    - Semi-supervised learning enhances generalization by leveraging both labeled and unlabeled data.
    - It is adaptable and can be applied to various types of data.
  - Disadvantages of Semi- Supervised Machine Learning:

- Semi-supervised learning can be more complex to implement than other approaches.
- It still depends on labeled data, which may not always be readily available.
- The quality of unlabeled data can influence the model's performance.

- Applications of Semi-Supervised Learning: Here are some common applications of semi-supervised learning: Image Classification and Object Recognition - Healthcare and Medical Imaging …etc.

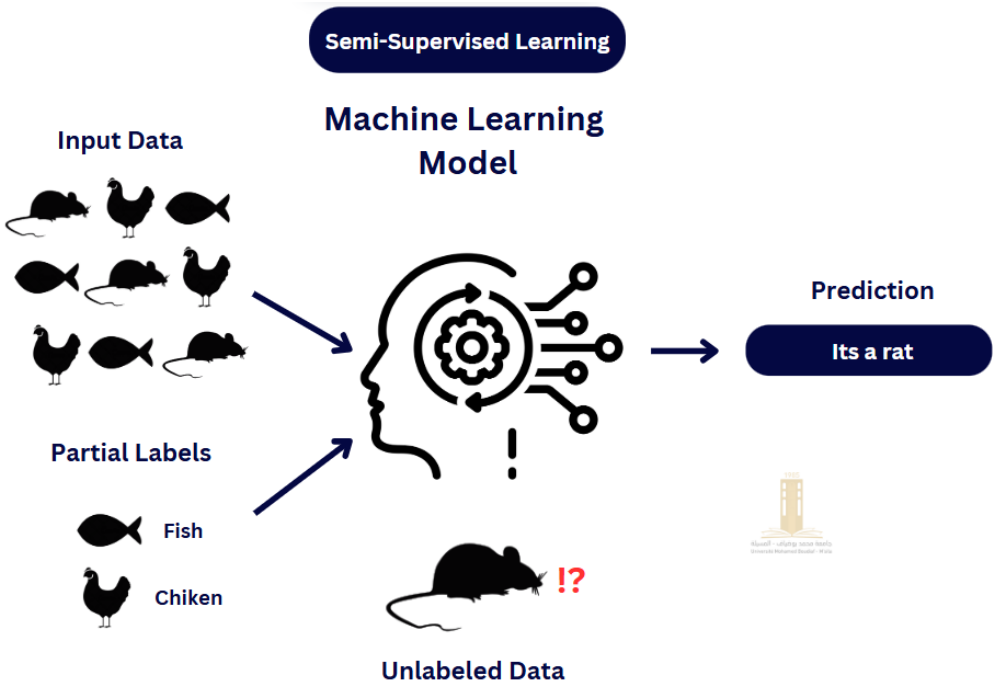


Figure 1.4 - Semi-Supervised Learning Illustration

- *Reinforcement:* Reinforcement learning is a machine learning approach that enables software agents and machines to determine optimal behavior within a given environment to maximize efficiency. It follows an environment-driven strategy, where learning is guided by rewards and penalties. The goal is to take actions that increase rewards while minimizing risks. This technique is particularly useful for training AI models to automate or optimize complex systems, such as robotics,

autonomous driving, manufacturing, and supply chain logistics. However, it is not well-suited for solving simple or straightforward problems.

- Types of Reinforcement Machine Learning: There are two main types of reinforcement learning:
  - Positive reinforcement:
    - Reinforcement learning rewards the agent for performing desired actions.
    - It encourages the agent to repeat beneficial behaviors.
    - Example: include awarding points in a game for correct answers.
  - Negative reinforcement:
    - Reinforcement learning can remove an undesirable stimulus to reinforce a desired behavior.
    - It discourages the agent from repeating unfavorable actions.
    - Example: include avoiding a penalty by completing a task.

- Advantages of Reinforcement Machine Learning:
  - Reinforcement learning enables autonomous decision-making, making it ideal for tasks requiring sequential decision-making, such as robotics and game-playing.
  - It is preferred for achieving long-term goals that are difficult to attain through other methods.
  - This approach is useful for solving complex problems that conventional techniques cannot effectively address.

- Disadvantages of Reinforcement Machine Learning:
  - Training reinforcement learning agents requires significant computational resources and time.
  - It is not well-suited for solving simple problems.
  - The approach demands large amounts of data and computation, making it costly and sometimes impractical.

- Applications of Reinforcement Machine Learning: Here are some applications of reinforcement learning: Game Playing - Finance and Trading - Game AI - Adaptive Personal Assistants - Virtual Reality (VR) and Augmented Reality (AR) - Industrial Control – Education – Agriculture …etc

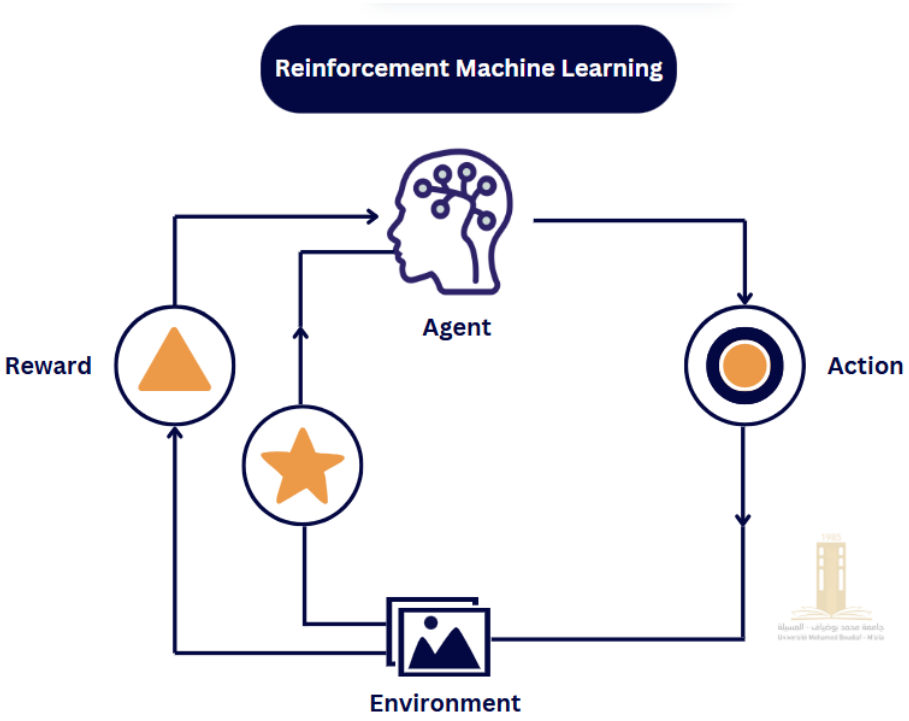


Figure 1.5 - Reinforcement Machine Learning Illustration

1.6 Machine Learning Tasks and Algorithms:

Machine learning comprises of various methods and algorithms including but not limited to Classification, Regression, Data Clustering, Association Rule Learning, and Reinforcement Learning. (see Figure 1.6 for the general framework of a machine learning predictive model, covering both training and testing phases). [4]

Machine learning techniques are widely used for data-driven decision-making across various domains. Specifically, classification analysis categorizes data into predefined classes, regression analysis predicts continuous values based on input variables, and cluster analysis identifies hidden patterns by grouping similar data points. These techniques enable effective pattern recognition and predictive modeling. Additionally, a machine learning-based predictive model typically operates in two phases: training, where the model learns from historical data, and testing, where it generates predictions for new data.

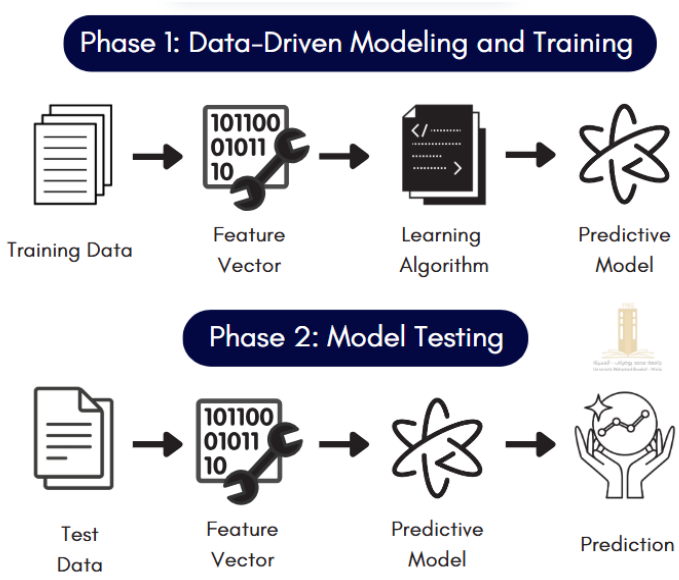


Figure 1.6 - Phase 1 and Phase 2 of machine learning based predictive model

1.6.1 Classification Analysis:

Classification, the task of sorting input data into categories ("classes"), is a supervised learning technique in machine learning that predicts class labels for given examples (where the correct answers/labels are known). In other words, classification uses known data (both its inputs and corresponding correct outputs) to learn how to categorize new, previously unseen data into one of the known categories or classes (Sarker 2021a). For example, a supervised classification task might involve learning to categorize emails as "spam" or "not spam" based on a training dataset of emails for which the correct categorization is already known. [5]

There are three common classification problems:

- *Binary Classification:* This method categorizes data into two distinct classes, such as "true" or "false." One class represents a

normal state, while the other represents an abnormal state. For example, in advertising, it can determine if an ad for a mature-themed video game should be labeled as "General Audience" or "Age Restricted."

- *Multiclass Classification:* Unlike binary classification, this method involves more than two class labels. Instead of distinguishing between normal and abnormal states, it categorizes data into multiple groups, assuming each instance belongs to only one category. For example, in advertising, a car can be classified as "Luxury," "Sport," "SUV," or "Economy" based on its features and target audience.
- *Multi-label Classification:* This classification method differs from binary and multiclass classification by allowing multiple labels to be assigned to a single instance. It enables hierarchical classification, meaning an example can belong to multiple categories at different levels. For example, in online advertising, a smartphone ad could be tagged with multiple labels such as "Technology," "Photography," "Gaming," and "Outdoor Activities," allowing it to appear in various relevant categories.

A few dominant classification algorithms:

- *Naïve Bayes (NB):* Naïve Bayes is a classification algorithm based on Bayes' theorem, a key principle in probability and statistics that calculates the likelihood of an event based on prior knowledge of related conditions. This algorithm assumes feature independence, making it simple yet powerful for tasks such as text classification and spam detection. For example, it can distinguish between a legitimate email and a spam message about a sale. Naïve Bayes is particularly effective in binary and multiclass classification, especially when working with noisy data and limited training. However, its strong assumption of independence between features may limit its accuracy in some scenarios.

- *Linear Discriminant Analysis (LDA)**:*** Linear Discriminant Analysis (LDA) is a classification technique that applies Bayes' rule to create linear decision boundaries by analyzing class distributions. In simple terms, it finds the best straight-line (or plane) separation between different categories of data. LDA reduces dataset dimensionality and model complexity by evaluating how data points are distributed within each class and updating probabilities accordingly. In advertising, for instance, LDA can help analyze users' online behaviors and characteristics to predict and target those most likely to engage with specific ads, improving marketing efficiency.
- *Logistic Regression (LR):* Logistic Regression (LR) is a widely used probabilistic statistical model in machine learning, primarily for binary classification tasks. Unlike linear regression, which predicts continuous values, logistic regression estimates the probability that a given input belongs to a particular class. It achieves this by applying a logistic (sigmoid) function to the linear combination of input features. The sigmoid function, mathematically defined as:

$$g(z) = \frac{1}{1 + \exp(-z)}$$

converts any real-valued number zzz into a probability ranging between **0 and 1**. Here, "z" represents the weighted sum of input features and their corresponding coefficients. The sigmoid function ensures that the model outputs values interpretable as probabilities, which are then used to classify instances into two categories, typically by setting a decision threshold (e.g., if **g(z) ≥ 0.5** classify as class 1. otherwise, classify as class 0).

Despite its effectiveness, logistic regression has limitations. One major drawback is its assumption of a linear relationship between independent variables and the log-odds of the dependent

variable. This assumption may not always hold in complex datasets. Additionally, logistic regression is prone to overfitting, especially in high-dimensional datasets. To address this, regularization techniques such as L1 (Lasso) and L2 (Ridge) regularization add penalties to the model's complexity, preventing it from fitting too closely to noise in the data.

Logistic regression is widely used in various applications, including spam detection, medical diagnosis, fraud detection, and online advertising. For example, in digital marketing, logistic regression can predict whether a user will click on an ad (1) or not (0) based on features such as age, browsing history, and time of day. Its simplicity, interpretability, and efficiency make it a fundamental classification technique in machine learning.

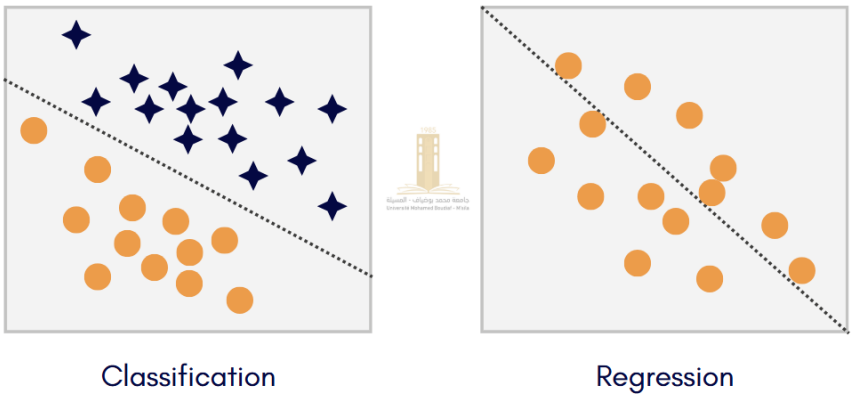


Figure 1.7 - Classification vs. Regression: Decision Boundaries and Relationships

In classification the dotted line represents a linear boundary that separates the two classes. in regression, the dotted line models the linear relationship between the two variables.

- *K-Nearest Neighbors (KNN):* it is a "lazy learning" algorithm that classifies new data points based on their similarity to their k nearest neighbors. It does not build a general model but stores all training data, making classification decisions based on majority votes. KNN is effective for classification and regression tasks. In advertising, it can recommend products to users based on the preferences of similar customers.

- *Support Vector Machine (SVM):* it is a machine learning model that creates a decision boundary (hyperplane) to separate data classes while maximizing the margin between them for better accuracy. It ensures more precise classification by maintaining the widest possible separation between groups. In advertising, SVM can analyze customer behavior, such as predicting whether users will like or dislike an ad, by determining the best boundary between different reactions.

- *Decision Trees (DT):* are non-parametric methods used for both classification and regression. They operate by breaking down a dataset into smaller subsets through a series of decision rules, forming a tree-like structure. Each decision node represents a choice based on an attribute, and branches lead to possible outcomes, ultimately reaching leaf nodes that provide final classifications or predictions. In machine learning, decision trees are trained using input data, and classification occurs by following paths from the root to the leaves based on attribute values and splitting criteria.

  A key aspect of decision tree construction is determining the **best attribute to split on**, which is done using measures like **Entropy** and **Gini Index**. **Entropy**, defined as:

$$H(x) = -\sum_{i=1}^{n} p(x_i).log_2.p(x_i)$$

  quantifies the impurity in a dataset, with lower values indicating more homogeneity. **Gini Index**, given by:

$$Gini(E) = 1 - \sum_{i=1}^{c} {p_i}^2$$

measures the probability of incorrect classification by randomly selecting a label from the dataset. The lower the Gini Index, the better the split. These measures guide the decision tree in choosing attributes that maximize information gain, ensuring efficient classification.

In advertising, decision trees help businesses target customers effectively. For instance, an agency placing a luxury watch ad might use a decision tree to assess the target audience's platform preferences (social media, TV, or magazines) and refine ad placement based on age, browsing habits, or TV watching time. By leveraging entropy and Gini Index, decision trees optimize the selection process, ensuring ads reach the most relevant consumers.

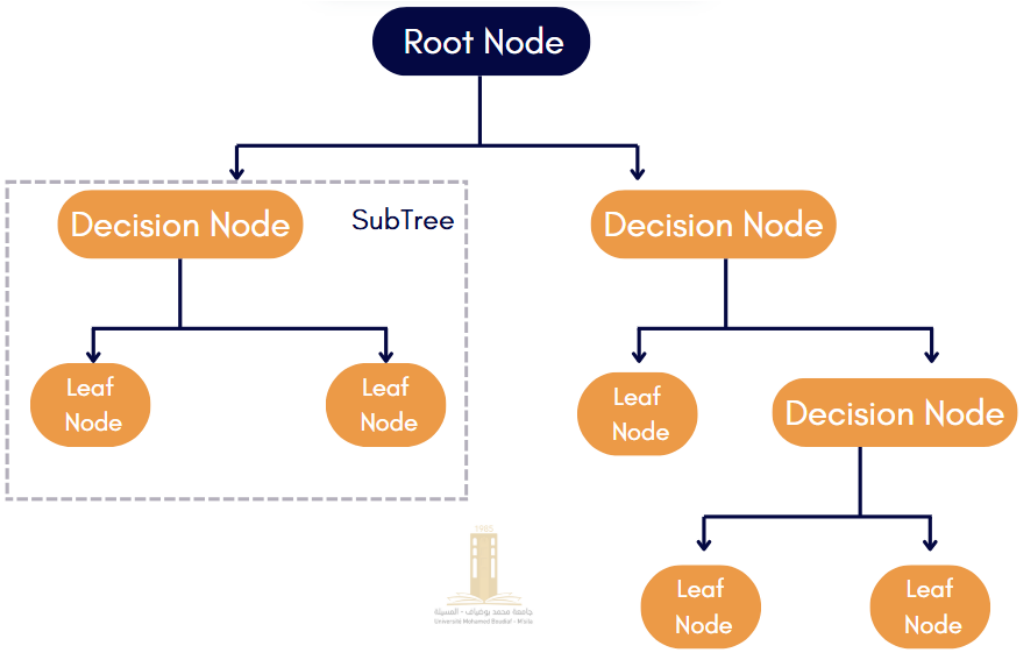


Figure 1.8 - An example of a decision tree structure

- *Random Forest (RF):* An ensemble method that constructs multiple decision trees in parallel and aggregates their predictions using majority voting (for classification) or averaging (for regression). It reduces overfitting and enhances accuracy compared to single decision trees. Random forests use bootstrap aggregation (bagging) and random feature selection to improve robustness.

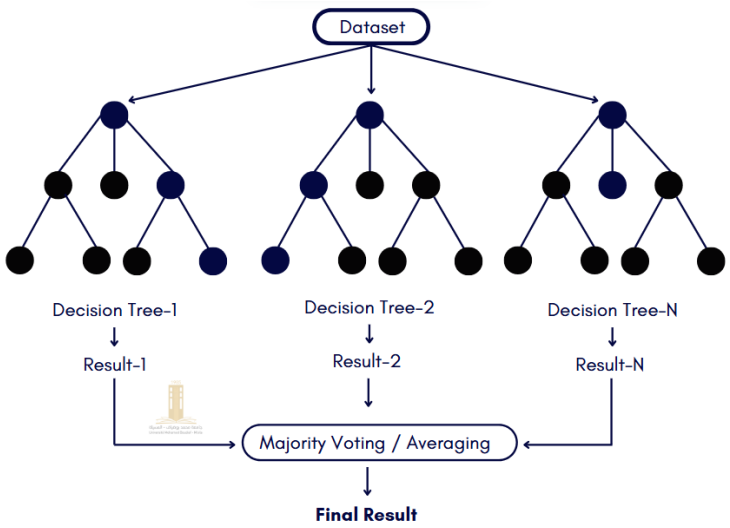


Figure 1.9 - random forest structure considering multiple decision trees

- *Adaptive Boosting (AdaBoost):* A sequential ensemble learning technique that strengthens weak classifiers by iteratively adjusting their weights based on errors. Unlike random forests, which build trees in parallel, AdaBoost trains classifiers sequentially to minimize errors. It is effective for improving decision trees but can be sensitive to noisy data and outliers.

1.6.2 Regression Analysis:

Regression analysis is a fundamental machine learning technique used to predict continuous outcomes based on one or more predictor variables. Unlike classification, which assigns discrete class labels, regression focuses on modeling numerical relationships. It is widely applied in financial forecasting, trend analysis, marketing, cost estimation, and time series prediction. By feeding an algorithm historical data where the outcome is known, the algorithm learns the relationship between the independent and dependent variables and can then make predictions on new, unseen data. Regression analysis also plays a crucial role in advertising, allowing businesses to understand consumer behavior, optimize marketing campaigns, predict sales trends, and fine-tune advertising strategies.

One of the most popular techniques is:

1.6.2.1 *Simple and Multiple Linear Regression:* where the dependent variable is continuous, and the independent variables can be continuous or discrete. A linear regression model establishes a relationship between

the dependent variable "Y" and one or more independent variables "X" using the best-fit straight line, as shown in the following equations:

$$y = a + bx + e \quad (1)$$

$$y = b_1x_1 + b_2x_2 + \cdots + b_nx_n + e \quad (2)$$

Here, "a" represents the intercept, "b" represents the slope, and "e" is the error term. Simple linear regression uses one independent variable, while multiple linear regression extends this by incorporating multiple independent variables.

1.6.2.2 *Polynomial Regression:* is another variation of regression that models non-linear relationships by introducing polynomial terms into the equation. It takes the form:

$$y = b_0 + b_1x + b_2x^2 + b_2x^3 + \cdots + b_nx^n + e \quad (3)$$

This is useful when the data exhibits a non-linear trend that cannot be captured by simple linear regression.

To improve model accuracy and prevent overfitting, LASSO & Ridge Regression techniques are often applied.

- LASSO Regression (L1 Regularization): Shrinks coefficients and sets some to zero, effectively selecting only the most relevant predictors.
- Ridge Regression (L2 Regularization): Reduces the magnitude of coefficients without eliminating any, making it useful in cases of multicollinearity.

These regression techniques are essential in data science and machine learning, providing accurate predictions, optimizing marketing strategies, and making informed business decisions.

1.6.3 Cluster Analysis:

Cluster analysis is an unsupervised machine learning technique that groups similar data points into clusters without a specific predefined outcome. It is widely used to uncover hidden patterns and relationships within large datasets by organizing objects

into clusters, where members of the same group share greater similarity compared to those in other clusters. This method is commonly applied in e-commerce, health analytics, cybersecurity, mobile data processing, and user behavior analysis to detect trends, segment customers, and enhance decision-making.

There are several clustering techniques, each suited for different types of data and structures:

- *Partitioning Methods:* (e.g., K-Means, K-Medoids): Divide data into a fixed number of clusters based on distance metrics.
- *Hierarchical Clustering:* Constructs a tree-like structure (dendrogram) to represent nested clustering relationships.
- *Density-Based Clustering:* (e.g., DBSCAN, OPTICS): Identifies clusters by detecting high-density regions, making it useful for anomaly detection.
- *Model-Based Clustering:* (e.g., Gaussian Mixture Models - GMMs): Assumes that data is generated from multiple probability distributions.

By selecting the appropriate clustering technique, organizations can efficiently analyze large datasets, extract meaningful insights, and make data-driven decisions across various domains.

1.7 Some Applications of Machine Learning:

Today, companies are using Machine Learning to improve business decisions, increase productivity, detect disease, forecast weather, and do many more things. With the exponential growth of technology, we not only need better tools to understand the data we currently have, but we also need to prepare ourselves for the data we will have. To achieve this goal, we need to build intelligent machines. We can write a program to do simple things. But most of the time, Hardwiring Intelligence in it is difficult. The best way to do it is to have some way for machines to learn things themselves. A mechanism for learning – if a machine can learn from input, then it does the hard work for us. This is where Machine Learning comes into action. Some of the most common examples are [6]:

- *Image Recognition:* ML algorithms can identify and classify objects within images, facilitating advancements in areas such as facial recognition, medical imaging, and automated tagging in photo management applications.

- *Speech Recognition:* Systems like Siri and Alexa utilize ML to interpret and process human speech, enabling voice-activated commands and virtual assistants that enhance user interaction.
- *Recommender Systems:* By analyzing user behavior and preferences, ML-driven recommender systems suggest products, movies, or music, as seen on platforms like Netflix and Amazon, thereby personalizing user experiences.
- *Fraud Detection:* Financial institutions employ ML models to detect unusual transaction patterns, helping to identify and prevent fraudulent activities in real-time.
- *Self-Driving Cars:* Autonomous vehicles rely on ML to interpret sensory data, allowing them to navigate roads, recognize obstacles, and make driving decisions without human intervention.
- *Medical Diagnosis:* ML assists in diagnosing diseases by analyzing medical data and images, leading to early detection and personalized treatment plans for patients.
- *Stock Market Trading:* ML algorithms analyze market trends and historical data to forecast stock prices, aiding investors in making informed trading decisions.

1.8 Conclusion:

Artificial Intelligence (AI) and Machine Learning (ML) have revolutionized various industries by enabling intelligent decision-making, pattern recognition, and data-driven insights. Their impact spans multiple domains, including healthcare, finance, and cybersecurity. However, alongside these advancements, challenges such as ethical concerns, data privacy, and model interpretability must be carefully addressed to ensure responsible deployment. As AI and ML continue to evolve, their role in enhancing cybersecurity becomes increasingly significant. This chapter has established the fundamental concepts of AI and ML, setting the stage for an in-depth exploration of their role in cybersecurity, particularly in the development of Intrusion Detection Systems (IDS), in the following chapter.

| Chapter 2 |

Intrusion Detection System (IDS)

Chapter 2 | Intrusion Detection System (IDS)

2.1 Introduction:

Intrusion Detection Systems (IDS) play a crucial role in cybersecurity as cyber threats become more sophisticated. Traditional security measures often fail to keep up with evolving attack techniques, making AI-driven solutions essential for detecting and preventing cyberattacks. Machine Learning (ML), a subset of AI, enhances IDS by analyzing vast datasets, identifying attack patterns, and predicting potential threats before they cause harm.

This chapter examines the integration of AI in Intrusion Detection Systems (IDS), highlighting its advantages, challenges, and applications. AI-powered IDS, particularly those using Machine Learning (ML), enhance proactive threat detection, automate security processes, and improve the resilience of digital infrastructures. Additionally, we mention existing research and related works in this domain.

2.2 What Is an Network Intrusion in Cybersecurity?

A network intrusion refers to any unauthorized activity within a digital network. These intrusions often involve the theft of critical network resources, compromising data security and system integrity. They may manifest as serious threats like ransomware attacks or unintentional data leaks caused by employees or other network users.

Intrusions can be categorized as passive (where unauthorized access occurs quietly without detection) or active (where intruders alter network resources). They may originate externally—through cyberattacks on web servers and firewalls—or internally, from employees, customers, or business partners.

Some intruders aim only to deface websites with messages or offensive graphics, while others seek to exfiltrate sensitive data. In more sophisticated attacks, cybercriminals may implant malicious code or establish a persistent presence to continuously siphon data.

Attackers can gain access in multiple ways:

2.2.1 Spoofing and Identity Manipulation: Attackers manipulate source IP or MAC addresses to disguise their identity and evade detection. This method is often used in:

- Physical Access Attacks: Gaining direct control over restricted hardware allows attackers to alter IP/MAC addresses, making their activities harder to trace.
- External Attacks: Cybercriminals use address spoofing to launch phishing attacks, impersonate legitimate users, or evade security measures in network-based intrusions.

2.2.2 Packet Manipulation and Fragmentation: By breaking malicious data into smaller fragments, attackers can bypass detection mechanisms that inspect packet headers instead of reassembling full payloads. This technique is used in:

- External Attacks: Fragmented packets help attackers bypass firewalls and IDS when breaching networks or injecting malware into systems.
- Hybrid Threats: Attackers may use fragmentation in combination with insider threats to introduce malicious payloads without triggering standard IDS rules.

2.2.3 Adaptive and Evasive Attack Patterns: Hackers adjust their attack architecture to avoid known IDS detection rules. This includes modifying payload structures, encoding malicious commands, and using polymorphic malware. These tactics are common in:

- External Attacks: Cybercriminals obfuscate attack patterns in SQL injection, cross-site scripting (XSS), and malware infections to bypass signature-based detection.
- Internal Threats: Insiders may alter system logs, disable security software, or gradually exfiltrate data to avoid raising alerts.

2.2.4 Coordinated and Distributed Attacks: In a coordinated attack, multiple hosts, attackers, or botnets are used to overwhelm security defenses. IDS struggles to correlate scattered attack sources, making it difficult to detect patterns. This approach is prevalent in:

- Hybrid Threats: Attackers combine internal and external resources to conduct large-scale breaches, such as Advanced Persistent Threats (APTs) and Distributed Denial-of-Service (DDoS) attacks.
- External Attacks: Network scans and brute-force attempts are distributed among multiple compromised systems, making it difficult for IDS to identify a single attack source.

2.3 What Is an Intrusion Detection System (IDS)?

An intrusion detection system (IDS) is an application that monitors network traffic and searches for known threats and suspicious or malicious activity. The IDS sends alerts to IT and security teams when it detects any security risks and threats. [7]

Most IDS solutions simply monitor and report suspicious activity and traffic when they detect an anomaly. However, some can go a step further by taking action when it detects anomalous activity, such as blocking malicious or suspicious traffic.

IDS tools typically are software applications that run on organizations' hardware or as a network security solution. There are also cloud-based IDS solutions that protect organizations' data, resources, and systems in their cloud deployments and environments.

2.4 Types Of Intrusion Detection Systems (IDS): IDS solutions come in a range of different types and varying capabilities. Common types of intrusion detection systems (IDS) include:

2.4.1 Network intrusion detection system (NIDS): A NIDS solution is deployed at strategic points within an organization's network to monitor incoming and outgoing traffic. This IDS approach monitors and detects malicious and suspicious traffic coming to and going from all devices connected to the network.

2.4.2 Host intrusion detection system (HIDS): A HIDS system is installed on individual devices that are connected to the internet and an organization's internal network. This solution can detect packets that come from inside the business and additional malicious traffic that a NIDS solution cannot. It can also discover malicious threats coming from the host, such as a host being infected with malware attempting to spread it across the organization's system.

2.4.3 Signature-based intrusion detection system (SIDS): A SIDS solution monitors all packets on an organization's network and compares them with attack signatures on a database of known threats.

2.4.4 Anomaly-based intrusion detection system (AIDS): This solution monitors traffic on a network and compares it with a predefined baseline that is considered "normal." It detects anomalous activity and behavior across the network, including bandwidth, devices, ports, and protocols. An AIDS solution uses machine-learning techniques to build a baseline of normal behavior and establish a corresponding security policy. This ensures businesses can discover new, evolving threats that solutions like SIDS cannot.

2.4.5 Perimeter intrusion detection system (PIDS): A PIDS solution is placed on a network to detect intrusion attempts taking place on the perimeter of organizations' critical infrastructures.

2.4.6 Virtual machine-based intrusion detection system (VMIDS): A VMIDS solution detects intrusions by monitoring virtual machines. It enables organizations to monitor traffic across all the devices and systems that their devices are connected to.

2.4.7 Stack-based intrusion detection system (SBIDS): SBIDS is integrated into an organization's Transmission Control Protocol/Internet Protocol (TCP/IP), which is used as a communications protocol on private networks. This approach enables the IDS to watch packets as they move through the organization's network and pulls malicious packets before applications or the operating system can process them.

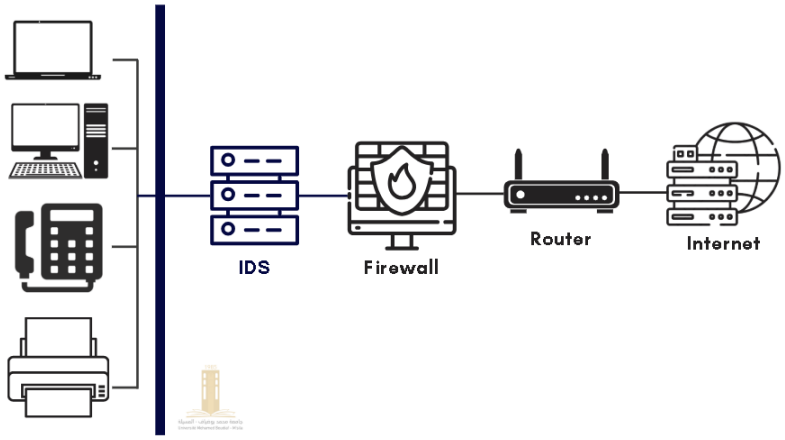


Figure 2.1 - Intrusion detection system illustration

2.5 Intrusion Detection System Evasion Techniques:

- Fragmentation: Dividing the packet into smaller packet called fragment and the process is known as fragmentation. This makes it impossible to identify an intrusion because there can't be a malware signature.
- Packet Encoding: Encoding packets using methods like Base64 or hexadecimal can hide malicious content from signature-based IDS.
- Traffic Obfuscation: By making message more complicated to interpret, obfuscation can be utilised to hide an attack and avoid detection.
- Encryption: Several security features, such as data integrity, confidentiality, and data privacy, are provided by encryption. Unfortunately, security features are used by malware developers to hide attacks and avoid detection. [8]

2.6 Detection Methods of IDS:

2.6.1 Signature-Based Method: Signature-based IDS detects the attacks on the basis of the specific patterns such as the number of bytes or a number of 1s or the number of 0s in the network traffic. It also detects on the basis of the already known malicious instruction sequence that is used by the malware. The detected patterns in the IDS are known as signatures. Signature-based IDS can easily detect the attacks whose pattern (signature) already exists in the system but it is quite difficult to detect new malware attacks as their pattern (signature) is not known.

2.6.2 Anomaly-Based Method: Anomaly-based IDS was introduced to detect unknown malware attacks as new malware is developed rapidly. In anomaly-based IDS there is the use of machine learning to create a trustful activity model and anything coming is compared with that model and it is declared suspicious if it is not found in the model. The machine learning-based method has a better-generalized property in comparison to signature-based IDS as these models can be trained according to the applications and hardware configurations [9].

2.7 Why Are Intrusion Detection Systems (IDS) Important?

An Intrusion Detection System (IDS) adds extra protection to your cybersecurity setup, making it very important. It works with your other security tools to catch threats

that get past your main defenses. So, if your main sdcystem misses something, the IDS will alert you to the threat.

2.8 How Does an Intrusion Detection System Work? What Are Its Uses?

IDS solutions excel in monitoring network traffic and detecting anomalous activity. They are placed at strategic locations across a network or on devices themselves to analyze network traffic and recognize signs of a potential attack.

An IDS works by looking for the signature of known attack types or detecting activity that deviates from a prescribed normal. It then alerts or reports these anomalies and potentially malicious actions to administrators so they can be examined at the application and protocol layers.

This enables organizations to detect the potential signs of an attack beginning or being carried out by an attacker. IDS solutions do this through several capabilities, including:

- Monitoring the performance of key firewalls, files, routers, and servers to detect, prevent, and recover from cyberattacks.
- Enabling system administrators to organize and understand their relevant operating system audit trails and logs that are often difficult to manage and track.
- Providing an easy-to-use interface that allows staff who are not security experts to help with the management of an organization's systems.
- Providing an extensive database of attack signatures that can be used to match and detect known threats.
- Providing a quick and effective reporting system when anomalous or malicious activity occurs, which enables the threat to be passed up the stack
- Generating alarms that notify the necessary individuals, such as system administrators and security teams, when a breach occurs.
- In some cases, reacting to potentially malicious actors by blocking them and their access to the server or network to prevent them from carrying out any further action.

The increasingly connected nature of business environments and infrastructures means they demand highly secure systems and techniques to establish trusted lines of communication. IDS has an important role within modern cybersecurity strategies

to safeguard organizations from hackers attempting to gain unauthorized access to networks and stealing corporate data. [9]

2.9 What Is the Difference Between a Firewall And IDS?

Firewalls and intrusion detection systems (IDS) are cybersecurity tools that can both safeguard a network or endpoint. Their objectives, however, are very different from one another.

2.9.1 IDS: Intrusion detection systems are passive monitoring tools that identify possible threats and send out notifications to analysts in security operations centers (SOCs). In this way, incident responders can promptly look into and address the potential event.

2.9.2 Firewall: A firewall, on the other hand, analyzes the metadata contained in network packets and decides whether to allow or prohibit traffic into or out of the network based on pre-established rules. A firewall essentially creates a barrier that stops certain traffic from crossing through it.

An IDS is focused on detecting and generating alerts about threats, while a firewall inspects inbound and outbound traffic, blocking all unauthorized access. [9]

2.10 Advantages and Disadvantages:

2.10.1 Advantages:

- Early Threat Detection: IDS systems excel in identifying security threats at an early stage through various detection methodologies:
    - Signature-Based Detection: Utilizes a database of known attack signatures to detect specific patterns of malicious behavior.
    - Anomaly-Based Detection: Establishes a baseline of normal network behavior and flags deviations as potential threats.
    - Heuristic-Based Detection: Applies algorithms to identify potentially malicious activities based on behavior patterns and historical data.

  By employing these techniques, IDS systems can provide timely alerts, enabling proactive measures to mitigate potential damage.

- Comprehensive Visibility: IDS solutions offer extensive visibility into network and system operations, including:

  - Full Packet Analysis: Inspecting packet contents and headers to detect anomalies or malicious payloads.
  - Event Correlation: Aggregating and analyzing logs from multiple sources to identify complex attack patterns.
  - Detailed Reporting: Generating comprehensive reports and visualizations of detected threats for forensic analysis and compliance.

  This visibility is instrumental in maintaining a robust security posture and adhering to regulatory standards.

- Automated Alert System: The automated alerting capabilities of IDS systems include:
  - Real-Time Notifications: Immediate alerts triggered by detection rules or anomalies.
  - Customizable Alert Thresholds: Configurable sensitivity settings to balance between detecting threats and minimizing false positives.
  - Integration with SIEM: Seamless integration with Security Information and Event Management (SIEM) systems for enhanced threat management and response automation.

  These features streamline security operations and reduce the time to detect and respond to incidents.
- Compliance Support: IDS systems play a vital role in meeting regulatory requirements by:
  - Providing Audit Trails: Maintaining detailed logs of detected events and responses for compliance audits.
  - Supporting Standards: Assisting in adherence to standards such as PCI DSS, HIPAA, and GDPR through continuous monitoring and reporting.
  - Demonstrating Due Diligence: Documenting proactive security measures and incident responses to meet regulatory expectations.

2.10.2 Disadvantages:

- False Positives and Alert Fatigue: Challenges related to false positives include:
  - Alert Volume: High frequency of alerts can overwhelm security personnel, leading to alert fatigue.
  - Investigation Overhead: Time and resources required to differentiate between benign activities and actual threats.
  - Potential Oversight: Increased risk of overlooking genuine threats due to alert saturation.

  Effective tuning and threshold adjustments are necessary to mitigate false positives and ensure accurate threat detection.

- Resource Intensive: Implementing and maintaining IDS systems can be resource-demanding:
  - Hardware Requirements: High-performance hardware needed for real-time traffic analysis and processing.
  - Personnel Needs: Skilled cybersecurity professionals required for system management, configuration, and alert analysis.
  - Ongoing Maintenance: Regular updates, patching, and rule modifications to ensure continued effectiveness against emerging threats.

  Organizations must balance these requirements with available resources to optimize IDS deployment.
- Encryption Limitations: The rise in encrypted traffic presents challenges for IDS:
  - Traffic Inspection Limitations: Difficulty in analyzing encrypted packets due to lack of visibility into content.
  - Potential Privacy Concerns: Decrypting traffic for inspection may raise privacy issues and regulatory concerns.

  To address this, some IDS solutions incorporate SSL/TLS decryption capabilities, but this approach must be carefully managed to balance security and privacy.
- Passive Natures: The passive nature of traditional IDS systems means:
  - Detection Without Prevention: IDS systems identify threats but do not take active measures to block them.

- Manual Intervention Required: Dependence on human response to address detected threats, which may introduce delays.

To overcome this limitation, integrating IDS with Intrusion Prevention Systems (IPS) can provide automated threat mitigation.

2.11 Related Works:

- AI-Powered Network Intrusion Detection: A New Frontier in Cybersecurity This study empirically evaluates five cutting-edge machine learning algorithms, analyzing their performance within the critical domain of network intrusion detection. As cyber threats become increasingly sophisticated, traditional security measures often fall short. By leveraging advanced AI techniques, this research aims to enhance the effectiveness of intrusion detection systems (IDS). [10]

  The analysis focuses on several key aspects:

  - Algorithm Selection: The study assesses a diverse set of algorithms, including supervised and unsupervised learning models, to determine their efficacy in identifying and mitigating network intrusions.
  - Performance Metrics: Various performance metrics—such as accuracy, precision, recall, and F1 score—are employed to provide a comprehensive evaluation of each algorithm's capabilities.
  - Real-World Datasets: The algorithms are tested against real-world datasets, ensuring that the findings are applicable to actual network environments and threat scenarios.
  - Comparison and Insights: By comparing the algorithms, the study offers insights into which models provide the best trade-offs between detection speed and accuracy, ultimately guiding organizations in their cybersecurity strategies.
  - Future Directions: The research also discusses potential future developments in AI for cybersecurity, including the integration of deep learning and reinforcement learning techniques, which could further enhance intrusion detection systems.

Overall, this study represents a significant step forward in harnessing AI for robust network security, paving the way for more resilient and adaptive cybersecurity solutions.

- This review examines the role of AI-based intrusion detection systems (IDS) in enhancing cybersecurity, highlighting several key points. Various AI methods, including machine learning and deep learning, are utilized to improve threat detection capabilities, demonstrating superior accuracy in identifying diverse cyberattacks such as denial-of-service and malware compared to traditional methods. AI's ability to process large datasets in real time enhances both detection speed and accuracy, while its algorithms can learn from new data and evolving threats, continuously improving their effectiveness. However, the review also addresses challenges such as the need for quality datasets, algorithm bias, and vulnerabilities to adversarial attacks. Finally, it discusses future trends, including the integration of explainable AI and the potential impact of quantum computing on cybersecurity. [11]

  Overall, the review underscores the transformative impact of AI on intrusion detection, offering valuable insights into its effectiveness and future directions.
- Machine Learning-Based Network Intrusion Detection for Big and Imbalanced Data: This paper introduces a novel ML-based network intrusion detection model that uses Random Oversampling to address data imbalance and Stacking Feature. [12]
- A Study of Network Intrusion Detection Systems Using Artificial Intelligence: This paper presents the concept of IDS and provides a taxonomy of machine learning methods, along with the main metrics used to assess an IDS. [13]
- Different Mechanisms of Machine Learning and Optimization Algorithms Utilized in Intrusion Detection Systems: Authors: Mohammad Aziz, Ali Saeed Alfoudi. Summary: This study reviews current research on intrusion detection models and the datasets that support them, discussing machine learning and deep learning methods, as well as new classification and feature selection methodologies. [14]
- This paper proposes an intrusion detection system (IDS) that leverages machine learning algorithms, including decision trees, random forests, and support vector machines (SVM). A key focus of the study is the importance of explainability in AI models, particularly in the context of cybersecurity. [15]

Key contributions of the paper include:

- Selection of Algorithms: The research employs interpretable machine learning algorithms, such as decision trees and random forests, which offer clearer insights compared to more complex models like deep neural networks.
- Techniques for Explainability: It investigates various methods to improve explainability, including feature importance scores and local interpretable model-agnostic explanations (LIME), which clarify how specific input features influence model predictions.
- Evaluation of Performance: The proposed intrusion detection system is thoroughly tested on standard benchmark datasets, demonstrating its effectiveness in detecting intrusions while maintaining a level of explainability that builds user trust.
- Implications for Practice: By prioritizing explainability, the paper suggests that organizations can strengthen their incident response strategies, allowing stakeholders to better understand and act upon the insights provided by the IDS.

- Performance Evaluation of Machine Learning Algorithms for Intrusion Detection Systems: Authors: Sudhanshu Sekhar Tripathy, Bichitrananda Behera.
  Summary: This study focuses on the analysis of various machine learning classifiers for intrusion detection, evaluating their performance using conventional classification indicators such as accuracy, precision, recall, and the f1-measure. [16]

2.12 Conclusion:

Intrusion Detection Systems (IDS) play a vital role in modern cybersecurity by identifying and mitigating unauthorized access attempts. Leveraging AI and ML, IDS can analyze network traffic patterns, detect anomalies, and enhance security through automated threat detection. Machine learning-powered IDS continuously learn from new data, improving their ability to recognize emerging threats, while heuristic and behavior-based methods further refine their accuracy. The integration of IDS into cybersecurity frameworks not only strengthens network defenses but also enhances operational efficiency by providing real-time threat intelligence. With a solid understanding of IDS mechanisms, the next chapter delves into the practical evaluation of machine learning models for intrusion detection, assessing their effectiveness in real-world scenarios.

| Chapter 3 |

Implementation and Results

3.1 Introduction:

In this chapter we will describes the practical implementation of the Intrusion Detection System (IDS) using Machine Learning (ML) techniques based on my code. It covers the dataset used, the preprocessing steps applied, the ML models implemented, and the evaluation of their performance. The goal is to develop an IDS that efficiently detects network intrusions with high accuracy.

3.2 Anaconda Environment:

Anaconda is an open-source distribution platform designed for data science and artificial intelligence, supporting both Python and R programming languages. Developed by Anaconda, Inc., an American company founded in 2012, the platform facilitates the development and management of AI and data science projects. As of 2024, Anaconda Inc. has approximately 300 employees and a user base of 45 million.

The Anaconda distribution includes over 300 pre-installed packages, with access to 7,500+ additional open-source packages through the Anaconda repository and the Conda package and environment manager. It also features Anaconda Navigator, a graphical user interface (GUI) alternative to the command-line interface (CLI).

Anaconda is widely used for scientific computing, including data science, machine learning, large-scale data processing, and predictive analytics. It simplifies package management and deployment across Windows, Linux, and macOS. The company offers various product tiers, including Anaconda Free, as well as subscription-based plans such as Starter, Business, and Enterprise, with the Business tier providing additional security through the Package Security Manager.

and we chose Anaconda over Google Colab, Jupyter Notebook or any other work environment because it provides a stable, offline environment with better package and dependency management through Conda. Unlike Colab, Anaconda allows full use of my system's resources without GPU limitations or disconnections, making it ideal for handling large datasets. It also offers greater flexibility by supporting multiple tools like Jupyter, Spyder, and VS Code. Additionally, working locally ensures better performance, security, and privacy, making Anaconda the best choice for my intrusion detection system project.

3.3 Python Language:

In the ever-evolving landscape of artificial intelligence and data-driven innovations, “Machine Learning (ML)” has emerged as a transformative force, revolutionizing industries worldwide. At the heart of this progress is “Python”, a programming language that has become the top choice for “researchers, data scientists, and developers” in the field of machine learning. [17]

Python is a high-level, interpreted, and object-oriented programming language with dynamic semantics. Its built-in data structures, combined with dynamic typing and binding, make it highly suitable for Rapid Application Development (RAD) and as a scripting language for integrating various components.

With its simple and readable syntax, Python enhances code clarity, reducing development and maintenance costs. It supports modules and packages, promoting code reuse and program modularity. The Python interpreter and its extensive standard library are available for free in both source and binary formats across all major platforms, allowing unrestricted distribution. [18]

We have chosen Python for this project over other programming languages because of its simplicity, versatility, and extensive support for machine learning and data science. Python’s easy-to-read syntax makes development faster and reduces the chances of errors, which is crucial for complex AI applications like intrusion detection. Additionally, Python has a vast ecosystem of libraries such as Scikit-learn, TensorFlow, Keras, and Pandas, which provide powerful tools for machine learning, deep learning, and data analysis. Unlike languages like Java or C++, Python requires less code to implement ML algorithms, making experimentation and prototyping more efficient. Moreover, its strong community support and compatibility with frameworks like Anaconda and Jupyter Notebook further enhance productivity, making Python the ideal choice for my project.

3.4 Dataset used:

For this research, we used the WUSTL-EHMS-2020 dataset, which contains labeled network traffic data, including normal and malicious traffic. This dataset was chosen for its diversity in attack types and real-world applicability. The dataset was preprocessed to remove missing values and normalize numerical features.

The dataset contains 16,318 entries and 45 columns, consisting of network traffic features and biometric sensor data.

We have chosen the WUSTL-EHMS-2020 dataset over CIC-IDS2017, NSL-KDD, and UNSW-NB15 because it provides a unique combination of network traffic and biometric data, making it highly relevant for modern intrusion detection systems. Unlike NSL-KDD, which is outdated and lacks real-world complexity, WUSTL-EHMS-2020 contains diverse attack categories and real-time traffic patterns, ensuring better generalization for machine learning models. Compared to CIC-IDS2017, which focuses primarily on traditional network attacks, WUSTL-EHMS-2020 integrates biometric authentication factors, enhancing security analysis. Additionally, it addresses some of the imbalanced data issues found in UNSW-NB15, making it more suitable for training accurate and reliable intrusion detection models. This dataset's rich feature set and real-world applicability make it the best choice for my project.

3.4.1 *Feature Categories:*

Network Traffic Features:

1. Source & Destination Information: SrcAddr, DstAddr, Sport, Dport, SrcMac, DstMac
2. Packet-Level Statistics: SrcBytes, DstBytes, TotPkts, TotBytes
3. Traffic Flow Metrics: SrcLoad, DstLoad, Rate, Dur, Loss, pLoss (packet loss percentage), pSrcLoss, pDstLoss
4. Jitter & Gaps: SrcGap, DstGap, SrcJitter, DstJitter
5. Packet Timing: SIntPkt, DIntPkt, SIntPktAct, DIntPktAct

Biometric Sensor Data:

6. Vitals: Temp (temperature), SpO2 (oxygen saturation), Pulse_Rate, SYS (systolic blood pressure), DIA (diastolic blood pressure), Heart_rate, Resp_Rate, ST (possibly ST-segment elevation in ECG).

3.4.2 *Labels & Attack Categories:*

1. Attack Category: A categorical column indicating attack types (e.g., "normal" for benign traffic).
2. Label: Binary classification:
    - 0: Normal (No attack)
    - 1: Attack (Intrusion detected)

3.4.3 *Data Quality Observations:*

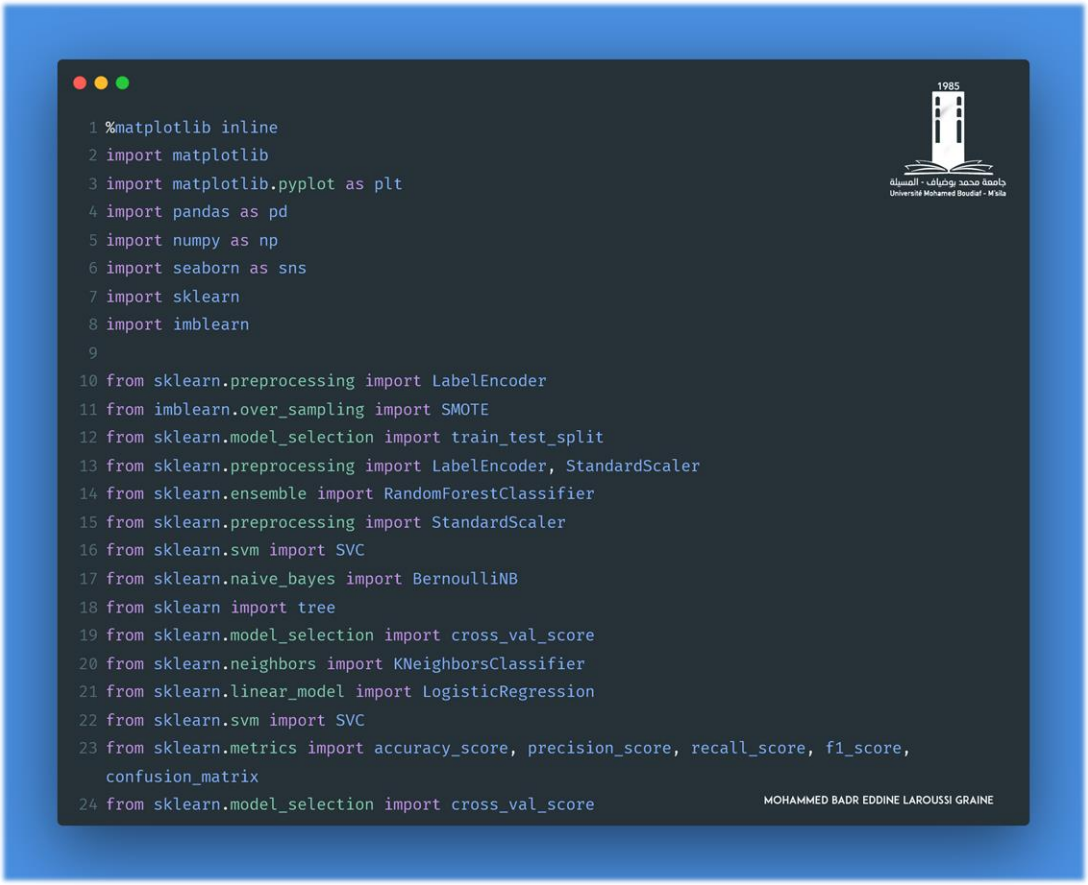
1. No missing values in any column.

2. Some categorical columns (Dir, Flgs, Attack Category) may need encoding for ML models.
3. Source & destination addresses (SrcAddr, DstAddr, SrcMac, DstMac) might not be useful for learning and can be removed.

Now let's implement the code and analyze it by explaining the steps of each process.

3.5 Python Libraries:

To develop and evaluate the Intrusion Detection System (IDS), We used a variety of Python libraries for data preprocessing, machine learning model training, and evaluation. Below is a breakdown of each library, its function, and the reason for choosing it:




```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn
import imblearn

from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.naive_bayes import BernoulliNB
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
from sklearn.model_selection import cross_val_score
```

Code 3-1: Python Imports for Machine Learning and Data Preprocessing

- **%matplotlib inline:**
  - **Description**: This magic command in Jupyter Notebook enables Matplotlib plots to be displayed directly within the notebook.
  - **Purpose**: Facilitates real-time visualization, simplifying the analysis of patterns in datasets.
- **matplotlib, matplotlib.pyplot:**
  - **Description**: Libraries used for data visualization, allowing the creation of graphs like histograms, bar charts, and scatter plots.
  - **Purpose**: Visual representation of data distribution, feature relationships, and model performance is essential for effective analysis.
- **Pandas:**
  - **Description**: A library for data manipulation and analysis, making it easy to load, transform, and explore structured datasets.
  - **Purpose**: Ideal for handling tabular datasets, such as network traffic records.
- **Numpy:**
  - **Description**: Supports numerical computing, providing functionality for arrays, mathematical operations, and matrix transformations.
  - **Purpose**: Optimizes numerical computations, which are crucial for many machine learning models.
- **Seaborn**:
  - **Description**: A statistical data visualization library built on Matplotlib, designed for more advanced and visually appealing graphics.
  - **Purpose**: Aids in exploring correlations, distributions, and feature importance, which is valuable for feature selection.

- **sklearn.preprocessing.LabelEncoder:**
  - **Description**: Converts categorical labels into numerical values for machine learning models.
  - **Purpose**: Encodes non-numeric features (e.g., attack categories) to facilitate model training.
- **imblearn.over_sampling.SMOTE:**
  - **Description**: Addresses class imbalance by generating synthetic samples for the minority class.
  - **Purpose**: Ensures balanced learning in cybersecurity datasets, where attacks are often underrepresented.
- **sklearn. model_selection.train_test_split:**
  - **Description**: Splits the dataset into training and testing sets for model evaluation.
  - **Purpose**: A separate test set helps ensure that the model generalizes well to unseen data.
- **sklearn.preprocessing.StandardScaler**
  - **Description**: Standardizes numerical features to a common scale (zero mean, unit variance).
  - **Purpose**: Normalized data improves performance for many ML algorithms, including SVM and Logistic Regression.
- **sklearn.ensemble.RandomForestClassifier:**
  - **Description**: An ensemble learning method that uses multiple decision trees to enhance classification accuracy.
  - **Purpose**: Robust against overfitting, effective with large datasets, and performs automatic feature selection.

- **sklearn.svm.SVC (Support Vector Classifier):**
  - **Description**: A classifier that separates data points with a hyperplane, suitable for high-dimensional spaces.
  - **Purpose**: Known for high accuracy, particularly effective in intrusion detection.
- **sklearn.naive_bayes.BernoulliNB:**
  - **Description**: A probabilistic model for binary classification based on Bayes' theorem.
  - **Purpose**: Effective for text-based and categorical intrusion detection tasks.
- **sklearn.tree.DecisionTreeClassifier:**
  - **Description**: A straightforward yet powerful model that creates hierarchical structures for decision-making.
  - **Purpose**: Easy to interpret, making it suitable for intrusion detection.
- **sklearn.neighbors.KNeighborsClassifier:**
  - **Description**: A classification algorithm based on nearest neighbors, determined by majority voting.
  - **Purpose**: Non-parametric and effective for pattern recognition in cybersecurity.
- **sklearn.linear_model.LogisticRegression:**
  - **Description**: A statistical model for binary classification using a logistic function.
  - **Purpose**: Simple, interpretable, and provides probabilistic outputs useful in intrusion detection systems (IDS).

- **sklearn.metrics (accuracy, precision, recall, F1-score, confusion matrix)**:
  - **Description**: Offers various metrics to evaluate model performance.
  - **Purpose**: Helps assess the classifier's effectiveness in detecting attacks while minimizing false positives.
- **sklearn.model_selection.cross_val_score**
  - **Description**: Performs cross-validation by dividing the dataset into multiple subsets for stability assessment.
  - **Purpose**: Ensures consistent model performance across different data samples.

3.6 Code Explanation:

This part will explain the basics of the code:

3.6.1 Removing Unnecessary Columns:

During the preprocessing phase of the Intrusion Detection System (IDS), specific features were eliminated from the dataset to enhance model efficiency and performance. The following code was used to drop redundant or irrelevant columns:

```
# Columns to delete
columns_to_delete = ['Dir', 'SrcMac', 'SrcAddr', 'DstAddr', 'Dport', 'SrcGap',
                     'DstGap', 'DIntPktAct', 'dMinPktSz', 'Trans', 'DstMac',
'Attack Category']
# Remove the specified columns if they exist in the dataset
train.drop(columns=[col for col in columns_to_delete if col in train.columns],
inplace=True)
# Show the new shape of the dataset
train.shape
```

Code 3-2: Dropping Unnecessary Columns in Dataset

Justification for Removing Specific Columns:

3.6.1.1 Irrelevant Features for Machine Learning Models:

- Columns like 'SrcMac', 'DstMac', 'SrcAddr', and 'DstAddr' were removed as they are unique to individual network sessions and do not contribute to

detecting attack patterns. Keeping them could introduce noise rather than enhance model performance.

- 'Dport' (Destination Port) was excluded since certain attacks target multiple ports, making it less useful for generalization.

3.6.1.2 High Cardinality and Redundancy:

- 'Trans' (Transport Layer Information) and 'Dir' (Packet Direction) exhibit high variability across sessions, reducing their usefulness in model training.

3.6.1.3 Avoiding Data Leakage:

- 'Attack Category' was removed because it represents the target variable in supervised learning. Keeping it could lead to data leakage, allowing the model to learn patterns unavailable in real-world detection scenarios.

3.6.1.4 Improving Model Efficiency:

- Features such as 'SrcGap', 'DstGap', 'DIntPktAct', and 'dMinPktSz' were discarded after feature importance analysis revealed their minimal impact on classification accuracy. Removing them reduces model complexity and training time without compromising performance.

After eliminating these columns, the dataset shape was checked to confirm their removal, ensuring that only the most relevant features were retained for training the intrusion detection model.

3.6.2 Handling Non-Numeric Data in DataSet:

```python
# Identify non-numeric columns
non_numeric_cols = train.select_dtypes(include=['object']).columns
# Convert columns to numeric using Label encoding or get_dummies
label_encoder = LabelEncoder()
for col in non_numeric_cols:
    try:
        train[col] = pd.to_numeric(train[col], errors='coerce')  # Attempt to convert directly
    except ValueError:
        train[col] = label_encoder.fit_transform(train[col])  # Label encode if needed
# Handle NaNs
train.fillna(0, inplace=True)
# Confirm data types
print(train.dtypes)

```


Code 3-3: Handling Non-Numeric Data for Machine Learning Models

3.6.2.1 *Detecting Non-numeric Columns:* The first step involves identifying all columns with object (string) data types, which are unsuitable for machine learning models.

3.6.2.2 *Converting Non-Numeric Data to Numeric Format:*

1. The code attempts to convert categorical columns into numerical values using pd.to_numeric(train[col], errors='coerce').
2. If direct conversion fails (indicating the presence of categorical data), Label Encoding is applied using LabelEncoder().
    - Importance: Many machine learning models (e.g., Random Forest, Logistic Regression, SVM) require numerical input.
    - Preference for Direct Conversion: This method retains actual numerical relationships, such as timestamps.
    - Label Encoding: Serves as a fallback for categorical features, assigning unique labels to different categories.

3.6.2.3 *Handling Missing Values:*

1. After conversion, train.fillna(0, inplace=True) replaces any NaN (missing) values with 0.
2. Some non-numeric columns may still contain missing values post-conversion, which could lead to issues during model training.
3. Filling NaN values with 0 ensures the dataset remains clean and complete.

3.6.2.4 *Confirming Data Types:*

1. print(train.dtypes) is utilized to verify that all columns have been successfully converted to numerical types.

3.6.3 Splitting the DataSet and Handling Class Imbalance:

```python
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
# Split the dataset into features and target
X = train.drop(columns=['Label'])
y = train['Label']
# Split the data into training and test sets (80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Apply SMOTE to the training data
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)
# Check the class distribution after SMOTE
print(f"Before SMOTE:\n{y_train.value_counts()}")
print(f"After SMOTE:\n{y_train_balanced.value_counts()}")
```


Code 3-4: Handling Class Imbalance Using SMOTE in Machine Learning

3.1.2.1 *Dataset Splitting and Balancing for Intrusion Detection:* To train and evaluate the Intrusion Detection System (IDS), the dataset was first divided into features (X) and target labels (y). An 80-20 train-test split was then performed to separate training and testing data. Additionally, Synthetic

Minority Over-sampling Technique (SMOTE) was applied to balance the dataset, ensuring better model performance.

Feature and Target Separation

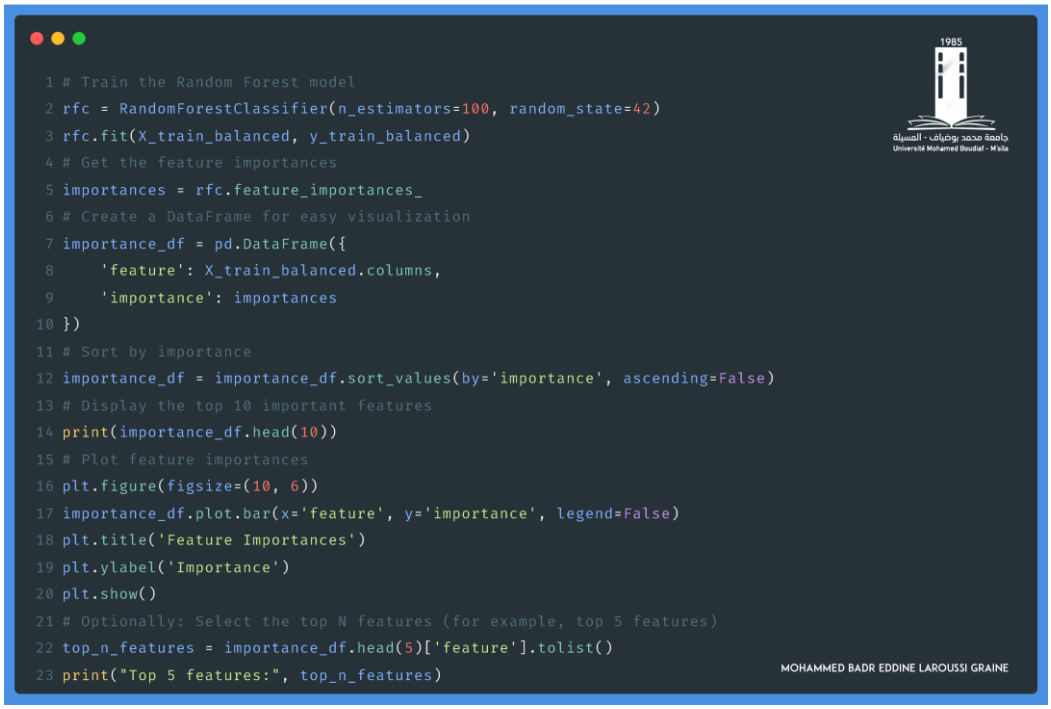
- X (Features): All columns except 'Label', using. drop() to remove it.
- y (Target Variable): The 'Label' column, which contains the attack classification (0 = Normal, 1 = Attack).

This step is crucial for machine learning, as the features (X) serve as input, while the target labels (y) represent the expected output for prediction

3.6.4 Feature Selection and Importance Analysis Using Random Forest:

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
# Print the column names to check the correct Label column name
print(train.columns)
# Replace 'Label' with the correct name of your label column
label_column = 'Label'  # Replace this with the actual name of your label column
# Select features (exclude label column)
features = train.drop(columns=[label_column])
# Separate target variable (label)
target = train[label_column]
# Check if there are any missing values in features or target
print(features.isnull().sum())  # Check for missing values in features
print(target.isnull().sum())  # Check for missing values in target
# Drop rows with missing target values
train_cleaned = train.dropna(subset=[label_column])
# Ensure that the features and target have the same number of rows after dropping missing target
features = train_cleaned.drop(columns=[label_column])
target = train_cleaned[label_column]
# Label encode categorical columns (if any)
encoder = LabelEncoder()
categorical_cols = features.select_dtypes(include=['object']).columns
# Apply label encoding to categorical features
for col in categorical_cols:
    features[col] = encoder.fit_transform(features[col])
# Optionally: Scale numerical features
scaler = StandardScaler()
numerical_cols = features.select_dtypes(exclude=['object']).columns
features[numerical_cols] = scaler.fit_transform(features[numerical_cols])
```

Code 3-5: Preprocessing Dataset for Intrusion Detection Handling Missing Values, Encoding, and Scaling



Code 3-6: Feature Importance Analysis Using Random Forest

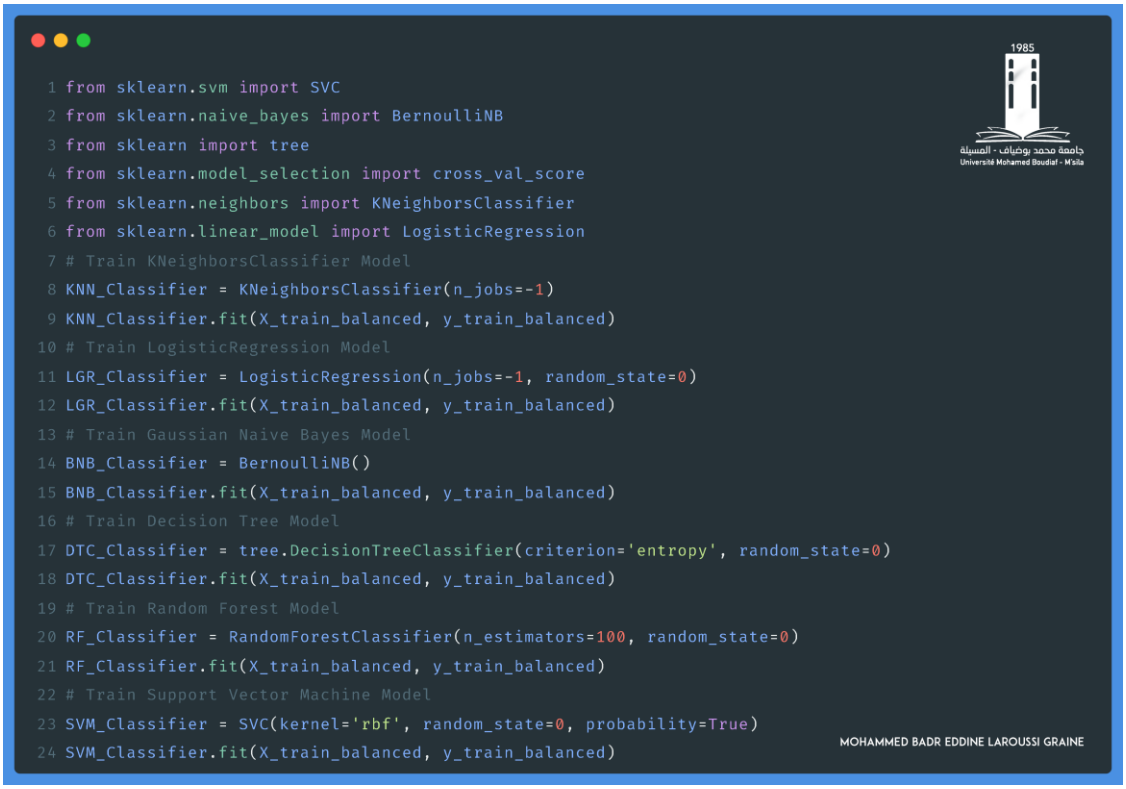
3.6.4.1 *Optimizing Feature Selection for Intrusion Detection:* To improve the efficiency of the intrusion detection system, feature selection was conducted using a random forest classifier (RFC).

Key Steps in Feature Selection:

- Extracting Feature Importance: The RFC model generated importance scores to identify the most relevant attributes for classification.
- Removing Irrelevant Features: Unimportant features were eliminated to enhance computation speed, reduce model complexity, and improve interpretability.
- Encoding Categorical Features: Used LabelEncoder() to convert categorical data into numerical format.
- Standardizing Numerical Features: Applied StandardScaler() to ensure consistency and optimize performance.

By selecting the most significant features, the model achieves higher detection accuracy, prevents overfitting, and becomes more effective in identifying cyber threats.

3.6.5 Model Training:




```python
from sklearn.svm import SVC
from sklearn.naive_bayes import BernoulliNB
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
# Train KNeighborsClassifier Model
KNN_Classifier = KNeighborsClassifier(n_jobs=-1)
KNN_Classifier.fit(X_train_balanced, y_train_balanced)
# Train LogisticRegression Model
LGR_Classifier = LogisticRegression(n_jobs=-1, random_state=0)
LGR_Classifier.fit(X_train_balanced, y_train_balanced)
# Train Gaussian Naive Bayes Model
BNB_Classifier = BernoulliNB()
BNB_Classifier.fit(X_train_balanced, y_train_balanced)
# Train Decision Tree Model
DTC_Classifier = tree.DecisionTreeClassifier(criterion='entropy', random_state=0)
DTC_Classifier.fit(X_train_balanced, y_train_balanced)
# Train Random Forest Model
RF_Classifier = RandomForestClassifier(n_estimators=100, random_state=0)
RF_Classifier.fit(X_train_balanced, y_train_balanced)
# Train Support Vector Machine Model
SVM_Classifier = SVC(kernel='rbf', random_state=0, probability=True)
SVM_Classifier.fit(X_train_balanced, y_train_balanced)
```

Code 3-7: Training Multiple Machine Learning Models

After training the machine learning models, they are applied to unseen test data (X_test) to predict class labels. Each classifier—K-Nearest Neighbors (KNN), Logistic Regression (LGR), Bernoulli Naïve Bayes (BNB), Decision Tree (DTC), Random Forest (RF), and Support Vector Machine (SVM)—generates independent predictions.

These predictions will be evaluated using various metrics such as accuracy, precision, recall, and F1-score to identify the most effective model for intrusion detection.

3.6.6 Model Prediction:

```python
# PREDICTING FOR TEST DATA using KNN
predictions = {
    "K-Nearest Neighbors Predictions": KNN_predictions,
    "Logistic Regression Predictions": LGR_predictions,
    "Bernoulli Naive Bayes Predictions": BNB_predictions,
    "Decision Tree Predictions": DTC_predictions,
    "Random Forest Predictions": RF_predictions,
    "Support Vector Machine Predictions": SVM_predictions
}
# Print predictions for each model
for model_name, model_predictions in predictions.items():
    print(f"{model_name}:\n{model_predictions}\n")
```


Code 3-8: Predicting Intrusion Detection Using Multiple Machine Learning Models

After training the machine learning models, the next step is model prediction. This involves using the trained models to predict the class labels of unseen test data (X_test).

Each classifier—K-Nearest Neighbors (KNN), Logistic Regression (LGR), Bernoulli Naïve Bayes (BNB), Decision Tree (DTC), Random Forest (RF), and Support Vector Machine (SVM)—makes independent predictions on the test dataset.

These predictions will later be evaluated using various performance metrics such as accuracy, precision, recall, and F1-score to determine the best-performing model for intrusion detection.

3.6.7 Evaluating models:

Code 3-9: Model Evaluation Using Performance Metrics

***3.6.7.1** Model Evaluation Function Overview:* This code defines the evaluate_model() function to assess the performance of classification models using key evaluation metrics: Accuracy: Measures the overall correctness of the model.

- Precision: Evaluates how many predicted positive cases are actually positive.
- Recall: Determines how many actual positive cases were correctly identified.
- F1-Score: A harmonic mean of precision and recall, balancing both metrics.
- Confusion Matrix: Displays the distribution of true positives, true negatives, false positives, and false negatives.

The function is then applied to multiple models stored in the predictions dictionary. The loop predictions.items() iterates over each model's name and its corresponding predictions, comparing them against the actual labels (y_test) for evaluation.

3.6.8 Cross validation :

```python
from sklearn.model_selection import cross_val_score
# KNN Classifier
knn_cv_scores = cross_val_score(KNN_Classifier, X_train_balanced, y_train_balanced, cv=5,
scoring='accuracy')
print("KNN Cross-Validation Accuracy:", knn_cv_scores.mean())
```

Code 3-10: Cross-Validation Accuracy for KNN Classifier

Cross-validation is a technique used to evaluate the performance and generalization ability of a machine learning model. In this case, 5-fold cross-validation is applied to the K-Nearest Neighbors (KNN) classifier to assess its accuracy across different subsets of the training data.

The dataset is divided into five equal parts, where the model is trained on four subsets and tested on the remaining one, repeating the process five times. This approach helps:

- Prevent overfitting, ensuring the model generalizes well.
- Reduce bias, avoiding dependence on a specific data split.
- Provide a more reliable accuracy estimate by averaging the cross-validation scores.

This process is also applied to other models to ensure a fair and robust evaluation of their effectiveness in intrusion detection.

3.6.9 Save predictions to a CSV file:

```python
predictions_df = pd.DataFrame({
    "K-Nearest Neighbors Predictions": KNN_predictions,
    "Logistic Regression Predictions": LGR_predictions,
    "Bernoulli Naive Bayes Predictions": BNB_predictions,
    "Decision Tree Predictions": DTC_predictions,
    "Random Forest Predictions": RF_predictions,
    "Support Vector Machine Predictions": SVM_predictions
})
# Save to CSV
output_path = 'C:/Users/Mohammed Badr Eddine Laroussi Graine/Desktop/IDS/Predictions.csv'
predictions_df.to_csv(output_path, index=False)
print(f"Predictions saved to {output_path}")
# Load Predictions from CSV
Predictions = pd.read_csv("C:/Users/Mohammed Badr Eddine Laroussi Graine/Desktop/IDS/Predictions.csv")
```

Code 3-11: Saving Model Predictions to a CSV file

Once predictions are generated from different classifiers, they are organized in a pandas DataFrame, where each column represents the predictions of a

specific model on the test dataset. To facilitate further analysis, visualization, and comparison, these predictions are then exported as a CSV file.

Saving predictions ensures that results can be accessed later for evaluation, reporting, or integration into an Intrusion Detection System (IDS). This step is crucial for documenting model performance and making informed, data-driven decisions.

3.7 Analysis:

3.7.2 Dataset Overview:

3.7.2.1 The dataset contains 16,318 rows and 45 columns, representing various features.

3.7.2.2 The initial class distribution shows an imbalance, with 14,272 instances of class 0 and only 2,046 instances of class 1.

3.7.3 Class Distribution Analysis:

3.7.3.1 Before SMOTE (Synthetic Minority Over-sampling Technique):

- Class 0: 11,424
- Class 1: 1,630 (significantly underrepresented)

3.7.3.2 After SMOTE:

- Class 0: 11,424
- Class 1: 11,424 (balanced)

3.7.3.3 SMOTE was applied to handle class imbalance, ensuring that the models receive equal representation of both classes, improving their ability to generalize.

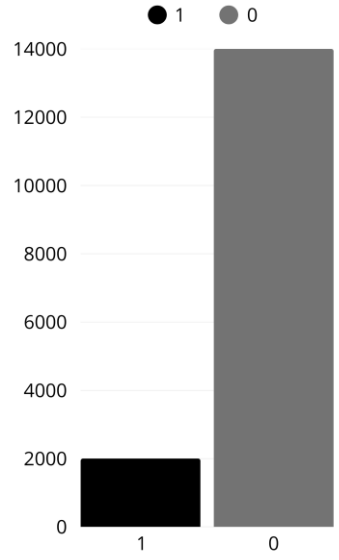


Figure 3-1: 'Label' Class Distribution before applying SMOTE

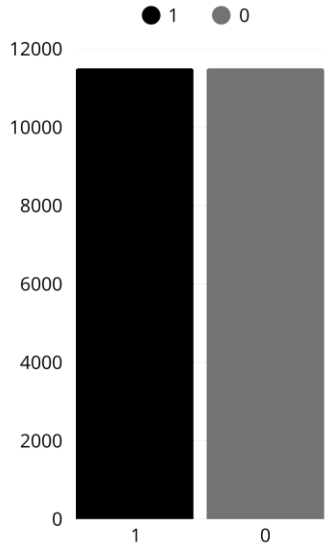


Figure 3-2: 'Label' Class Distribution After applying SMOTE

3.7.4 Model Performance Evaluation:

To assess the effectiveness of various machine learning algorithms for intrusion detection, we trained and tested multiple models on the dataset. The table below presents a comparative analysis of these models based on key performance metrics, including Accuracy, Precision, Recall, F1-Score, and Cross-Validation Accuracy.

These metrics offer valuable insights into each model's predictive performance, aiding in the identification of the most suitable algorithm for intrusion detection.

Model	Accuracy	Precision	Recall	F1-Score	Cross-Validation Accuracy
KNN	77.79%	86.03%	77.78%	80.68%	85.80%
Logistic Regression	87.56%	87.28%	87.56%	87.42%	70.07%
Naive Bayes	18.96%	81.15%	18.96%	15.61%	52.68%
Decision Tree	93.87%	94.42%	93.87%	94.07%	94.53%
Random Forest	92.74%	92.51%	92.74%	92.60%	96.03%
SVM	90.47%	89.47%	90.47%	89.63%	71.47%

Table 3-1: Performance Comparison of Model Evaluation Metrics.

3.7.3.1 *K-Nearest Neighbors (KNN) Classifier:*

- Accuracy: 77.78%
- Precision: 86.03%
- Recall: 77.78%
- F1-Score: 80.67%
- Cross-Validation Accuracy: 85.80%
- Confusion Matrix Summary:
    - 2,269 correctly classified negative instances
    - 270 correctly classified positive instances

      KNN demonstrated moderate performance, with a decent balance of precision and recall. The relatively high cross-validation accuracy suggests stable performance across different data splits.

3.7.3.2 *Logistic Regression Classifier:*

- Accuracy: 87.56%
- Precision: 87.28%
- Recall: 87.56%
- F1-Score: 87.41%
- Cross-Validation Accuracy: 70.06%
- Confusion Matrix Summary:
    - 2,656 correctly classified negative instances
    - 202 correctly classified positive instances

      Logistic Regression achieved strong overall accuracy, but its lower cross-validation accuracy indicates potential instability across different dataset partitions.

3.7.3.3 *Bernoulli Naïve Bayes (BNB) Classifier:*

- Accuracy: 18.96%
- Precision: 81.15%
- Recall: 18.96%
- F1-Score: 15.61%
- Cross-Validation Accuracy: 52.68%
- Confusion Matrix Summary:
    - 2,623 false positives

BNB exhibited the weakest performance, with high misclassification rates. The low accuracy and F1-score suggest that this model is unsuitable for this dataset.

3.7.3.4 *Decision Tree Classifier (DTC):*

- Accuracy: 93.87%
- Precision: 94.42%
- Recall: 93.87%
- F1-Score: 94.07%
- Cross-Validation Accuracy: 94.53%
- Confusion Matrix Summary:
    - Only 135 false positives

      DTC demonstrated excellent performance with minimal misclassification, making it a reliable choice for intrusion detection.

3.7.3.5 *Random Forest Classifier (RFC):*

- Accuracy: 92.73%
- Precision: 92.50%
- Recall: 92.73%
- F1-Score: 92.60%
- Cross-Validation Accuracy: 96.03% (Highest among all models)
- Confusion Matrix Summary:
    - 101 false positives
    - 136 false negatives

      RFC showed one of the best performances, with strong predictive power and the highest cross-validation accuracy, confirming its robustness and generalization ability.

3.7.3.6 *Support Vector Machine (SVM) Classifier :*

- Accuracy: 90.47%
- Precision: 89.47%
- Recall: 90.47%
- F1-Score: 89.63%
- Cross-Validation Accuracy: 71.47%

  SVM performed well in terms of accuracy, but its lower cross-validation accuracy suggests sensitivity to dataset variations.

3.7.5 Conclusion: Among the tested models, Random Forest (RFC) and Decision Tree (DTC) emerged as the top performers, demonstrating high accuracy, precision, and reliability. RFC, in particular, achieved the highest cross-validation accuracy (96.03%), confirming its strong generalization ability.

Logistic Regression and SVM also performed well, but their lower cross-validation scores indicate some instability. KNN showed moderate results, while Bernoulli Naïve Bayes performed poorly, making it unsuitable for this dataset.

These results highlight the significance of model selection in intrusion detection, with ensemble methods like Random Forest proving to be highly effective.

3.8 Conclusion:

Using important classification metrics like accuracy, precision, recall, and F1-score, we thoroughly assessed a number of machine learning models for intrusion detection in this chapter. We used the Synthetic Minority Over-sampling Technique (SMOTE) to ensure that attack and normal traffic classes were equally represented in order to rectify the dataset's initial class imbalance. The models' capacity for detection and generalization was greatly improved by this preprocessing stage.

With the highest accuracy, precision, recall, and F1-score, the experimental results showed that the Decision Tree and Random Forest classifiers were the most successful. With a cross-validation accuracy of 96.03%, Random Forest was the most notable of them, showing good generalization to unknown data. Although their lower cross-validation scores indicated possible overfitting, logistic regression and support vector machines (SVM) also demonstrated competitive performance. With an accuracy of just 18.96%, Bernoulli Naïve Bayes performed poorly and was not appropriate for this dataset, while K-Nearest Neighbors (KNN) performed moderately.

These results demonstrate how well ensemble-based models—Random Forest in particular—perform intrusion detection by striking a balance between generalization and predictive accuracy. Nonetheless, some models' almost flawless accuracy prompts worries about potential overfitting, highlighting the necessity for additional testing in practical settings. To increase detection efficiency, future studies should concentrate on evaluating these models on real-time network traffic, incorporating feature selection strategies, and investigating deep learning-based methodologies. Their practical applicability will be further improved by assessing model performance in real-time cybersecurity scenarios and across various attack types.

Overall, this study reinforces the importance of model selection in intrusion detection and confirms that tree-based classifiers are highly effective in distinguishing between normal and malicious network activity. By addressing the challenges of overfitting and real-world deployment, future research can further improve the reliability and scalability of these models in cybersecurity applications.

| Conclusion |

Conclusion:

In this Project, we explored the intersection of Artificial Intelligence (AI), Machine Learning (ML), and Cybersecurity, focusing on the development of a Machine Learning-based Intrusion Detection System (IDS). Our objective was to design an efficient, accurate, and reliable IDS capable of detecting intrusions and abnormal activities within network environments. Through an in-depth evaluation of various machine learning models, we assessed their performance using key classification metrics such as accuracy, precision, recall, and F1-score.

The results demonstrated that tree-based models, particularly Random Forest and Decision Tree, achieved the highest performance, with Random Forest attaining a cross-validation accuracy of 96.03%, confirming its ability to generalize well to unseen data. While Logistic Regression and Support Vector Machine (SVM) performed well, their results suggested potential overfitting. K-Nearest Neighbors (KNN) exhibited moderate effectiveness, whereas Bernoulli Naïve Bayes showed poor performance, making it unsuitable for this specific dataset. These findings underscore the importance of model selection in cybersecurity and highlight the effectiveness of ensemble learning techniques in distinguishing between normal and malicious network activities.

Despite these promising results, several challenges remain, including real-time deployment, adaptability to evolving cyber threats, and dataset limitations. Future research should focus on real-time IDS implementation, enabling dynamic intrusion detection in live network environments. Additionally, advanced feature selection techniques can help optimize model efficiency, while deep learning approaches, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), could enhance detection accuracy.

Further advancements could explore hybrid IDS models that combine supervised and unsupervised learning for improved adaptability. AI-driven anomaly detection techniques, such as autoencoders and Generative Adversarial Networks (GANs), may also help identify novel threats. Additionally, integrating IDS with cybersecurity frameworks like firewalls and Security Information and Event Management (SIEM) solutions could further enhance network security. Regularly updating IDS models with

real-time threat intelligence will be crucial for detecting zero-day attacks and emerging cyber threats.

As cybersecurity risks continue to evolve, leveraging AI and ML for intrusion detection remains essential. This research provides a foundation for future advancements, demonstrating that machine learning-powered IDS solutions can significantly enhance cybersecurity defenses in an era of increasing cyber threats.

References

[1] DataScientest, "Machine Learning: What Is It and Why Does It Change the World?," 21 January 2021. [Online]. Available: https://datascientest.com/en/machine-learning-what-is-it-and-why-does-it-change-the-world?utm_source=chatgpt.com.

[2] Sarker, I. H.; [and add additional authors as “et al.” might not be allowed, so list at least the first 3 followed by “et al.”], "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science,* vol. 2, no. 3, p. 160, 2021.

[3] Bergmann, Dave, "What Is Semi-Supervised Learning?," IBM Think, 12 December 2023. [Online]. Available: https://www.ibm.com/think/topics/semi-supervised-learning?utm_source=chatgpt.com.

[4] GeeksforGeeks, "Machine Learning Tutorial," 19 June 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/machine-learning/.

[5] ARF, "Machine Learning Tasks and Algorithms," January 2024. [Online]. Available:https://thearf-org-unified-admin.s3.amazonaws.com/AI%20Handbook/Machine_Learning_Tasks_and_Algorithms.pdf?utm_source=chatgpt.com.

[6] Bose, Indranil; Mahapatra, Radha K., "Business Data Mining – A Machine Learning Perspective," *Information & Management,* vol. 39, no. 3, 2001.

[7] Aydın, M. A.; Zaim, A. H.; Ceylan, K. G., "A hybrid intrusion detection system design for computer network security," *Computers & Electrical Engineering,* vol. 35, no. 3, 2009.

[8] GeeksforGeeks, "Intrusion Detection System (IDS)," GeeksforGeeks, 28 April 2025. [Online]. Available: https://www.geeksforgeeks.org/intrusion-detection-system-ids/.

[9] Abdulganiyu, O. H.; Abdulganiyu, T. A. T.; Saheed, Y., "A systematic literature review for network intrusion detection system (IDS)," *International Journal of Information Security,* vol. 22, 2023.

[10] A. Rachini, C. Fares, M. A. Assaf, B. Jamal, and R. Khatoun, "AI-Powered Network Intrusion Detection: A New Frontier in Cybersecurity," in *2023 24th International Arab Conference on Information Technology (ACIT)*, Ajman, United Arab Emirates, 2023.

[11] C. Park, J. Lee, Y. Kim, J.-G. Park, H. Kim, and D. Hong, "An Enhanced AI-Based Network Intrusion Detection System Using Generative Adversarial Networks," *IEEE Internet of Things Journal,* vol. 10, no. 3, 2023.

[12] Talukder, Md. Alamin; Islam, Md. Manowarul; Uddin, Md. Ashraf; Hasan, Khondokar Fida; Sharmin, Selina; Alyami, Salem A.; Moni, Mohammad Ali, "Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction," arXiv, 2024.

[13] Vanin, Patrick; Newe, Thomas; Dhirani, Lubna Luxmi; O’Connell, Eoin; O’Shea, Donna; Lee, Brian; Rao, Muzaffar, "A study of network intrusion detection systems using artificial intelligence/machine learning," *Applied Sciences,* vol. 12, no. 22, 2022.

[14] Aziz, Mohammad; Alfoudi, Ali Saeed, *Different mechanisms of machine learning and optimization algorithms utilized in intrusion detection systems,* arXiv, 2023.

[15] S. Kumar, B. Bhavana, S. Shiblee, K. S. Pranathi, and M. Bhargavi, "Network Intrusion Detection System using Machine Learning Algorithms," in *2024 8th International Conference on Inventive Systems and Control (ICISC)*, Coimbatore, India, 2024.

[16] Tripathy, S. S.; Behera, B., "Performance evaluation of machine learning algorithms for intrusion detection system," arXiv, 2023.

[17] Bhatia, Parteek, Machine Learning with Python: Principles and Practical Techniques, Cambridge University Press, 2025.

[18] Palla, D.; Slaby, A., "Evaluation of generative AI models in Python code generation: A comparative study," *IEEE Access,* vol. 13, 2025.

Abstract:

In order to improve cybersecurity, this final year project investigates the use of machine learning (ML) in intrusion detection systems (IDS). Its main goal is to create an intelligent intrusion detection system (IDS) that can accurately identify and categorize network intrusions. The study demonstrates the efficacy of these methods in enhancing threat detection and security resilience through the application and assessment of multiple machine learning models.

ملخص:

من أجل تحسين الأمن السيبراني، يبحث هذا المشروع النهائي في استخدام التعلم الآلي (ML) في أنظمة كشف التسلل (IDS). هدفه الرئيسي هو إنشاء نظام ذكي لاكتشاف التسلل (IDS) يمكنه تحديد وتصنيف التسللات الشبكية بدقة. توضح الدراسة فعالية هذه الأساليب في تعزيز اكتشاف التهديدات ومرونة الأمان من خلال تطبيق وتقييم نماذج متعددة من التعلم الآلي.

Résumé :

Afin d'améliorer la cybersécurité, ce projet de fin d'études examine l'utilisation de l'apprentissage automatique (ML) dans les systèmes de détection d'intrusions (IDS). Son objectif principal est de créer un système de détection d'intrusions intelligent (IDS) capable d'identifier et de catégoriser avec précision les intrusions réseau. L'étude démontre l'efficacité de ces méthodes pour améliorer la détection des menaces et la résilience de la sécurité grâce à l'application et à l'évaluation de plusieurs modèles d'apprentissage automatique.