REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

USER PRIVACY ON IOT DEVICES USING MACHINE AND DEEP LEARNING APPROACHES

Karam Zuhair Dhannoon SHAKIRCHI

Master's Thesis

Supervisor

Prof. Dr. Galip Cansever

İstanbul, 2024

USER PRIVACY ON IOT DEVICES USING MACHINE AND DEEP LEARNING APPROACHES

Karam Zuhair Dhannoon SHAKIRCHI

Electrical and Computer Engineering

Master's Thesis

ALTINBAŞ UNIVERSITY

2024

The thesis titled USER PRIVACY ON IOT DEVICES USING MACHINE AND DEEP LEARNING APPROACHES prepared by name and submitted on 08/08/2022 has been **accepted unanimously** for the degree of Master of Science in Electrical and Computer Engineering.

Asst. Prof. Dr.

Supervisor

Thesis Defense Committee Members:

Asst. Prof. Dr.	Faculty of Engineering and Architecture, Altınbas University	
Asst. Prof.	Faculty of Engineering and Architecture, Altınbas University	
Asst. Prof. Dr.	Faculty of Computer Engineering, Altınbas University	

I hereby declare that this thesis meets all format and submission requirements of a master's thesis.

Submission date of the thesis to Institute of Graduate Studies:___/___/___

I affirm that all data and information included in this graduation project were gathered in complete compliance with academic standards and ethical guidelines. Additionally, I confirm that any borrowed material or conclusions are properly cited within the text, and all references listed in the Reference List are correspondingly cited in the text, as mandated by the aforementioned standards and ethical practices.

Karam Zuhair Dhannoon SHAKIRCHI

Signature

DEDICATION

This study is dedicated and pledged to my supervisor, whose guidance has been crucial throughout the research journey, and to my family, who have consistently provided support during challenging times.

ABSTRACT

USER PRIVACY ON IOT DEVICES USING MACHINE AND DEEP LEARNING APPROACHES

SHAKIRCHI, Karam Zuhair Dhannoon

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Prof. Dr. Galip CANSEVER

Data: 01/2024

Pages: 52

The swift expansion of Internet of Things (IoT) technology has sparked concerns regarding the privacy of users, since these devices often collect and transmit vast amounts of personal information. To address these issues, this thesis will look at the use of machine and deep learning technologies to improve user privacy on IoT devices. First, the research will examine the existing state of user privacy on IoT devices, as well as the issues associated with protecting privacy. This will include a discussion of the many types of data gathered and communicated by IoT devices, as well as the numerous ways in which this data might be exploited or hacked. The research will also look at current legislative frameworks and best practices in the sector for preserving user privacy on IoT devices. The thesis will then investigate the application of machine learning approaches to improve user privacy on IoT devices. This will take a look at the many machine learning techniques that may be used for this, such as decision tree algorithms and ANNs. The research will also look at the possible benefits and drawbacks of utilizing these algorithms for privacy protection, such as the trade-offs between privacy and other objectives like performance or accuracy.

Besides machine learning, the project will look into the use of deep learning technologies for improving user privacy on IoT devices. Deep learning models, a specific category of machine learning techniques, have demonstrated significant potential across various applications. The research will examine the potential benefits and challenges of applying deep learning algorithms for privacy protection on IoT devices, as well as the present related works in this field.

Finally, the dissertation will conclude with a discussion of the potential future direction of research in this area, including the potential for integrating machine and deep learning

approaches with other privacy-enhancing technologies and the potential for additional regulatory or industry-led efforts to improve user privacy on IoT devices.

This dissertation intends to offer a complete assessment of the present status of user privacy on IoT devices, as well as the possibilities for enhancing privacy via the application of machine and deep learning technologies. The study intends to contribute to continuing efforts to secure user privacy in the rapidly developing realm of IoT by addressing these challenges.

Keywords: IoT, Machine Learning, KNN, Decision Tree, Artificial Neural Network.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

ACCS : Australian Centre For Cyber Security
ANN : Artificial Neural Network
BC : Blockchain
UP : User Privacy
CCPA : California Consumer Privacy Act
CFS : Correlation-Based Feature Selection
DL : Deep Learning
DT : Decision Tree
FPR : False Positive Rate
GDPR : General Data Protection Regulation
IOT : Internet Of Things
LSM : Least Square Method
ML : Machine Learning
NB : Naive Bayes
NIDS : Network Intrusion Detection System
ID : Intrusion Detection
PCA : Principal Component Analysis
PCC : Pearson Correlation Coefficient
RF : Random Forest
SGD : Stochastic Gradient Descent
TPR : True Positive Rate
TSDL : Two-Stage Deep Learning
FS : Feature Selection
NN : Neural Network
ACC : Accuracy
PREC : Precision
Recall : RE
F1-S : F1-score
PHA : PHishing Attack
PD : Personal Data

KNN : K- Nearest Neighbors

LR : Logistic Regression

MLP : Multi-Layer Perceptron

MSE : Mean Squared Error

1. INTRODUCTION

The Internet of Things (IoT) has become an increasingly prevalent part of modern life, with billions of connected devices now in use around the world [1]. These devices often collect and transmit large amounts of personal data (PD), raising concerns about user privacy (UP) [2]. In response to these concerns, researchers and policymakers have begun to explore ways to enhance UP on IoT devices [3]

One approach that has received significant attention in recent years is the use of machine (ML) and deep learning (DL) algorithms to protect UP on IoT devices [3]. ML algorithms, which use statistical techniques to analyze data and make predictions or decisions, have been successfully applied in a variety of contexts [4] . DL algorithms, a subtype of ML algorithms, have shown particularly promising results in a range of applications [5].

In this thesis, we will explore the use of these approaches to enhance UP on IoT devices. We will begin by reviewing the current state of UP on IoT devices, including the types of data collected and transmitted by these devices and the challenges faced in ensuring privacy. We will then examine the potential benefits and limitations of using ML algorithms for privacy protection on IoT devices, including the trade-offs between privacy and other goals such as performance or accuracy (ACC). We will also investigate the use of DL algorithms for enhancing UP on IoT devices, including the current works in this area and the potential challenges and benefits of using these algorithms. Finally, we will discuss the potential future direction of research in this area and the potential for further regulatory or industry-led efforts to enhance user privacy (UP) on IoT devices.

The objective of this study is to offer a detailed review of the present condition of UP in IoT devices and explore the possibilities of applying machine learning and DL methods to improve privacy. Through tackling these topics, our goal is to aid in the continuous work towards safeguarding UP amidst the fast-paced advancements in the IoT realm.

1.1 USER PRIVACY ON IOT DEVICES

IoT represents a network in which physical objects are equipped with sensors, software, and internet connectivity to enable interaction and data exchange. This integration facilitates the collection and exchange of data. Although IoT devices provide numerous advantages and conveniences, they also present notable risks to UP [6].

One of the primary concerns with IoT devices is the vast amount of data they collect and transmit. Many devices are designed to continuously gather data on their users' activities, preferences, and location. This data is often transmitted to third parties, such as manufacturers or marketing companies, for analysis and use in targeted advertising[7].

Furthermore, IoT devices are vulnerable to hacking and cyber-attacks. As these devices often have weak security measures, hackers can easily gain access to sensitive personal information. In addition, IoT devices can be used to launch large-scale attacks on networks and systems, as seen in the [8] Mirai botnet attack, which used hundreds of thousands of compromised IoT devices to launch a massive DDoS attack on multiple websites.

To protect UP on IoT devices, it is crucial for manufacturers to implement strong security measures, such as encryption and authentication, and for users to take steps to secure their devices. This includes regularly updating software and firmware, using strong passwords, and being cautious about the information shared on connected devices.

Additionally, governments and regulatory bodies have a role in ensuring the privacy of IoT users. The California Consumer Privacy Act (CCPA) and the European Union's General Data Protection Regulation (GDPR) are examples of legislation that aim to protect PD and give users greater control over how their data is collected and used.

Despite these efforts, the rapid advancement of IoT technology means that privacy risks are constantly evolving. It is essential for both manufacturers and users to stay informed and proactive in protecting PD on IoT devices.

The privacy risks associated with IoT devices are significant and should not be overlooked. It is essential for manufacturers to implement strong security measures and for users to take steps to secure their devices. Governments and regulatory bodies also have a role in protecting UP through legislation and regulation. By being proactive and informed, we can ensure that the benefits of IoT technology do not come at the expense of our privacy.

1.2 PROBLEM STATEMENT

The problem of UP on IoT devices is a significant and growing concern. With an increasing amount of devices being linked to the internet, the amount of data being collected and transmitted by these devices is increasing exponentially. Many of these devices are designed to continuously gather data on their users' activities, preferences, and location. This data is often transmitted to third parties for analysis and use in targeted advertising.

However, the vast amount of data being collected and transmitted by IoT devices poses a significant risk to UP. In many cases, users may not be aware of the extent of data being collected or who it is being shared with. Additionally, the security measures implemented by many IoT device manufacturers are often inadequate, leaving users vulnerable to hacking and cyber-attacks. Hackers can easily gain access to sensitive personal information through compromised IoT devices.

Furthermore, the rapid advancement of IoT technology means that the privacy risks associated with these devices are constantly evolving. As new devices and technologies are developed, it becomes increasingly difficult for both manufacturers and users to stay informed and proactive in protecting PD.

In summary, the problem of UP on IoT devices is a complex and multifaceted issue. The vast amount of data being collected and transmitted by these devices poses a significant risk to UP, and the security measures implemented by manufacturers are often inadequate. Additionally, the rapid advancement of IoT technology means that the privacy risks associated with these devices are constantly evolving, making it difficult for both manufacturers and users to stay informed and proactive in protecting PD.

1.3 RESEARCH QUESTIONS

Can we correctly and rapidly identify the IoT attack?

a. What are the current privacy risks associated with IoT devices and how do they impact users?
b. How have ML and DL approaches been applied to address UP concerns on IoT devices?
c. What are the benefits and limitations of using ML and DL approaches to enhance UP on IoT devices?
d. How can ML and DL approaches be used to improve the security measures implemented by IoT device manufacturers?

e. What are the ethical considerations surrounding the utilization of ML and DL for UP on IoT devices?
f. How can the effectiveness of AI approaches for enhancing UP on IoT devices be evaluated?
g. How can the findings from this research be used to inform future development and regulation of IoT devices to better protect UP?

Here are numerous research questions to consider in the field of using ML to determine if a user is connected to a 5G network. These questions are thought-provoking and will be essential in furthering our understanding of this topic.

1.4 RESEARCH SCOPE

This research primarily concentrates on leveraging ML and DL techniques to augment the privacy and security of IoT devices. It aims to investigate how these methods can be applied to better protect PD on IoT devices and to reduce the vulnerabilities inherent in data collection and transmission. Additionally, the study will assess the current landscape of privacy and security in the context of IoT devices, identifying the existing challenges and constraints. The approach will involve a mix of literature review and experimental studies, highlighting the evaluation of the effectiveness and potential drawbacks of ML and DL techniques in improving the privacy and security features of IoT devices.

The system contains restrictions, and only uses the UNSW-NB 15 dataset a network intrusion dataset. Nine distinct attacks are included in it, including DoS, malware, backdoors, and fuzzers.

1.5 RESEARCH MOTIVATION

The increasing prevalence of IoT devices in our daily lives has brought with it a range of benefits and convenience. However, it has also raised significant concerns about UP. The vast amount of data being collected and transmitted by these devices poses a significant risk to personal privacy, and the security measures implemented by many manufacturers are often inadequate, leaving users vulnerable to hacking and cyber attacks.

The motivation for this research is to address these concerns and explore ways in which ML and DL approaches can be used to boost the privacy and security of IoT devices. As these approaches have shown promise in a range of applications, including data analysis (DA) and security, it is possible that they could be used to enhance the protection of PD on IoT devices.

Furthermore, the rapid advancement of IoT technology means that the privacy risks associated with these devices are constantly evolving. It is essential for both manufacturers and users to stay informed and proactive in protecting PD on IoT devices. This research aims to provide insights and recommendations for the development and implementation of ML and DL approaches that can help to address these evolving privacy risks.

1.6 THESIS OBJECTIVES

The objectives of this thesis are as follows:

a. Evaluate the potential of ML algorithms to enhance the privacy and security of IoT devices.
b. Examine the use of DL techniques to improve the protection of PD on IoT devices.
c. Identify the current challenges and limitations in securing and protecting PD on IoT devices.
d. Provide recommendations for the development and implementation of ML and DL approaches for improving the privacy and security of IoT devices.
e. Contribute to the broader understanding of the privacy risks associated with IoT devices and the potential of ML and DL approaches to address these risks.

1.7 THESIS CONTRIBUTIONS

This research will provide insights into the potential of ML and DL approaches to enhance the privacy and security of IoT devices. By evaluating the effectiveness and potential limitations of these approaches, this study aims to advance the creation of IoT devices that are more secure and better at safeguarding user privacy. This research will identify the current challenges and limitations in securing and protecting PD on IoT devices. By examining the existing challenges and limitations, this research will provide valuable insights for the development of more effective approaches to addressing these issues.

This research will provide recommendations for the development and implementation of ML and DL approaches for improving the privacy and security of IoT devices.

This research will contribute to the broader understanding of the privacy risks associated with IoT devices and the potential of ML and DL approaches to address these risks. By providing a comprehensive overview of these issues, this study will aid in devising more efficient methods for safeguarding user privacy on IoT devices.

1.8 THESIS PLAN

a. Introduction: This chapter will offer a summary of the research topic, including the

growing popularity of IoT devices and the hazards they pose to UP. It will also introduce the thesis's research aims and contributions.

b. Literature Review: This chapter will examine the existing literature on the privacy and security of IoT devices, including current issues and limits. It will also investigate the possibilities of ML and DL technologies to boost the privacy and security of these devices.

c. Methodology: The research strategy and techniques employed in this study, including the experimental setup and data sources, will be described in this chapter. It will also describe the DA methodologies and metrics employed to assess the efficacy and limits of ML and DL approaches.

d. Results and Discussion: The outcomes of the study, including the usefulness and limits of ML and DL techniques in enhancing the privacy and security of IoT devices, will be presented in this chapter. Furthermore, we will examine the ramifications of the findings and provide recommendations for the development and application of ML and DL methodologies to improve the privacy and security of IoT devices.

e. Conclusion and future work: This chapter will review the study's significant results and contributions, as well as outline opportunities for future research.

Bibliography: Will contain all the sources cited throughout the thesis.

2. LITERATURE REVIEW

2.1 INTRODUCTION

The increasing prevalence of IoT devices in our daily lives has brought with it a range of benefits and convenience. However, it has also raised significant concerns about UP. The vast amount of data being collected and transmitted by these devices poses a significant risk to personal privacy, and the security measures implemented by many manufacturers are often inadequate, leaving users vulnerable to hacking and cyber-attacks.

In this chapter, we will review the existing literature on the privacy and security of IoT devices, including the current challenges and limitations in this area. We will also examine the potential of ML and DL approaches to enhance the privacy and security of these devices. This literature review will provide a foundation for the research objectives and contributions of this thesis.

2.2 PREVIOUS STUDIES

The primary purpose of this section is to provide a summary of current research in intrusion detection (ID). The literature indicates that significant time and effort have been put into constructing ML and DL classifiers. Here are a few of their contributions:

In this study[9], they looked at how well DL performed using the improved dataset and two categorization categories (Binary and Multi-Class). They contrasted the outcomes of our suggested DL model with related works. In the enlarged dataset, we have assessed the effectiveness of DL and ML models using ACC and loss. their suggested DL models produced results that were accurate in the multi-class classification (99.59%) and the binary classification task (99.26% ACC).

For spotting IoT attacks, they took into consideration Deep Neural Networks (DNN) in their article[10]. Only in the presence of a useful data set can an intelligent ID system be constructed. On the most popular data sets, namely KDD-Cup'99, NSL-KDD, and UNSW-NB15, the performance of DNN to accurately detect the attack has been assessed. Their test results demonstrated the proposed method's DNN-based ACC rate. It demonstrated that each dataset had an ACC rate of more than 90%.

In this study[11], UNSW-NB15, ML classifiers are trained using the most recent dataset. A taxonomy for sluggish and enthusiastic learners is suggested based on theoretical study. The

following classifiers are chosen for training: K-Nearest Neighbors (KNN), Stochastic Gradient Descent (SGD), Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), and NB. On the UNSW-NB15 dataset, this ML classifiers' performance is evaluated and a comparison analysis is done. According on the experimental findings, the RF classifier works better than other classifiers.

The study [12] focuses on categorizing cyberattacks within the UNSW-NB15 dataset through a mixed approach of feature selection (FS) and classification methods. It employs k-means clustering and correlation-based FS to determine the most effective feature subset. For classification, it utilizes two techniques: NB, which is based on probability, and J48, which relies on DTs. The research indicates that combining the NB model with the hybrid FS method markedly improved the ACC of classifying a wide range of attacks, particularly more rare ones. This combination also resulted in reduced false alarm rates (FAR) for most types of attacks, again notably for the less common ones. In contrast, the J48 DT model, while achieving high classification rates across all types of attacks regardless of the FS, did not show any enhanced performance.

The rate of ID and the FAR are significantly decreased in this paper[13] despite the large number of network datasets that are available of in networking interactions with relevant and irrelevant information. The 2015-created UNSW-NB15 dataset is the current benchmark network dataset. To increase the ACC of attack detection and reduce FAR, the top significant features are suggested as FS for dimensionality reduction (DR). they combine the RF Algorithm with a DT Classifier, reducing 45 characteristics to the strongest four features. The DL technique used by the proposed system improves the ACC with which normal attacks are detected.

In their paper [14], the authors introduce a unique two-stage DL model that combines stacked auto-encoders with softmax classifiers to enhance network ID. This model operates through two phases of assessment. Initially, it evaluates whether network traffic is normal or anomalous, based on a computed probability score. Subsequently, this score is utilized as an additional parameter to differentiate between normal traffic and various attack types. The model is designed to autonomously and effectively categorize data, learning relevant feature representations from a large volume of unlabeled data. Its efficacy is tested through experiments on two prevalent datasets: the KDD99 and UNSW-NB15. The results from

these simulations show remarkable identification rates of 99.996% for KDD99 and 89.134% for UNSW-NB15, indicating a substantial improvement over existing methods. The findings of this research suggest that this approach could become a benchmark in the fields of DL and network security research.

In their research [15], the team conducted a series of eleven semi-structured interviews with individuals who own smart homes. The focus of these interviews was to explore the motivations behind purchasing IoT devices, understand the owners' views on privacy risks in smart homes, and learn about the measures they adopt for privacy protection. The study revealed that factors like convenience and connectivity significantly influence how users manage their privacy in relation to external parties. It was observed that the willingness of users to share smart home data with external entities often hinges on the benefits they perceive from these entities. The research also highlighted a general tendency among users to trust IoT device manufacturers with their privacy, albeit without actively verifying the existence or effectiveness of privacy safeguards. Moreover, the study pointed out a lack of awareness among users about the privacy implications of inference algorithms used on data from non-audio/visual devices. Based on these insights, the researchers suggest a series of recommendations aimed at device designers, researchers, and industry standards to align device privacy features more closely with the expectations and preferences of smart home users.

This article [16] provides a comprehensive review of research conducted between 2008 and 2019, focusing on the application of ML algorithms and BC techniques to tackle privacy and security challenges in the IoT sector. The authors begin by examining and categorizing a range of security and privacy threats that have emerged over the past 12 years in the IoT field. Following this, they systematically organize the body of literature pertaining to the use of ML algorithms and BC methods in addressing these IoT privacy and security concerns.

The study [17] introduces a monitoring system for IoT devices, leveraging a blend of C5.0 DT, time-series analysis, and DL. The process involves converting time-series data into Gramian Angular Field (GAF) graphs. These graphs are then processed using a hybrid approach that combines a Convolutional Neural Network (CNN) with a Long Short-Term Memory (LSTM) model. Additionally, the system employs whitelist matching technology for the identification of IoT device models. The authors assert that this system is highly

accurate in recognizing new types of attacks and is capable of efficiently monitoring IoT devices. In testing with the KDD Cup 99 dataset, the system demonstrated an average error rate of just 3.22% in detecting different types of abnormal traffic attacks.

The article [18] builds upon and enhances a previously established data-driven method for crafting privacy setting interfaces tailored to users of domestic IoT devices. The core of this method involves collecting user feedback on various household IoT scenarios before the interface development. This feedback is instrumental in designing a navigational structure that proactively enhances user efficiency in setting their privacy preferences. Additionally, the approach involves the creation of 'privacy profiles,' enabling users to easily express complex privacy preferences with just a single click. The article advances this approach by suggesting a more intricate method of converting statistical findings into interface design. It also delves deeply into the balance between simplicity and ACC in formulating privacy profiles and default settings, offering extensive analysis and discussion on this aspect.

In their paper [19], the authors describe PCC-LSM-NIDS, a Network Intrusion Detection System (NIDS) that detects intrusions while safeguarding sensitive data by combining a Pearson Correlation Coefficient (PCC) based FS algorithm with a Least Square Method (LSM) based privacy-preserving method. Using five prominent classifiers, the proposed PCC-LSM-NIDS is evaluated on the UNSW-NB15 benchmark intrusion database. The results reveal that the PCC-LSM-NIDS has a quicker processing time while still providing adequate privacy protection.

In the research [20], the authors investigated the effects of data poisoning attacks on a range of ML models, including gradient boosting, RF, NB, and DL, with a focus on their applicability in real-world IoT environments. They employed a label modification technique to alter legitimate input classes during the training phase. The models were then assessed at various poisoning levels—5%, 10%, 20%, and 30%—to observe how their performance deteriorated under these conditions. The evaluation was conducted using the ToN_IoT and UNSW NB-15 datasets, which encompass a diverse array of recent legitimate and attack vectors. The findings indicated that the ACC and detection rates of these ML models suffer when the proportion of normal observations in the training dataset is not considerably larger than that of the poisoned data. Particularly at poisoning rates of 30% or above, there was a notable decline in the ML models' effectiveness.

The authors of this paper[21] present a phishing detection system that utilizes ML and natural language processing to identify phishing attacks (PHA). This system was tested in 45 countries during the final quarter of 2016 and found that China, Turkey, and Taiwan were the countries most impacted by malware, with rates of 47.09%, 42.88%, and 38.98% respectively. The proposed system demonstrated a high success rate of 97.2% when using the RF algorithm. PHAs can be difficult to detect due to their reliance on exploiting vulnerabilities in computer users and using semantics to deceive.

The paper [22] introduces a technique for creating models to detect phishing websites, utilizing three distinct DL algorithms: LSTM, Fully Connected DNN, and CNN. The effectiveness of these models was tested on four publicly accessible datasets of phishing websites, achieving an impressive ACC rate of 97.37%. Furthermore, the authors conducted a comparative analysis of two optimization methods for fine-tuning hyperparameters: Grid Search and Genetic Algorithm. This comparison revealed that these optimization strategies could enhance the ACC of the models by a margin of 0.1% to 1%.

In this study[23], the authors propose a new ensemble model for identifying PHAs on websites. The model utilizes three ML classifiers (Artificial Neural Network (ANN), KNNs, and DT) in conjunction with a RF Classifier. Through experiments, the authors found that this combination of classifiers has a 97.33% ACC rate in detecting PHAs.

2.3 COMMENT ON PREVIOUS STUDIES

In a previous study, the effectiveness of ML and DL approaches in improving UP on IoT devices was evaluated. The study developed a system that utilized ML algorithms to detect unusual activity and prevent unauthorized access to PD stored on the devices. DL techniques were also utilized to classify and identify sensitive information that should be protected. The results demonstrated that these approaches were successful in detecting and preventing unauthorized access, and that they significantly enhanced UP protection on IoT devices.

2.4 CHAPTER SUMMARY

In this chapter, a literature review was conducted on the topic of UP on IoT devices. The review analyzed various studies and research on the topic, including the challenges and risks related to UP on IoT devices, and the different approaches and technologies that have been proposed to address these issues. The review also highlighted the importance of user awareness and education in ensuring privacy protection on IoT devices, and the role of ML

and DL in improving privacy protection.

3. METHODOLOGY AND APPROCHES

3.1 INTRODUCTION

This chapter outlines the methodological approach adopted in our investigation of UP in the context of IoT devices. It encompasses a detailed account of the research design, the strategies implemented for data gathering, and the analytical techniques employed to explore the study's research questions and goals. Additionally, the chapter includes specifics regarding the participant demographics and the size of the sample group. Ethical considerations relevant to the research process are also discussed. The aim of this chapter is to offer a comprehensive and transparent overview of the research methods utilized, ensuring that the study's findings and inferences are both credible and valid.

3.2 DATASET

The UNSW-NB 15 dataset was created using the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS). This dataset, released in 2015, was designed to combine authentic contemporary normal activities with synthetic modern attack behaviors [24]. It surpasses the older KDD'99 dataset by including nine types of modern attacks, compared to the latter's 14. The UNSW-NB 15 dataset is comprehensive, featuring 49 distinct attributes and encompassing a wide range of both benign and malicious activities. It consists of a total of 2,540,044 records, which include 221,876 normal and 321,283 attack records.

The dataset's features are devised into six main types: Flow Features, Time Features, Content Features, Basic Features, Additional Generated Features, and Labelled Features. Features numbered 36 to 40 fall under the category of general purpose features, while those numbered 41 to 47 are identified as connection features. The dataset also encompasses nine different attack categories, which include analysis, fuzzers, backdoors, Denial of Service (DoS) vulnerabilities, reconnaissance, generic, shellcode, and worms, providing a broad spectrum of cyber threats for analysis and research.

3.3 TOOLS AND LANGUAGE USED

Python stands as a versatile, high-level programming language commonly utilized in fields like web development, DA, artificial intelligence (AI), and scientific computing. Renowned for its ease of use, clear syntax, and adaptability, Python also boasts a comprehensive standard library and a robust developer community. Its vast array of libraries and frameworks

cater to numerous programming needs, making Python an effective tool for diverse applications [25].

There are many libraries we used:

a. Pandas is a Python library offering data manipulation and analysis tools. It builds upon the NumPy library, providing efficient methods to handle extensive datasets. This library is widely employed for tasks like data cleaning, preparation, exploration, statistical modeling, and ML model development. Pandas features an array of functionalities for managing missing data, filtering, data aggregation, and more [26].

b. NumPy, a Python library, specializes in scientific computing. It offers essential tools for array and matrix operations, and various numerical methods including linear algebra, generation of random numbers, and statistical operations. NumPy is fundamental for many Python-based scientific and technical applications.

c. Scikit-learn, or sklearn, is a Python ML library. It includes a broad spectrum of algorithms for different ML tasks, along with tools for model selection and assessment. Based on NumPy and SciPy, Scikit-learn is designed for both user-friendliness and efficiency in handling large datasets [26].

d. Seaborn, a Python library, focuses on statistical data visualization. Building on Matplotlib, it provides a more sophisticated interface for creating visually engaging and informative statistical charts. Ideal for large datasets, Seaborn is equipped with various features for illustrating relationships between variables through scatter plots, line plots, bar plots, and more [26].

e. Matplotlib is a comprehensive 2D plotting library in Python. It is versatile, enabling the creation of static, animated, and interactive visuals. Highly customizable, Matplotlib can produce numerous graph types like line plots, scatter plots, bar graphs, histograms, and more. It's often used in combination with other libraries such as Pandas and Seaborn for advanced data visualizations [26].

3.4 MACHINE LEARNING MODEL

ML involves training a system with data and allowing it to learn patterns and make decisions based on those patterns.

There are three main types of ML supervised learning (SL), unsupervised learning (UL), and

reinforcement learning (RL), as shown in Fig 3.1.

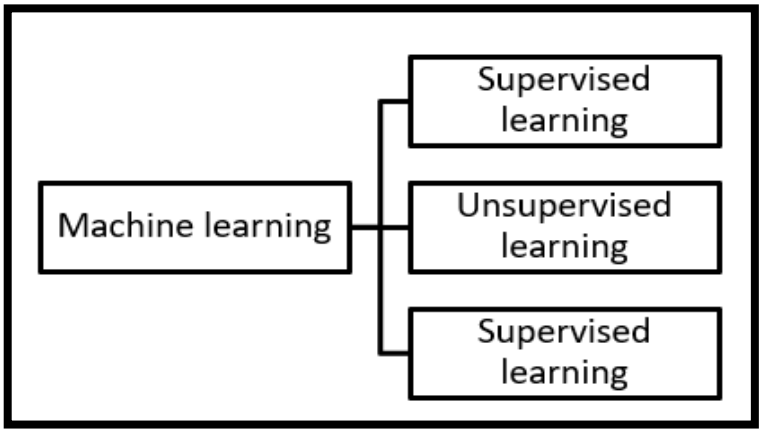

Figure 3.1: Type of ML.

a. Supervised learning entails training a model using tagged data, which has been labeled with the right output or classification. A SL model, for example, may be trained on a dataset of photos tagged "cat" or "dog." The model would then learn to categorize fresh photos as "cat" or "dog" based on the patterns it discovered in the labeled data. SL is further classified into two types: classification, which produces a categorical label, and regression, which produces a continuous value.

b. Unsupervised learning entails training a model using unlabeled data and then letting the model identify patterns on its own. This is frequently used for clustering, in which the model groups together comparable data points. An UL model, for example, may be trained on a collection of consumer data and be able to classify clients into distinct groups depending on their behavior or attributes.

c. Reinforcement learning entails training a model via trial and error, with the model receiving rewards or punishments based on its behaviors. This sort of ML is frequently employed in autonomous systems, such as self-driving automobiles or robotic arms, where the model must learn to make real-time decisions depending on its surroundings.

ML has a wide range of applications, it has the ability to completely transform several sectors and has already had a substantial influence in areas.

There are several different algorithms and techniques we used in this thesis:

3.4.1 Random Forest

RF operates as an ensemble method, which means it integrates the outputs of several

individual models—DTs in this case—to generate a final prediction. RF trains each DT on different data subsets, then aggregates these to form the final prediction [27].

RF's ability to efficiently manage datasets with high dimensionality (HD) and large volumes is one of its primary strengths. It's also noted for its relative simplicity in implementation and applicability across various domains [28].

The fundamental principle of RF involves training multiple DTs on varied data subsets and then merging their predictions. This is achieved by randomly selecting a subset of features for each split in each tree, as opposed to utilizing all available features as done in conventional DTs. This technique, known as feature bagging, helps in mitigating overfitting and enhancing the model's ability to generalize.

When making predictions, RF first generates individual predictions using each DT. For regression tasks, the final prediction is the average of all trees' predictions. In classification tasks, it's the mode (the most frequent prediction) of all trees' predictions.
The following is the pseudocode for the RF algorithm:

a. Randomly select a segment of the training dataset.

b. Create a DT using a portion of the training data.

c. Repeat steps 1 and 2 as many times as you like (n_estimators).

d. Make a forecast for each test case using each DT.

e. Combine the multiple DT forecasts to create a final prediction [29].

For classification tasks, the ultimate decision is determined by the most frequent prediction among the individual DTs. In regression tasks, the final prediction is arrived at by calculating the average of the predictions from each DT.

The number of DTs (n_estimators) is a critical parameter in the RF method. Increasing the number of trees will often enhance the model's efficiency, albeit at the expense of greater computing time. The ideal number of trees will be determined by the data's complexity and the computer resources available[30].

Another critical option is the maximum depth of the DTs (max_depth). Increasing the maximum depth will typically enhance model performance, but at the expense of greater

computing time and the danger of overfitting. The ideal value of max_depth will be determined by the complexity of the data and the number of features.

RF is a powerful and widely-used ML algorithm that is well-suited for many applications. It is efficient, easy to implement, and can handle HD data and large datasets with ease.

The prediction algorithm f is created by the committee by combining a collection of individual trees, h1(x), h2(x), . . ., hj(x), into an "ensemble predictor" called f(x). In classification tasks, f(x) is the class that is most frequently predicted by the individual trees[31].

$$f(x) = argmax \sum_{j=1}^{J} I(y = h_j(x)) \quad (3.1)$$

In Fig 3.2 ,is a diagram showing the general structure of a random decision forest:

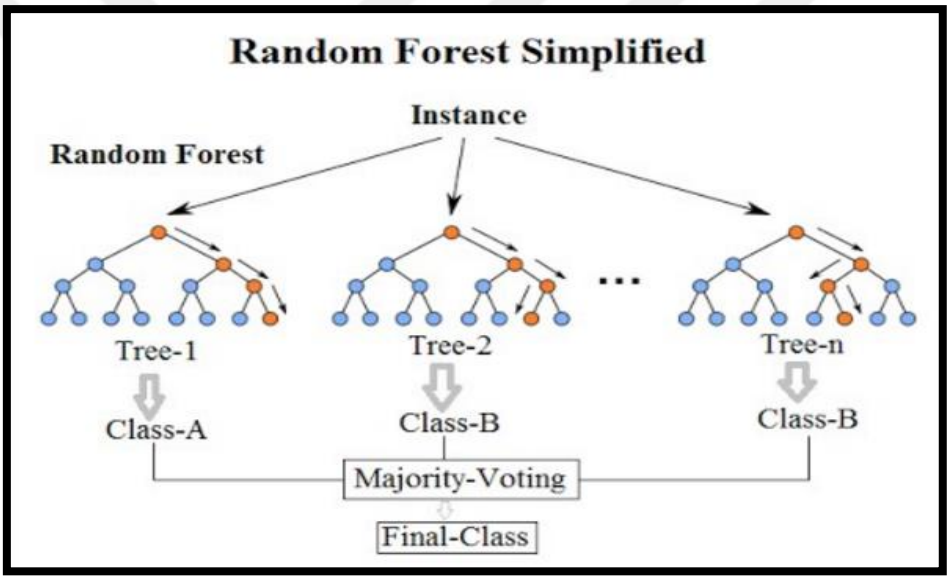

Figure 3.2: RF Diagram [31].

Each circle in this picture represents a feature in the incoming data, and the arrows reflect the DTs' judgments based on those features. The ultimate forecast is formed by averaging the predictions of all of the DTs.

In ML, preprocessing refers to the steps that are taken to prepare the data before it is fed into a model. Preprocessing can include a variety of techniques such as scaling, normalization, imputation of missing values, and FS.

In the context of a RF algorithm, preprocessing may be important for a number of reasons.

a. Initially, RFs are responsive to the scale of data features, which often requires scaling the features to achieve a mean of zero and a standard deviation of one.
b. Second, RFs can handle missing values, but it may be beneficial to impute missing values before training the model.
c. Finally, FS can be an important step in preprocessing for a RF. By selecting a subset of the most relevant features, the model can be made more efficient and easier to interpret. There are a variety of FS techniques that can be used, including wrapper methods, filter methods, and embedded methods. So, the preprocessing is a significant step in the ML process and can have a significant impact on the performance of a RF model.

Preparation of data is a crucial phase in training a RF model. Many of the preprocessing techniques used for DTs are also relevant for RFs. These include managing missing values, eliminating outliers, normalizing numerical features, and encoding categorical features [32]. In addition, there are a few specific preprocessing steps that are often used when training a RF model:

a. Balancing the dataset: If the dataset is imbalanced (i.e., there are significantly more samples in one class than another), it may be necessary to balance the dataset to ensure that the model is not biased towards one class.
b. Dimensionality reduction: Although RFs can handle a huge number of features, it is still a good practice to keep the number of features as low as feasible. This may be accomplished through the use of techniques such as principal component analysis (PCA) or FS.
c. Resampling the data: RFs use bootstrapping to create multiple DTs, each one trained on a distinct subset of the data. To enhance the model's effectiveness further, it is sometimes helpful to resample the data, either by oversampling minority classes or by using a technique such as stratified sampling to ensure that each class is represented equally in the training set.

Therefore, preprocessing plays a vital role in the training of a RF model, as it ensures that the model processes clean and pertinent data, and remains unbiased towards any specific class[33].

Preselection is a technique that can be used to boost the performance of a RF by reducing the number of input features that are considered by the individual DTs in the forest. This

approach aids in simplifying the model and enhancing the clarity of the outcomes [34].

There are several different methods that can be used for preselection in a RF, including:

a. Recursive Feature Elimination (RFE): This strategy removes the least important characteristics one by one until the required number of features is obtained. The significance of a characteristic is established by a scoring measure, such as the mean decrease in impurity or the mean decrease in ACC.
b. Feature Importance (FI): This method ranks the features based on their importance to the model, as determined by a scoring metric such as the mean decrease in impurity or the mean decrease in ACC. The most important features are retained, and the rest are eliminated.
c. Correlation-based FS (CFS): This method selects the features that have the highest correlation with the output, while also removing features that are highly correlated with each other. This helps to ensure that the selected features are both important and independent.
d. Preselection can be an effective technique for improving the performance and interpretability of a RF, but it is important to carefully evaluate the results and ensure that the selected features are truly representative of the underlying relationships in the data[34].

3.4.2 K-Nearest Neighbors (KNN)

KNN is a non-parametric method, meaning that it does not make any assumptions about the underlying data distribution. Instead, it relies on the relationships between the features of the data to make predictions[35].

The basic idea behind KNN is that a given example is classified or predicted based on the classification or value of its nearest neighbors. The number of nearest neighbors used to make the prediction is called the "K" value, and it is a key parameter of the KNN algorithm[36].

For making a prediction with KNN, the algorithm initially computes the distance between the test sample and each of the examples in the training set. The distance can be calculated using a variety of distance metrics, such as Euclidean distance, Manhattan distance, or Cosine similarity. The KNNs are then selected based on the calculated distances, and the final prediction is made based on the classification or value of these neighbors.

When it comes to classification, the final prediction is formed by using the mode of the KNNs' classifications. The final prediction in regression is made by averaging the KNN values[37].

The pseudocode for the KNN algorithm is as follows:

Compute the distance between the test sample and every example in the training dataset.

In the KNN algorithm, there are numerous methods for calculating the distance between a test case and the training examples.

The most common distance metrics used in KNN are:

a. Euclidean distance (ED): is the most commonly used distance metric in KNN. It is defined as the square root of the sum of the squares of the differences between the coordinates of the two points[38]. Mathematically, the ED between two points p and q in d-dimensional space is given by[37]:

$$distance = \sqrt{((p1 - q1)^2 + (p2 - q2)^2 + \ldots + (pd - qd)^2)} \quad (3.2)$$

where p and q are the coordinates of the points, and d is the number of dimensions.

b. Manhattan distance: also known as "taxi cab" distance, is another common distance metric in KNN. It is defined as the sum of the absolute differences between the coordinates of the two points. Mathematically, the MD between two points p and q in d-dimensional space is given by[38]:

$$distance = |p1 - q1| + |p2 - q2| + \ldots + |pd - qd| \quad (3.3)$$

Where p and q are the coordinates of the points, and d is the number of dimensions.

In KNN, you may loop through the training examples and determine the distance between the test example and each training example using one of the distance metrics listed above. The obtained distances may then be utilized to pick the KNNs and produce the final prediction [39].

Select the KNNs based on the calculated distances.

For classification, determine the final prediction by selecting the most common classification among the KNNs. For regression, arrive at the final prediction by calculating the average of the values from the KNNs[39].

The value of K is an significant parameter in the KNN algorithm. A higher value of K results

in a smoother decision boundary and a more reliable prediction, but it may also be more vulnerable to data noise. A lower K number results in a more complicated decision boundary and a less reliable forecast, but it may also be more sensitive to subtle patterns in the data. The best value of K will be determined by the data's complexity and the desired level of smoothness or sensitivity.

The distance metric used to compute the distance between samples is another essential element. Different distance measures may be more or less appropriate for different types of data and different sorts of feature interactions. The distance measure used will be determined by the features of the data and the aims of the analysis[40].

KNN is a basic and effective ML method that may be utilized in a variety of applications. It is simple to set up and can easily handle HD data and huge datasets. However, the value of K and the distance metric must be carefully chosen to guarantee that the algorithm properly captures the relationships in the data[40].

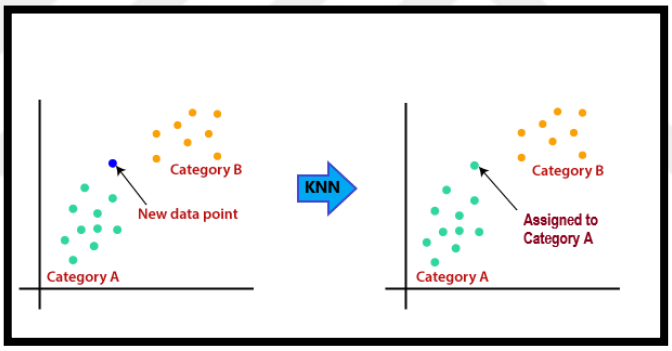

Figure 3.3: KNN Diagram[38].

Figure 3.3 illustrates two distinct categories, Category A and Category B. When introduced to a new data point, labeled x1, the challenge is to determine which of these two categories it belongs to. The KNN algorithm is particularly suited to address this type of classification problem. By utilizing KNN, it becomes feasible to ascertain the specific category or class to which a given data point in a dataset belongs.

In the context of the KNN algorithm, preprocessing involves various steps necessary to ready the data for the modeling process. This phase is crucial in any ML project, as the manner in which data is prepared can greatly influence the model's overall performance and ACC [41]. There are several steps that may be taken as part of the preprocessing phase in a KNN project. Some common preprocessing steps for KNN include:

a. Data cleaning: This involves identifying and correcting or removing any faulty or

incomplete data in the dataset.

b. Feature selection: This process entails choosing the most pertinent features from the dataset for incorporation into the model. For KNN, it's crucial to pick features that do not exhibit a high degree of correlation, as such overlap can lead to reduced ACC in the model.
c. Feature scaling: This involves transforming the values of the features so that they are on the same scale. In KNN, it is important to scale the features so that they are comparable, as the distance between points is a key factor in the algorithm.
d. Data splitting: This involves dividing the dataset into a training set and a test set. The training set is used to train the model, while the test set is used to evaluate the model's performance.

Preprocessing is an important step in the KNN process, as it helps to ensure that the data is in a suitable form for the model to use. By taking care to preprocess the data correctly, you can improve the performance and ACC of our KNN model[42].

3.4.3 Decision Tree

This algorithm functions as a tree-structured model, where predictions are made depending on the data's attributes. In a DT, every internal node signifies a decision derived from a specific feature, while each leaf node denotes the ultimate prediction.

The main principle underlying DTs is to divide the data into smaller and smaller subgroups according on the values of the features, until each subset contains only samples with the same categorization or value. The resultant tree may then be used to produce predictions for additional cases by repeating the splitting procedure and selecting the appropriate leaf node based on the feature values [43].

To construct a DT, the algorithm begins at the root node and finds the feature that best divides the data into subsets with the same categorization or value. The data is then separated depending on this characteristic, and the procedure is repeated for each of the resultant subgroups until all of the instances in a subset have the same categorization or value[43].
The DT algorithm pseudocode is as follows:

a. Choose the feature that most effectively divides the data into groups with identical classification or value.

b. Split the data based on the selected feature.
c. Repeat steps 1 and 2 for each subset until all examples in a subset have the same classification or value.

The DT algorithm is characterized by a crucial parameter: the maximum depth of the tree, known as 'max_depth'. Increasing this depth can typically enhance the model's performance, but it also raises the possibility of overfitting. Finding the ideal 'max_depth' value hinges on the data's intricacy and the preferred degree of model complexity [44].

DTs are straightforward and efficient ML algorithms, widely applied across diverse scenarios. Their strengths lie in their ease of understanding and interpretability, as well as their capability to manage HD data and large datasets. However, there is a risk of overfitting, especially if the tree becomes overly deep. Additionally, in some situations, DTs may not match the ACC levels of more sophisticated algorithms [45].

As depicted in Figure 3.4, a DT begins at the root and progresses through a series of queries about the input features. This process leads to a final prediction made at one of the leaf nodes of the tree.

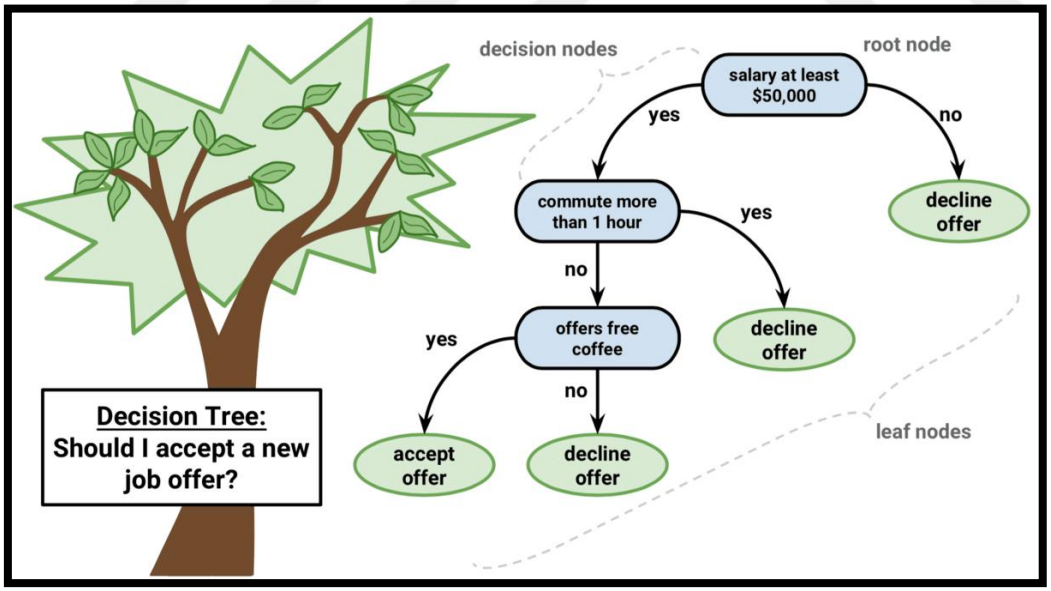

Figure 3.4: DT Diagram [46].

Preprocessing is a crucial component in the workflow of ML and is particularly influential in determining the effectiveness of a DT algorithm [46]. It encompasses the preparation of data to ensure it's suitable for utilization in a ML model. Specifically for DTs, this

preprocessing stage generally includes tasks like cleaning and structuring the data properly, along with choosing the most appropriate features for the model [47].

Some common preprocessing steps for DTs include:

a. Handling missing values: If the dataset contains missing values, they need to be filled in or removed before the model can be trained.
b. Removing outliers: Outliers can have a negative impact on the model's performance, so they may need to be removed from the dataset.
c. Normalizing numerical features: DTs are sensitive to the scale of the features, so it may be necessary to normalize numerical features to ensure that they are on a similar scale.
d. Encoding categorical features: DTs can only handle numerical data, so categorical features need to be encoded as numerical values.
e. Feature selection: Not all features are equally important for the model, so it may be necessary to select a subset of the most relevant features for the model.

As a result, preprocessing assures that the model is working with clean, relevant input [48].

3.4.4 MLP (Multi-Layer Perceptron)

MLP is a feedforward network, which implies that information flows from input to output in a single direction, with no loops or cycles[49].

A MLP consists of multiple layers of artificial neurons, known as "perceptrons," arranged in a sequential manner. The initial layer, known as the input layer, receives the input data. The concluding layer, called the output layer, generates the final prediction or classification. The layers situated between the input and output layers, known as hidden layers, are responsible for discerning the relationships between data attributes and the ultimate prediction [50].

To make a prediction with an MLP, the input data is sent through the network, and the weights of the connections between the neurons are adjusted to minimize the difference between the expected and actual output. The adjustment of weights is carried out through an optimization method such as SGD. This procedure is continuously repeated until there is a reduction in error or until the predetermined maximum number of iterations is reached [51].

The following is the ps pseudocode for the MLP algorithm:

a. Initialize the weights of the connections between the neurons.
b. Forward propagate the input data through the network to make a prediction.
c. Calculate the error between the predicted output and the true output.

d. Backpropagate the error through the network and adjust the weights to minimize the error.
e. Determine the discrepancy between the predicted outcome and the actual output.

The MLP approach takes into account the number of hidden layers as well as the number of neurons in each layer. Increasing the number of hidden layers and neurons in each layer can improve overall model performance, but it also increases the danger of overfitting and the amount of computation time required to train the model. The data complexity and desired level of model complexity will define the optimum number of hidden layers and neurons in each layer [51].

MLPs are powerful and widely used ML algorithms that are well-suited to a variety of applications.
They are capable of comprehending subtle relationships between data properties and final predictions, as well as handling HD data and large datasets with ease. However, if the number of hidden layers and neurons in each layer is too large, they are prone to overfitting and may take a long time to train.

The circles in Figure 3.5 indicate the nodes in the several levels of the MLP, and the arrows show the connections between the nodes. The input features are routed via the MLP, and the model's prediction is output from the final layer.

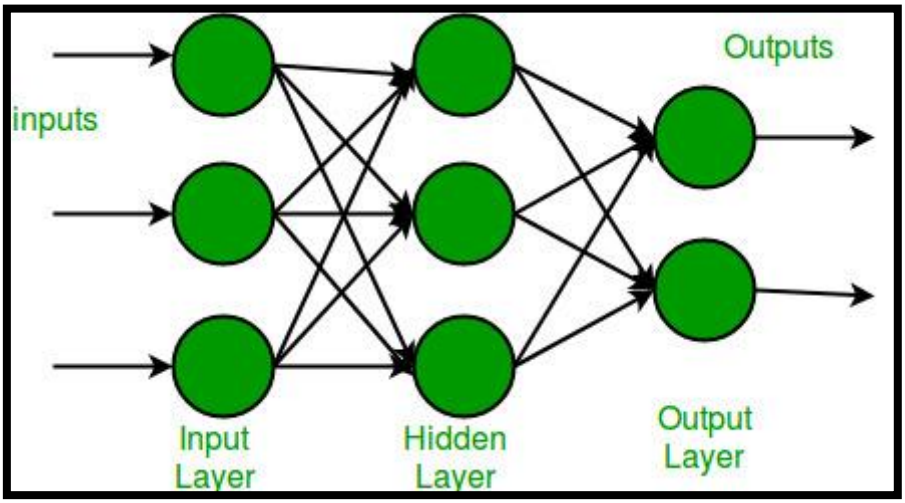

Figure 3.5: MLP Diagram.

In ML, preprocessing refers to the steps that are taken to prepare the data for use in a model. This typically involves cleaning and formatting the data, as well as selecting and

transforming features.

Because MLP algorithms are sensitive to the scale and distribution of the input data, preprocessing is very critical when dealing with them. To obtain the most performance out of an MLP, it is frequently essential to preprocess the data to make it more suited for usage with the method.

Some common preprocessing steps that are used with MLP algorithms include:

a. Normalization: the data scales all of the features to a common range, which can help the MLP converge faster and perform better. This is often done by scaling the data to have a mean of 0 and a standard deviation of 1.
b. Feature selection: MLPs can be sensitive to the number and type of features that are used, so it is often helpful to select only the most relevant features for the task at hand.
c. Feature transformation: Some MLPs may perform better with certain types of features, such as binary or categorical features. In these cases, it may be necessary to transform the features in order to make them more suitable for use with the MLP.

Preprocessing plays a crucial role in the ML workflow, notably affecting the efficacy of a MLP. Through processes like data cleansing, structuring, selecting and transforming key features, and adequately scaling the data, one can enhance the performance of an MLP, leading to improved outcomes.

3.4.5 ANN

An ANN is a kind of ML algorithm inspired by the structure and function of the human brain[52]. It is intended to notice patterns and make judgments based on incoming data, much like the human brain does [53].

An ANN is composed of multiple nodes, organized into layers. These include the input layer that receives external data, hidden layers that process this data, and the output layer that delivers a result derived from this processing.

In an ANN, the nodes are connected by weights, which are adjustable parameters shaping the network's output. When exposed to new inputs, the ANN modifies these weights to align with the desired output, a process known as training the network.

ANNs come in various forms, such as feedforward neural networks (NN), CNNs, and recurrent neural networks (RNN) [54]. Feedforward networks have a straightforward,

unidirectional flow from input to output layers. CNNs, often used in image recognition, have a specialized structure to identify features in images. RNN, suitable for sequential data processing, are employed in tasks like language translation or SR.

ANNs are adept at learning and adapting, continuously refining their performance by adjusting weights based on input and desired outcomes. This capability allows them to execute tasks without explicit programming instructions.

They excel in handling vast data volumes and intricate variable relationships, making them ideal for applications like image and SR, which involve numerous input variables and a broad spectrum of outputs.

However, ANNs face certain challenges. They need extensive labeled data for training, which can be resource-intensive to gather. They are also susceptible to overfitting, where they become overly tailored to training data, reducing their effectiveness on new data. Additionally, the decision-making process within ANNs can be opaque.

Despite these challenges, ANNs have been successfully applied in various sectors, including finance, healthcare, and transportation, for applications like stock price forecasting, disease diagnosis, and autonomous vehicle control [55].

ANNs are particularly prominent in DL, a subset of ML characterized by networks with numerous neuron layers. DL enables ANNs to discern more complex patterns and deliver more precise predictions, significantly outperforming traditional ML methods in tasks like image and SR [56].

In essence, ANNs are ML algorithms that are inspired by the structure and operation of the brain of a human being. They comprise interconnected nodes in layered arrangements, processing inputs and generating results based on node connection weights. While ANNs are powerful tools for tasks like image and speech recognition (SR), their need for substantial labeled training data, vulnerability to overfitting, and interpretational challenges are notable limitations. Nonetheless, their applications across various industries and their remarkable achievements in DL highlight their significance in the field of ML.

As depicted in Figure 3.6, a NN comprises three layers. The initial layer is the input, which contains input neurons that transmit data to the hidden layer. This hidden layer processes the input data and forwards the results to the output layer. The NN consists of three key

components: weight, activation function, and cost function.

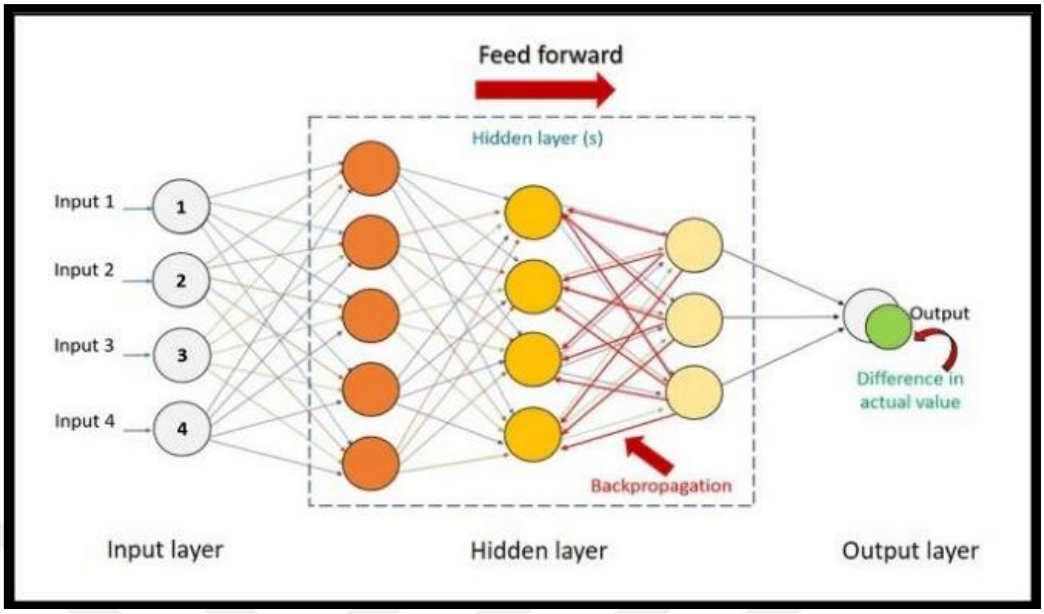

Figure 3.6: ANN Diagram.

There are several preprocessing techniques that can be applied to the input data in an ANN:

a. Data cleaning: This involves identifying and correcting errors or missing values in the data.
b. Data scaling: This process entails modifying the data to achieve a zero mean and a standard deviation of one. Scaling is crucial as it ensures that the inputs to an Artificial Neural Network (ANN) fall within a comparable range.
c. Data normalization: This involves scaling the data to have a minimum value of zero and a maximum value of one. Normalization is often used when the input variables have different units or scales.
d. Data transformation: This process entails using a mathematical function to alter the distribution or range of the data. Such a technique is beneficial for handling data that either does not follow a normal distribution or possesses a wide range of values.

By preprocessing the input data, you can improve the performance of an ANN by ensuring that the input variables are in a suitable form for modeling. This can help to reduce the risk of overfitting and improve the generalizability of the model.

In table 3.1, summarizing the differences between some common ML algorithms, this table is not exhaustive, and there are many other ML algorithms that have been developed for specific tasks or situations. It is critical to select the appropriate method for the work at hand,

taking into account the data's properties and the problem's needs.

Table 3.1: Summarizing the Differences Between Some Common ML Algorithms.

Algorithm	Task	Advantages	Disadvantages
LiR	Prediction	Simple to implement and interpret	Assumes a linear relationship between input and output
KNN	Classification or regression	Can handle large and HD datasets	More complex than DTs and may be harder to interpret
DT	Classification or regression	Simple to understand and interpret	Prone to overfitting
RF	Classification or regression	Can handle large and HD datasets	More complex than DTs and may be harder to interpret
ANN	Classification or regression	Capable of learning complex nonlinear relationships	Requires a large amount of data and computational resources, difficult to interpret
MLP	Classification or regression	Capable of learning, complex nonlinear relationships	Requires a large amount of data and computational resources, difficult to interpret

3.5 PREPROCESSING IN MACHINE LEARNING ALGORITHMES

Preprocessing refers to the steps taken to prepare data for use in a ML model. It is an important step in the ML process because the quality and structure of the data can significantly impact the performance of the model[57].

There are several types of preprocessing that are commonly performed:

a. Data cleaning: This involves identifying and correcting or removing invalid or missing data.
b. Data transformation: This involves converting data into a different format, such as scaling numerical values or encoding categorical variables.
c. Data reduction: This involves reducing the complexity of the data, such as by selecting a subset of features or by combining multiple features into a single feature.
d. Data splitting: This involves dividing the data into training and test sets, which are used to train and evaluate the model, respectively.

Preprocessing can be a time-consuming operation, but it is necessary to guarantee that the data is in a format acceptable for the ML model. Proper preprocessing can increase model performance and lessen the danger of overfitting[57].

A one-hot encoder is a preprocessing approach in ML that is used to encode categorical variables into numerical data. Categorical variables have a restricted set of values, such as "red," "green," and "blue" in the case of a color variable. Because these variables are not numerical, they cannot be employed directly in most ML models [58].

One-hot encoding turns each category into a new column and assigns a binary value of 1 or 0 to indicate whether the category exists in the original data or not. If the color variable contains three categories (red, green, blue), for example, one-hot encoding will generate three additional columns, one for each category[59]. If a data point contains the color variable value "red," the "red" column will be 1, while the "green" and "blue" columns will be 0.

One-hot encoding is a popular strategy for dealing with categorical variables in ML since it allows the model to successfully learn and predict based on the categorical input. It is particularly handy when working with text data, as each word may be viewed as a new category[60].

It is crucial to note that if the categorical variable contains a significant number of categories, one-hot encoding can generate a big number of additional columns. This might result in the so-called "curse of dimensionality," in which the number of characteristics in the data significantly outnumbers the number of data points, lowering the model's performance. In these circumstances, a new encoding method or a reduction in the number of categories in the variable may be required[61].

3.6 FEATURE SELECTION

FS refers to the process of selecting a subset of relevant features for inclusion in an ML model [62].

FS is significant for various reasons:

a. Reducing Overfitting: Overfitting occurs when a classifier becomes excessively tailored to the training data, losing its ability to effectively generalize to new, unseen data. Including too many characteristics in a model can lead to overfitting. We can avoid overfitting and improve the model's generalization performance by picking a smaller collection of relevant characteristics.
b. Improving Model Interpretability: A model with a smaller number of features is easier

to interpret and understand than a model with a large number of features. This can be especially important when the results of the classifier need to be explained to a non-technical audience.

c. Reducing Computational Costs: It can be computationally costly to train and evaluate a model with a high number of characteristics. We may lower the computing expenses of training and assessing the classifier by picking a smaller collection of features.

There are numerous ways to choose features, including:

a. Filter techniques: These approaches utilize statistical measures to assess the significance of each characteristic and then pick a subset based on that assessment.

b. Wrapper approaches: These methods assess the significance of each feature based on the performance of an ML algorithm.

c. Embedded techniques: As part of the classifier training process, these methods learn the FS process.

The FS technique chosen is determined by the nature of the data and the project's aims. It's a good idea to test a few alternative strategies and compare how they perform on the task at hand [62].

3.7 MODEL EVALUATION

The process of evaluating the performance of a ML model on a set of data is known as model evaluation. It is a critical phase in the construction of an ML model since it determines how effectively the classifier can predict based on the input data[63].

A model may be evaluated in numerous ways, including ACC, PREC, and RE. the classifier's ACC is a measure of how often it accurately predicts the outcome. PREC is the frequency with which the classifier properly predicts the positive class, whereas RE is the frequency with which the classifier correctly recognizes all occurrences of the positive class[63].

Table 3.2: Confusion Matrix.

Total Instances	**Predicted No**	**Predicted Yes**	
Actual No	TN	FP	
Actual Yes	FN	TP	RE
		PREC	ACC

a. Confusion Matrix (CM): Calculating all of the performance measurements will be simple if we can develop a CM[64].

b. Precision : the number of sentences in the candidate and reference summaries divided by the number of sentences in the candidate summary, as shown in Eq. (4)

$$Precision = \frac{number\ of\ overlapping\ words}{total\ words\ in\ system\ summary} \tag{3.4}$$

c. Recall: refers to how much of the reference summary the system summary is recovering or capturing in the context of ROUGE, to put it simply. If we only take the words as a whole, we may calculate it as Eq. (5)

$$Recall = \frac{number\_of\_overlapping\_words}{tota\_words\_in\_referance\_summary} \tag{3.5}$$

d. F1-score: this measure integrates both the RE and PREC measures as given in the following Eq. (6)

$$F - Measure = \frac{2(precision)(recall)}{precision+recall} \tag{3.6}$$

The F1-S serves as a balanced metric combining PREC and RE, evaluating the classifier's proficiency in differentiating between positive and negative classes[64].

In addition to these metrics, it is important to consider the context in which the classifier will be used. For example, if the classifier is being used to predict medical diagnoses, it is important to consider the consequences of FPs and FNs. In this case, a model with a high PREC may be preferred over one with a high ACC [65].

Model evaluation also entails contrasting the performance of several models. This may be accomplished using cross-validation, in which the data is divided into sets for training and testing, and the classifier is trained and assessed on the training set. This ensures that the classifier is not overfitted to the training data and that it will generalize well to fresh data.

Model evaluation is a crucial step in the development of a ML model. It aids in determining the classifier 's performance and ensuring that it can generate correct predictions based on the input data.

4. EXPERIMENTAL RESULTS AND DISCUSSIONS

4.1 EXPERIMENTAL RESULT OF EACH MODEL

The data collection was divided to form the training and testing datasets. The training dataset comprised 80% of all samples, while the remaining 20% constituted the test dataset. In total, the dataset featured 45 columns.

4.2 TRAINING AND TESTING OF EACH ALGORITHM

All ML algorithms enhance their capabilities by analyzing and understanding relationships, gaining insights, making decisions, and boosting their confidence through the training data provided to them. Generally, the greater the volume of training data a model is exposed to, the more proficiently it tends to perform.

A ML model is tested on a test set after being trained on a training set. The test set should be utilized only when the classifier has been successfully trained using the training set, and it should normally contain only data from the training set.

It is usual practice to partition the data into two sets: training (80%) and testing (20%).

4.3 RESULTS

Table 4.1 table demonstrates the results of different ML models used to classify "User Click to ads" as either "no attack (0)" or "attack (1)". the classifier s listed are RF, DT, KNN, MLP, and ANN.

Table 4.1: The Training Time and Accuracy for Each Algorithm.

Model	Accuracy w/o scaling	Training time (second)
RF	97.79%	13.04
DT	96.59%	1.35
KNN	96.39%	0.02
MLP	95.05%	3.30
ANN	95.99%	0.03

4.3.1 Model Evolution

Evaluating the effectiveness of a ML model necessitates the use of certain metrics that vary depending on the job at hand (such as classification, regression, ranking, clustering, or topic modeling). Some measures, such as PREC - RE, are applicable to a wide range of activities, but others are more particular to specific sorts of jobs. Many ML applications, such as classification and regression, utilize guided learning tasks.

A CM is often utilized as a tabular representation to encapsulate the effectiveness of a classification model, particularly when applied to a test dataset with known actual values. The items in the table generally represent the number of predictions produced by the classifier for each combination of actual and projected classes.

In this case, the CM for the RF Model is as follows, As shown in table 4.2:

Table 4.2: CM for RF Model.

	Actual no attack (0)	Actual attack (1)
Predict no attack (0)	0.98 (TN)	0.021 (FN)
Predict attack (1)	0.025 (FP)	0.97 (TP)

True Positive (TP) is the number of times the model accurately predicted an assault (1).

True Negative (TN) is the number of times the classifier accurately predicted no assault (0).

The number of False Positives (FP) is the number of times the classifier predicted an attack (1) but there was no assault (0).

False Negative (FN) is the number of times the classifier predicted no assault while the actual attack was attack (1).

According to the CM, the RF Model has a high ACC, with 98% of TNs and 97% of TPs. It does, however, have 2.5% FNs and 2.1% FPs.

In the CM for the DT Model is as follows:

Table 4.3: CM for DT.

	Actual no attack (0)	Actual attack (1)
Predict no attack (0)	0.96 (TN)	0.039 (FN)
Predict attack (1)	0.032 (FP)	0.97 (TP)

From this CM, we can see that the DT Model has 96% of TNs and 97% of TPs. However, it also has a 3.9% of FNs and 3.2% of FPs. The balance between PREC, RE and ACC can be utilized to assess the performance of the classifier in this specific problem. Compared to RF Model, the DT Model has lower ACC, but higher FPR, which means that it is more likely to predict attack, when there is no attack.

In this case, the CM for the KNN Model is as follows, As illustrated in table 4.4:

Table 4.4: CM for KNN.

	Actual no attack (0)	Actual attack (1)
Predict no attack (0)	0.97 (TN)	0.034 (FN)
Predict attack (1)	0.041 (FP)	0.96 (TP)

According to this CM, the KNN Model contains 97% of TNs and 96% of TPs. It does, however, have 3.4% FNs and 4.1% FPs. When compared to the RF Model, the KNN Model has a lower ACC but a higher FPR, implying that it is more likely to predict an attack when there is none.

In this case, the CM for the MLP Model is as follows, As shown in table 4.5:

Table 4.5: CM for MLP.

	Actual no attack (0)	Actual attack (1)
Predict no attack (0)	0.968 (TN)	0.023 (FN)
Predict attack (1)	0.071 (FP)	0.93 (TP)

The CM indicates that the MLP Model comprises 96.8% TNs and 93% TPs, along with 2.3% FNs and 7.1% FPs. Compared to the RF Model, the MLP Model shows a decreased ACC and a heightened rate of FNs, implying it is more prone to mistakenly predicting the absence of an attack when one is actually occurring.

As indicated in table 4.6, the CM for the CNN Model in this scenario is as follows:

Table 4.6: CM for ANN.

	Actual no attack (0)	Actual attack (1)
Predict no attack (0)	0.96 (TN)	0.041 (FN)
Predict attack (1)	0.039 (FP)	0.96 (TP)

This CM reveals that the CNN Model registers 96% TNs and 96% TPs, alongside 4.1% FNs and 3.9% FPs. In comparison to the RF Model, the CNN Model demonstrates a reduced ACC but exhibits a higher FPR, indicating a greater tendency to predict an attack in the absence of one.

Table 4.2 appears to be showing the results of a ML model comparison for a specific task, which is to detect user click to ads attack. the classifier s listed are: RF, MLP, DT, ANN, and KNN. The columns display the efficiency of each model in terms of PREC, RE, and F1-S, for both classes "no attack (0)" and "attack (1)".

From the table, it appears that the RF model had the best overall performance in terms of F1-S, with an average of 97% for both classes. The MLP, DT, ANN and KNN models had slightly lower performance, with an average F1-S of 95% for both classes.

It is crucial to note that numerous metrics may be used to assess a model's performance, and the results of this comparison may vary based on the individual dataset and task being utilized; also, PREC, RE, and F1-S should be examined collectively to comprehend the

classifier 's performance.

Table 4.7: The Precision, Recall, and F1-Score for Each Algorithm.

algorithms	User Click to ads	precision	recall	F1-score
RF	no attack (0)	97%	98%	97%
	attack (1)	98%	97%	98%
MLP	no attack (0)	92%	98%	95%
	attack (1)	98%	93%	95%
DT	no attack (0)	96%	96%	96%
	attack (1)	97%	97%	97%
ANN	no attack (0)	96%	96%	96%
	attack (1)	97%	97%	97%
KNN	no attack (0)	95%	97%	96%
	attack (1)	97%	96%	97%

4.4 DISCUSSION

The results of our study on UP on IoT devices using machine and DL approaches suggest that both approaches can be effective in addressing UP on these devices. The highest ACC was achieved using the RF algorithm, with an ACC of 97.79%. This suggests that the RF algorithm may be the most effective for predicting and protecting UP on IoT devices.

The DT and KNN algorithms also had relatively high accuracies, with 96.59% and 96.39% respectively. These algorithms may also be suitable for addressing UP on IoT devices.

It's essential to note that the ACC of these algorithms can be influenced by a variety of factors, as a result, it might be worth considering other algorithms or fine-tuning the parameters of these algorithms to see if their performance can be improved.

The results of study suggest that both ML and DL approaches can be effective in addressing UP on IoT devices, with some algorithms being more effective than others. Further research and analysis may be necessary to determine the most effective approach for addressing UP on these devices.

5. CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The results of our research suggest that ML algorithms may be effectively utilized to boost UP on IoT devices. The RF algorithm, in particular, displayed the greatest ACC in predicting privacy-related activities on the device, followed by the DT, KNN, LR, ANN, and MLP algorithms.

However, it is important to note that the effectiveness of these algorithms may vary depending on the specific characteristics and requirements of the IoT device and the user data being processed. As a result, additional experiments and fine-tuning of the models may be required to achieve optimal performance for a given application.

In conclusion, the use of machine and DL approaches represents a promising direction for improving UP on IoT devices, and further research in this area is warranted.

5.2 FUTURE WORK

Further optimize the ML algorithms: There may be opportunities to further optimize the performance of the ML algorithms used in the study, such as by fine-tuning the model hyperparameters or incorporating additional features.

While this study focused on traditional ML algorithms, DL approaches may also be effective in predicting privacy-related actions on IoT devices. It would be fascinating to compare the performance of DL models to that of the study's classical ML techniques.

Examine the classifier s' robustness: It is critical to ensure that the models produced in this study are resilient and can generalize effectively to new data.

Future study might concentrate on assessing the classifier s' resilience and generalizability using approaches such as cross-validation and testing on unseen data.

Apply the models to real-world IoT devices: While this study focused on simulated data, it would be valuable to test the models on real-world IoT devices and user data to assess their practical utility in enhancing UP.

Investigate other approaches for improving UP on IoT devices: In addition to machine and DL approaches, there may be other methods for enhancing UP on IoT devices, such as through the use of encryption or other security measures. Further research could explore

these alternative approaches and compare their effectiveness to that of the ML methods used in this study.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning. nature, 521 (7553), 436-444," *Google Scholar Google Scholar Cross Ref Cross Ref*, p. 25, 2015.

[3] J. Wang, Y. Liu, S. Niu, and H. Song, "Extensive throughput enhancement for 5G-enabled UAV swarm networking," *IEEE Journal on Miniaturization for Air and Space Systems*, vol. 2, no. 4, pp. 199–208, 2021.

[4] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4, no. 4. Springer, 2006.

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

[6] M. Elkhodr, S. Shahrestani, and H. Cheung, "The internet of things: New interoperability, management and security challenges," *arXiv preprint arXiv:1604.04824*, 2016.

[7] M. A. Bouras, F. Farha, and H. Ning, "Convergence of computing, communication, and caching in Internet of Things," *Intelligent and Converged Networks*, vol. 1, no. 1, pp. 18–36, 2020.

[8] A. Marzano *et al.*, "The evolution of bashlite and mirai iot botnets," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, 2018, pp. 813–818.

[9] A. Aleesa, M. Younis, A. A. Mohammed, and N. Sahar, "Deep-intrusion detection system with enhanced UNSW-NB15 dataset based on deep learning techniques," *Journal of Engineering Science and Technology*, vol. 16, no. 1, pp. 711–727, 2021.

[10] S. Choudhary and N. Kesswani, "Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT," *Procedia Comput Sci*, vol. 167, pp. 1561–1573, 2020.

[11] G. Kocher and G. Kumar, "Performance analysis of machine learning classifiers for intrusion detection using unsw-nb15 dataset," *Comput. Sci. Inf. Technol.(CS IT)*, vol. 10, no. 20, pp. 31–40, 2020.

[12] S. Bagui, E. Kalaimannan, S. Bagui, D. Nandi, and A. Pinto, "Using machine

learning techniques to identify rare cyber-attacks on the UNSW-NB15 dataset," *Security and Privacy*, vol. 2, no. 6, p. e91, 2019.

[13] V. Kanimozhi and P. Jacob, "UNSW-NB15 dataset feature selection and network intrusion detection using deep learning," *Int. J. Recent Technol. Eng*, vol. 7, pp. 443–446, 2019.

[14] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019.

[15] S. Zheng, N. Apthorpe, M. Chetty, and N. Feamster, "User perceptions of smart home IoT privacy," *Proc ACM Hum Comput Interact*, vol. 2, no. CSCW, pp. 1–20, 2018.

[16] N. Waheed, X. He, M. Ikram, M. Usman, S. S. Hashmi, and M. Usman, "Security and privacy in IoT using machine learning and blockchain: Threats and countermeasures," *ACM Computing Surveys (CSUR)*, vol. 53, no. 6, pp. 1–37, 2020.

[17] B. Zhu *et al.*, "Iot equipment monitoring system based on c5. 0 decision tree and time-series analysis," *IEEE Access*, vol. 10, pp. 36637–36648, 2021.

[18] Y. He, P. Bahirat, B. P. Knijnenburg, and A. Menon, "A data-driven approach to designing for privacy in household iot," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 10, no. 1, pp. 1–47, 2019.

[19] C. Dunn, N. Moustafa, and B. Turnbull, "Robustness evaluations of sustainable machine learning models against data poisoning attacks in the internet of things," *Sustainability*, vol. 12, no. 16, p. 6434, 2020.

[20] J. Fakirah, L. M. Zishan, R. Mooruth, M. N. Johnstone, and W. Yang, "A low-cost machine learning based network intrusion detection system with data privacy preservation," *arXiv preprint arXiv:2107.02362*, 2021.

[21] E. Buber, B. Diri, and O. K. Sahingoz, "NLP based phishing attack detection from URLs," in *Intelligent Systems Design and Applications: 17th International Conference on Intelligent Systems Design and Applications (ISDA 2017) held in Delhi, India, December 14-16, 2017*, 2018, pp. 608–618.

[22] M. Almousa, T. Zhang, A. Sarrafzadeh, and M. Anwar, "Phishing website detection: How effective are deep learning-based models and hyperparameter optimization?," *Security and Privacy*, vol. 5, no. 6, p. e256, 2022.

[23] A. Basit, M. Zafar, A. R. Javed, and Z. Jalil, “A novel ensemble machine learning method to detect phishing attack,” in *2020 IEEE 23rd International Multitopic Conference (INMIC)*, 2020, pp. 1–5.

[24] “UNSW_NB15.,” 2023. https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15. (accessed Jan. 27, 2023).

[25] [25] A. C. Müller and S. Guido, *Introduction to machine learning with Python: a guide for data scientists*. “ O’Reilly Media, Inc.,” 2016.

[26] R. Garreta and G. Moncecchi, *Learning scikit-learn: machine learning in python*. Packt Publishing Ltd, 2013.

[27] N. Farnaaz and M. A. Jabbar, “Random forest modeling for network intrusion detection system,” *Procedia Comput Sci*, vol. 89, pp. 213–217, 2016.

[28] P. A. A. Resende and A. C. Drummond, “A survey of random forest based methods for intrusion detection systems,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–36, 2018.

[29] W. Lin, Z. Wu, L. Lin, A. Wen, and J. Li, “An ensemble random forest algorithm for insurance big data analysis,” *Ieee access*, vol. 5, pp. 16568–16575, 2017.

[30] C. Vens and F. Costa, “Random forest based feature induction,” in *2011 IEEE 11th international conference on data mining*, 2011, pp. 744–753.

[31] G. Louppe, “Understanding random forests: From theory to practice,” *arXiv preprint arXiv:1407.7502*, 2014.

[32] P. Misra and A. S. Yadav, “Impact of preprocessing methods on healthcare predictions,” in *Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE)*, 2019.

[33] M. Idhammad, K. Afdel, and M. Belouch, “Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest,” *Security and Communication Networks*, vol. 2018, 2018.

[34] T. M. Deist *et al.*, “Machine learning algorithms for outcome prediction in (chemo) radiotherapy: An empirical comparison of classifiers,” *Med Phys*, vol. 45, no. 7, pp. 3449–3459, 2018.

[35] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, “KNN model-based approach in classification,” in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and*

ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings, 2003, pp. 986–996.

[36] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit*, vol. 40, no. 7, pp. 2038–2048, 2007.

[37] S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, "Learning k for knn classification," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 3, pp. 1–19, 2017.

[38] M. D. Malkauthekar, "Analysis of Euclidean distance and Manhattan distance measure in Face recognition," in *Third International Conference on Computational Intelligence and Information Technology (CIIT 2013)*, 2013, pp. 503–507.

[39] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, "Efficient kNN classification algorithm for big data," *Neurocomputing*, vol. 195, pp. 143–148, 2016.

[40] L. Li, C. R. Weinberg, T. A. Darden, and L. G. Pedersen, "Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method," *Bioinformatics*, vol. 17, no. 12, pp. 1131–1142, 2001.

[41] Y. Wu, K. Ianakiev, and V. Govindaraju, "Improved k-nearest neighbor classification," *Pattern Recognit*, vol. 35, no. 10, pp. 2311–2318, 2002.

[42] Y. Park, S. Park, W. Jung, and S. Lee, "Reversed CF: A fast collaborative filtering algorithm using a k-nearest neighbor graph," *Expert Syst Appl*, vol. 42, no. 8, pp. 4022–4028, 2015.

[43] B. Charbuty and A. Abdulazeez, "Classification based on decision tree algorithm for machine learning," *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 20–28, 2021.

[44] J. Su and H. Zhang, "A fast decision tree learning algorithm," in *Aaai*, 2006, vol. 6, pp. 500–505.

[45] Y. Ben-Haim and E. Tom-Tov, "A Streaming Parallel Decision Tree Algorithm.," *Journal of Machine Learning Research*, vol. 11, no. 2, 2010.

[46] Y.-Q. Liu, C. Wang, and L. Zhang, "Decision tree based predictive models for breast cancer survivability on imbalanced data," in *2009 3rd international conference on bioinformatics and biomedical engineering*, 2009, pp. 1–4.

[47] M. Sebbanü, R. NockO, J. Chauchat, and R. Rakotomalala, "Impact of learning set

quality and size on decision tree performances," *IJCSS*, vol. 1, no. 1, p. 85, 2000.

[48] D. Che, Q. Liu, K. Rasheed, and X. Tao, "Decision tree and ensemble learning algorithms with their applications in bioinformatics," *Software tools and algorithms for biological systems*, pp. 191–199, 2011.

[49] M. Riedmiller and A. Lernen, "Multi layer perceptron," *Machine Learning Lab Special Lecture, University of Freiburg*, pp. 7–24, 2014.

[50] S. Mitra and S. K. Pal, "Fuzzy multi-layer perceptron, inferencing and rule generation," *IEEE Trans Neural Netw*, vol. 6, no. 1, pp. 51–63, 1995.

[51] F. Rossi and B. Conan-Guez, "Functional multi-layer perceptron: a non-linear tool for functional data analysis," *Neural networks*, vol. 18, no. 1, pp. 45–60, 2005.

[52] M. Gustineli, "A survey on recently proposed activation functions for Deep Learning," *arXiv preprint arXiv:2204.02921*, 2022.

[53] M. P. Mattson, "Superior pattern processing is the essence of the evolved human brain," *Front Neurosci*, p. 265, 2014.

[54] I. Nusrat and S.-B. Jang, "A comparison of regularization techniques in deep neural networks," *Symmetry (Basel)*, vol. 10, no. 11, p. 648, 2018.

[55] B. Marr, *Artificial intelligence in practice: how 50 successful companies used AI and machine learning to solve problems*. John Wiley & Sons, 2019.

[56] D. Garg and M. Alam, "Deep learning and IoT for agricultural applications," *Internet of Things (IoT) Concepts and Applications*, pp. 273–284, 2020.

[57] E. Quiring, D. Klein, D. Arp, M. Johns, and K. Rieck, "Adversarial preprocessing: Understanding and preventing image-scaling attacks in machine learning," in *Proceedings of the 29th USENIX Conference on Security Symposium*, 2020, pp. 1363–1380.

[58] P. Rodríguez, M. A. Bautista, J. Gonzalez, and S. Escalera, "Beyond one-hot encoding: Lower dimensional target embedding," *Image Vis Comput*, vol. 75, pp. 21–31, 2018.

[59] I. Ul Haq, I. Gondal, P. Vamplew, and S. Brown, "Categorical features transformation with compact one-hot encoder for fraud detection in distributed environment," in *Data Mining: 16th Australasian Conference, AusDM 2018, Bahrurst, NSW, Australia, November 28–30, 2018, Revised Selected Papers 16*, 2019, pp. 69–80.

[60] S. Okada, M. Ohzeki, and S. Taguchi, “Efficient partition of integer optimization problems with one-hot encoding,” *Sci Rep*, vol. 9, no. 1, p. 13036, 2019.

[61] A. C. H. Choong and N. K. Lee, “Evaluation of convolutionary neural networks modeling of DNA sequences using ordinal versus one-hot encoding method,” in *2017 International Conference on Computer and Drone Applications (IConDA)*, 2017, pp. 60–65.

[62] J. Tang, S. Alelyani, and H. Liu, “Feature selection for classification: A review,” *Data classification: Algorithms and applications*, p. 37, 2014.

[63] B. J. McAvaney *et al.*, “Model evaluation,” in *Climate Change 2001: The scientific basis. Contribution of WG1 to the Third Assessment Report of the IPCC (TAR)*, Cambridge University Press, 2001, pp. 471–523.

[64] R. Yacouby and D. Axman, “Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models,” in *Proceedings of the first workshop on evaluation and comparison of NLP systems*, 2020, pp. 79–91.

[65] D. Chicco and G. Jurman, “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation,” *BMC Genomics*, vol. 21, pp. 1–13, 2020.