LUND UNIVERSITY
School of Economics and Management

Master's Programme in Data Analytics and Business Economics

Analysis and Prediction of Cyber-Threats using Machine Learning algorithms

A Masters Thesis submitted to Lund University School of Economics and Management (LUSEM)

by

Vinh Nguyen
Victor Kebande

DABN01
Master's Thesis (15 credits ECTS)
August, 2024
Supervisor: Farrukh Javed

Abstract

As digital transformation continues to expand, the threat landscape for cyber-attacks has become increasingly complex. Proactive detection and prevention of cyber-threats are essential to secure critical infrastructures. This thesis proposes an approach centered on leveraging data analysis and Machine Learning (ML) to analyze and predict the existence of potential cyber-threats. By harnessing the power of advanced analytics techniques, and predictive modeling, the study aims to uncover hidden insights from historical cyber-threat data. This study has leveraged Microsoft Malware Prediction Dataset that has been curated by Microsoft for the purpose of advancing research in malware detection and prediction using ML classification models. We then evaluate and compare the effectiveness of several classifiers, including Random Forest (RF), Logistic Regression (LR), Quadratic Discriminant Analysis (QDA), Linear Discriminant Analysis (LDA), XGBoost, and a Fully Connected Neural Network (FCNN). RF achieved an average accuracy of 62.24%, LR 61.41%, QDA model achieved an accuracy of 59.15%, LDA 60.86%, XGBoost 60.75% and FCNN 63.05% respectively. From these results, it is evident that Ensemble methods like RF and Deep learning techniques like Neural Nets outperformed other models in terms of accuracy and other evaluation metrics that perform consistently well in new, unseen data. LR and LDA also demonstrated competitive performance. These findings suggest that RF and FCNN techniques are promising approaches for cyber threat detection tasks.

Acknowledgements

We would like to thank our families, friends, our supervisor, examiner and Lund University School of Economics and Management (LUSEM), Sweden at large for the support.

Contents

List of Figures

List of Tables

1 Introduction

Technological advancements have fundamentally transformed the way organizations operate. It has been witnessed that digital transformation has led to pervasive integration of technology across industries, enabling automation, real-time decision-making, and enhanced connectivity. Consequently, organizations are increasing their dependence on technology to operate efficiently and profitably as many systems witness constant digitization. However, this digitization has also led to a complex and evolving landscape of cyberthreats that pose significant risks to the security and confidentiality of information systems. Cyber-attacks, such as data breaches or sophisticated malware intrusion, have become more frequent and severe, leading to financial losses, reputational damage, and regulatory challenges for businesses across various sectors.

Given the increased complexity, and persistence of cyber threats, it has been seen that traditional security approaches can hardly keep up with the advanced pace of attackers, where there is unpredictability of known and unknown vulnerabilities and attacks. Cyber threat in the context of this thesis is represented as a malicious action or event that has a potential of posing a risk to the security goals; Confidentiality, Integrity and Availability (CIA) of a system.

Nevertheless, it has been observed that the traditional security methods often struggle to keep pace with the dynamic nature of cyber threats, necessitating the exploration of advanced methodologies for threat analysis and prediction [MLA+24]. Looking at the above-mentionedchallenges, the cybersecurity community has opted to turn to advanced techniques and technologies for purposes threat analysis, detection and mitigation. In this context, Machine Learning (ML) has emerged as a powerful approach, offering the ability to analyze vast amounts of data, identify key factors, and make predictions based on historical trends. This is owing to the capabilities it has of processing huge volumes of data and ability to uncove hidden patterns, and also to predicting future events based on historical trends.

This thesis focuses on the analysis and prediction of cyber threats utilizing ML algorithms. By leveraging advanced data analytics techniques, this study aims to develop predictive models that enhance the detection of cyber threats. The study seeks to address critical questions regarding the effectiveness of different ML algorithms in identifying malicious activities and the possibility of applying the proposed approach in a practical scenario.

The remainder of this chapter is structured as follows: Section 1.1 introduces the research questions that guide this study, while Section 1.2 outlines the primary objectives. Section 1.3 presents the methodology adopted in this research, and Section 1.4 highlights the key contributions of this work. Finally, Section 1.5 provides an overview of the thesis layout, detailing its organization and structure.

1.1 Research Questions

The following research questions have been used to guide this study toward cyber-resilience in against cyber-threats.

1. **RQ 1:** What machine learning methods can be leveraged effectively to analyze and predict cyber-threat?
2. **RQ 2:** Can these machine learning algorithms perform well in new, unseen cyber-threats?
3. **RQ 3:** What are the key features and their importance in enhancing the prediction and analysis of cyber-threats?

1.2 Research aims and Objectives

The overaching aim of this study is to be able to develop proactive techniques that can be used to analyze and forecast cyber-threats. The following objectives will be achieved in this thesis.

- **Obj 1:** To explore the use of data analysis and machine learning techniques for analyzing cyber-threat data
- **Obj 2:** To develop predictive models for anticipating emerging cyber-threats, evaluate and compare their accuracy and reliability.
- **Obj 3:** To assess the performance of these models to new, unseen data split, ensuring they perform well in real-world scenarios.

1.3 Methodology

This thesis adopts a Design Science Research (DSR) approach [PTRC07], as illustrated in Figure 1.1. The process begins with identifying the problem of cyber-threat analysis and prediction, which serves as the foundation for addressing the research problem. A comprehensive review of relevant literature is conducted to gain a deeper understanding of the problem domain. Based on this understanding, specific research objectives are defined. Subsequently, techniques are designed and developed to address these objectives. Machine Learning (ML) algorithms are then applied to demonstrate the analysis and prediction of cyber threats. Finally, the proposed contributions are evaluated, and the outcomes are presented.

1.4 Contributions

The main contribution of this thesis lies in the techniques of analysis and prediction approaches to cyber-threat detection and can be summarized as follows:

- The thesis takes a hybrid approach by combining classical and non classical ML models to develop predictive approaches aimed at identifying cyber-threats

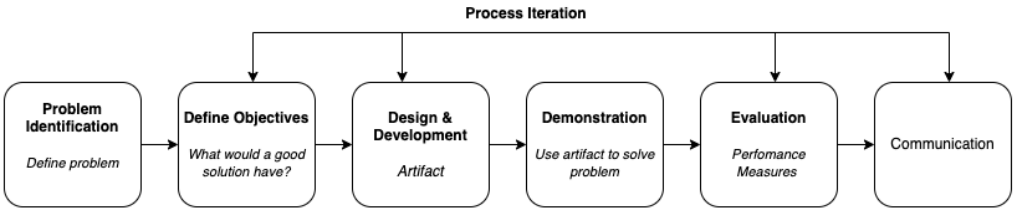

Figure 1.1: DSR Methodology used in this thesis

- The thesis demonstrates the feasibility and effectiveness of leveraging data-driven approaches to cyber-threat analysis and prediction.
- The thesis gives a contribution to the field of cybersecurity and data analytics by offering innovative ML-based solutions to enhance the analysis and prediction of cyber-threats.

1.5 Thesis Layout

This thesis comprises seven sections addressing various aspects of proactive cybersecurity strategies . **Section 1** introduces the research topic, outlines the problem statement, research questions, and objectives. **Section 2** delves into the theoretical background of cybersecurity and data analytics. **Section 3** reviews the related literature on proactive cybersecurity strategies and cyber-threat analysis. **Section 4** details the research methodology, including data collection, analysis techniques, and empirical research methods. **Section 5** presents the study evaluation. Finally, **Section 6** concludes the thesis, summarizing key findings, discussing implications, and offering recommendations for future research, thus providing a comprehensive framework for understanding and addressing cybersecurity issues.

2 Theoretical Background

In order to understand the complexity of the cybersecurity landscape, we explore the theoretical background. We concentrate on providing concepts and theories that underpin the analysis that is conducted for proactive cybersecurity strategies. We argue that by examining the theoretical foundations that cut across cybersecurity and critical infrastructure protection, it becomes clear to be able to establish a framework that guides in understanding the challenges and opportunities that are inherent in providing defense or safeguarding a critical infrastructure against cyber-threats.

This section, therefore, sets the stage for the subsequent discussions on the ever-evolving threat and attack landscape, the vulnerabilities in critical infrastructures and the major theoretical underpinnings of proactive cybersecurity. Based on this exploration, we seek to highlight the basis of our research and lay the groundwork for a more comprehensive understanding of proactive cyber threat detection.

2.1 Threats in Critical Infrastructures

Critical infrastructures, are usually important and susceptible to a wide range of vulnerabilities. These vulnerabilities are able to be exploited by malicious actors in order to disrupt operations, compromise the integrity of data, and also undermine trust [RTP+23]. These threats are attributed to various sources like technological dependencies, human factors, and external threats.

2.1.1 Taxonomy of Cyber Threats

One identifiable major threats in critical infrastructures is the reliance on interconnected digital technologies and networked systems [Uli07, Hel07]. The interconnected nature of critical infrastructure networks increases the chances of having potential cascading failures, and systemic disruptions in the event of a cyber-attack or if there is a fault or infrastructure failure [PAMM21]. In addition, the vulnerabilities in software applications, operating systems, and network protocols can be exploited by threat actors to gain unauthorized access, escalate privileges, or launch denial-of-service attacks.

Moreover, human factors such as insider threats, human error, and lack of cybersecurity awareness contribute to vulnerabilities in critical infrastructures as shown in Figure 2.1.Figure 2.1 shows selected general cyber threat classification that are prevalent in critical infrastructures. This include; malware, Phishing, Social Engineering, Insider threats and Denial of Service (DoS). The Malware classification

includes threats like Ransomware, Viruses, Spyware, Worms, and Trojans, which exploit vulnerabilities in systems. Next, the Phishing targets human deception with tactics such as Email Phishing, Spear Phishing, Whaling, and Vishing. This is followed by Social Engineering threats that manipulates human behavior through methods like Pretexting, Baiting, and Impersonation. Insider Threats highlight risks from Malicious, Negligent, and Compromised Insiders, emphasizing internal vulnerabilities. According to [NBL+14], Insider threats may arise from disgruntled employees, or through negligent behavior, or malicious intent. This poses a significant risk to the security and integrity of critical infrastructure systems. Human error, including misconfigurations, improper handling of sensitive information, and failure to adhere to security protocols, can inadvertently expose vulnerabilities and compromise system integrity, as shown in Figure 2.1, [AHT17]. Lastly, DoS and DDoS attacks disrupt service availability through volumetric, protocol, and application-layer attacks. This structured taxonomy that is shown in Figure 2.1 provides a comprehensive framework for understanding and mitigating cyber threats.

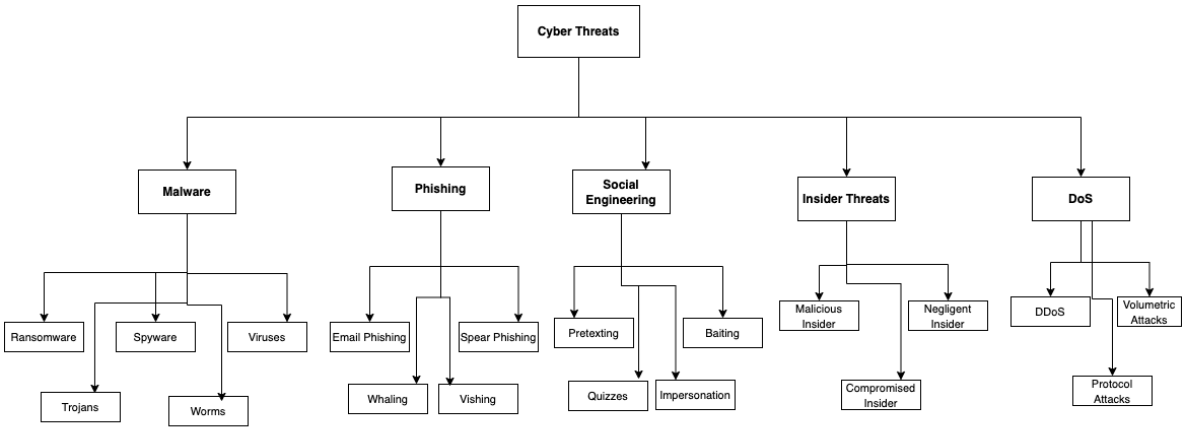

Figure 2.1: A Taxonomy of Vulnerabilities in Critical Infrastructures

2.1.2 External Threats

Other threats that are not covered in this taxonomy include the external threats, including cyber-attacks, natural disasters, and geopolitical events, also pose significant risks to critical infrastructures. Natural disasters, such as earthquakes, hurricanes, and floods, can damage physical infrastructure and disrupt essential services. Geopolitical events, such as geopolitical tensions, trade disputes, and regulatory changes, can impact critical infrastructure operations and resilience [RM18].

2.2 Machine Learning

Machine Learning (ML) as a field of study allows computers and systems to learn from data and to be able make predictions or decisions without having to be explicitly programmed [Sar21]. This section explores the foundations of training a ML model.

2.3 Machine Learning Models

ML models are an array of algorithms and techniques that are designed to extract patterns from data in order to make informed predictions or decisions [LRM20]. These ML models are broadly categorized into several types, and each has its own unique characteristics and applications. In the context of this study we will explore ML classification models in different methods, includes Random Forest, Logistic regression, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and Fully-Connected Neural Networks (FCNN) model

2.3.1 Ensemble Models

Considered to be powerful ML techniques, ensemble methods are able to combine the prediction of multiple models in order to improve performance and robustness [DQRT15]. The ensemble is able to achieve better predictive accuracy and good generalization ability when compared to individual models. Therefore, in this section, we explore Random Forest (RF) and XGBoost, given that they are widely used in the context of cyber detection.

2.3.1.1 Random Forests

Random Forest (RF) is a commonly used ensemble learning method that is able to constructs multiple decision trees during training [BHA09]. It also outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. RF usually introduces bagging or (Bootstrap Aggregating) and random feature selection that allows diversity among the trees in order to reduce overfitting [PIL06].

Bagging allows the creation multiple subsets of the training data by sampling, and each of the subset is used in training a separate decision tree. [BHBK06]. Given a training set;

$$D = \{(x_i, y_i)\}_{i=1}^{n} \tag{2.1}$$

Bagging will then generates B bootstrap samples D_b for $b = 1, 2, \ldots, B$. After this, the random feature selection is then introduced and applied when splitting for each node in a tree [SMT09]. In this context, instead of considering all p features, a random subset of m features is chosen (where $m < p$). This is done in order to help in the reduction of the correlation among the trees. Next, the best split will be determined determined based on these m features. For classification, $m = \sqrt{p}$, and for regression, $m = \frac{p}{3}$.

In order to predict, the RF usually aggregates the predictions of all the individual trees [YHV+11]. In classification, the final prediction $\hat{y}$ is obtained by majority voting [BS16]

$$\hat{y} = \text{mode}\{T_b(x)\}_{b=1}^{B} \tag{2.2}$$

where $\hat{y}$ represents the final predicted class, $T_b(x)$ is the predicted class from the b-th tree for the input x, and B is the total number of trees in the forest. Consequently, in regression, the final prediction $\hat{y}$ is the average of the predictions given as

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} T_b(x) \tag{2.3}$$

Here, $\hat{y}$ represents the final predicted value, $T_b(x)$ is the predicted value from the b-th tree for the input x, and B is the total number of trees in the forest. The averaging process ensures that the final prediction is the mean of all the predictions from the trees, thereby providing a more stable and accurate prediction.

RF primarily is able to handle large datasets with higher dimensionality. It also provides a measure of feature importance has been explored in the later sections of this research thesis. Ultimately, this portrays its ability of being robust to overfitting. It however may require significant computational resources, and it also has lower interpretability compared to a single decision trees.

2.3.1.2 XGBoost

Extreme Gradient Boosting (XGBoost), is a high-performance implementation of the gradient boosting algorithm, which is normally considered for the speed and accuracy [CHB+15]. It is able to combine multiple base models, in order to enhance predictive performance. In addition, boosting builds models sequentially and each model attempts to correct errors made by its predecessor [SZZZ19]. XGBoost minimizes a loss function $\mathcal{L}(\theta)$ that is defined as is shown in Equation 2.9:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2.4}$$

where $L(y_i, \hat{y}_i)$ measures the model's prediction error, and $\Omega(f_k)$ is a regularization term that penalizes model complexity (Equation 2.10):

$$\Omega(f_k) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \tag{2.5}$$

T is the number of leaves, w_j is the weight on leaf j, γ controls the number of leaves, and λ controls L_2 regularization. Each of the trees in XGBoost is optimized using gradients and Hessians of the loss function [FCT20]. The optimal leaf weights are given by (Equation 2.11):

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{2.6}$$

where g_i and h_i are the gradient and Hessian, respectively, and I_j is the set of instances in leaf j.

The strength of XGBoost is in its ability to combine multiple weak learners to create a powerful model, with a focus on optimization and regularization.

2.3.2 Logistic Regression

Logistic Regression (LR) is a statistical method used for binary classification problems. The goal is to predict the probability that an input belongs to a particular

class [CVA06]. Unlike linear regression, which predicts continuous values, logistic regression predicts probabilities using a logistic function, also known as the sigmoid function.

The logistic regression model is defined as shown in Equation 2.7:

$$P(y = 1|\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}} \tag{2.7}$$

In Equation 2.7 $\mathbf{x}$ represents the input features, $\boldsymbol{\theta}$ is the parameter vector, and y is the binary target variable. The logistic function maps the linear combination of input features and parameters to the range [0, 1], representing the probability of the positive class.

The parameters $\boldsymbol{\theta}$ are learned from the training data by maximizing the likelihood of observing the training labels given the input features. This optimization problem is typically solved using techniques such as gradient descent or Newton's method.

Consequently, LR is widely used in various fields like medicine, finance, and marketing, for tasks such as disease diagnosis, credit scoring, and customer churn prediction [NGK+06].

2.3.3 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a classification algorithm used to find a linear combination of features that best separates two or more classes [BG98]. LDA assumes that the data are normally distributed and that the classes have the same covariance matrix. Also, LDA aims to maximize the separability between classes while minimizing the variance within each class [BG98].

To represent LDA, Let $\mathbf{x}_i$ be a d-dimensional feature vector representing the i-th sample, and let y_i be the corresponding class label, where $i = 1, 2, \ldots, N$. The goal of LDA is to find a linear discriminant function $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ that optimally separates the classes.

To derive the linear discriminant function, LDA starts by computing the class means $\boldsymbol{\mu}_k$ and the within-class scatter matrix $\mathbf{S}_W$:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathbf{x}_i^{(k)} \tag{2.8}$$

$$\mathbf{S}_W = \sum_{k=1}^{K} \sum_{i=1}^{N_k} (\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_k)(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}_k)^T \tag{2.9}$$

where N_k is the number of samples in class k, and $\mathbf{x}_i^{(k)}$ is the i-th sample in class k.

Next, LDA computes the between-class scatter matrix $\mathbf{S}_B$ as is shown in Equation 2.10:

$$\mathbf{S}_B = \sum_{k=1}^{K} N_k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T \tag{2.10}$$

where $\boldsymbol{\mu}$ is the overall mean of the data.

The linear discriminant function coefficients $\mathbf{w}$ are then obtained by solving the generalized eigenvalue problem as is shown in Equation 2.11:

$$\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} = \lambda\mathbf{w} \tag{2.11}$$

where λ is the eigenvalue associated with $\mathbf{w}$.

It is the observed ultimately that, the decision boundary is defined as the hyperplane orthogonal to $\mathbf{w}$, given by equation 2.12:

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{2.12}$$

where b is a bias term.

Based on the aforementioned processes, it is worth mentioning that LDA is a powerful technique for linear classification and dimensionality reduction, with applications in various fields such as pattern recognition, image processing, and bioinformatics [TSYQ05].

2.3.4 Quadratic Discriminant Analysis (QDA)

Quadratic Discriminant Analysis (QDA) is a classification algorithm that, unlike LDA discussed in Section 2.6.3, allows for different covariance matrices for each class [DB09]. This means that QDA does not assume equal covariance matrices across classes and can capture more complex decision boundaries.

Let $\mathbf{x}_i$ be a d-dimensional feature vector representing the i-th sample, and let y_i be the corresponding class label, where $i = 1, 2, \ldots, N$. The goal of QDA is to find a discriminant function $g_k(\mathbf{x})$ for each class k, such that the probability of $\mathbf{x}$ belonging to class k is maximized.

Similar to LDA, QDA starts by computing the class means $\boldsymbol{\mu}_k$ for each class k, and the covariance matrix $\boldsymbol{\Sigma}_k$ for each class k.

The discriminant function for class k is then given as is shown in Equation 2.13:

$$g_k(\mathbf{x}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}_k| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) + \log P(y=k) \tag{2.13}$$

where $|\boldsymbol{\Sigma}_k|$ denotes the determinant of the covariance matrix $\boldsymbol{\Sigma}_k$, and $P(y=k)$ is the prior probability of class k.

The decision boundary between classes k and l is determined by Equation 2.14:

$$g_k(\mathbf{x}) - g_l(\mathbf{x}) = 0 \tag{2.14}$$

which represents a quadratic function of $\mathbf{x}$.

QDA is particularly useful when the classes have different covariance matrices, allowing it to capture more complex decision boundaries compared to LDA. However, QDA may be prone to overfitting, especially when the number of features is large relative to the number of samples.

2.3.5 Fully Connected Neural Networks (FCNN)

A Fully Connected Neural Networks (FCNNs), also known as dense or Feedforward Neural Networks (FNN), are the foundation of deep learning models [SVSS15]. A

FCNN consists of multiple layers of neurons, where each neuron in one layer is connected to every neuron in the next layer.

Considering a fully connected neural network with L layers, including an input layer, one or more hidden layers, and an output layer. For simplicity, let $x^{(i)}$ denote the input vector to the network, where i represents the i-th training example.

The computation in a fully connected layer can be described as is shown in Equation 2.15:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \quad \text{for } l = 1, 2, \ldots, L \tag{2.15}$$

where $\mathbf{a}^{(l-1)}$ is the activation vector from the previous layer, $\mathbf{W}^{(l)}$ is the weight matrix connecting layer $l-1$ to layer l, $\mathbf{b}^{(l)}$ is the bias vector for layer l, and $\mathbf{z}^{(l)}$ is the input to the activation function in layer l.

The activation function $\sigma(\cdot)$ introduces nonlinearity into the network and allows it to learn complex patterns. Common choices for activation functions include the sigmoid function, hyperbolic tangent (tanh), and rectified linear unit (ReLU).

The output of the activation function in layer l is denoted as $\mathbf{a}^{(l)} = \sigma(\mathbf{z}^{(l)})$.

The final output of the neural network, denoted as $\hat{y}$, is computed using the activations of the last layer as is shown in Equation 2.16:

$$\hat{y} = \text{softmax}(\mathbf{z}^{(L)}) \tag{2.16}$$

where $\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ for multiclass classification problems with K classes.

Training a FCNN involves optimizing the network parameters $\{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}$ with respect to a chosen loss function, often using backpropagation and gradient descent-based optimization algorithms.

3 Related Work

The importance of prediction in cyberspace has become prevalent owing to the fact that the digital landscape continues to evolve and the threat landscape keeps changing. Predictive analytics, which in essence is channeled by machine learning and big data analytics plays a significant role of mitigating and managing cyber threats. In this section, we explore the significance of predictive analysis in cybersecurity and we highlight relevant works and a summary of the methods tha were utilize, datasets and the key findings from the respective works have been summarized in 3.1.

3.1 Predictive Analytics in Cyberspace

According to [NK21], in the quest to safeguard digital assets, effective threat detection in cyberspace is important, especially when machine learning and big data analytics are leveraged [HMH+17]. Also, [Sea22] highlights that predicting and modeling threats forms part of the important risk management approaches within any organization.

That notwithstanding, it has been seen that Swedish financial sectors especially rely heavily on information systems, and there is a need to bring cyber-threat awareness. For example, research by [VBF21] has explored a survey of key actors and has implications like reputation loss, financial theft, service availability, and data confidentiality if not proactively addressed.

The cyber supply chain has been positioned as a complex system with diverse sub-systems with many tasks and research has shown that there exist inherent vulnerabilities and threats.[YOIL+21] have shown an approach of leveraging machine learning and cyber threat intelligence to analyse and predict threats based on threat intelligence properties where they applied logistic regression, support vector machine, Random Forest and decision trees to model a predictive analytics approach using Microsoft-based malware Prediction dataset.

3.2 Cyber Threat Hunting

Research by Gao [GSL+21] has leveraged a framework called THREATRAPTOR that is able to collect system-level logs from an Operating System (OS) kernel. These logs had system-based events that were able to give interactions between entities. The THREATRAPTOR has been able to facilitate cyber-threat hunting in computer systems. Also, a data- driven approach for threat hunting using Systmon has shown the essence of automated threat assessment by continuously assessing

logs. This has shown threat-hunting approaches where logs are classified based on different threat levels and potential characteristics, [MJ18].

An ensemble deep learning model focused on cyber threat hunting in industrial IoT has been able to extract a balanced data from an imbalanced dataset where data is fed into the e Long Short-Term Memory (LSTM) and the AutoEncoder (AE) architecture for purposes of identifying anomalies. The outcomes are compared with the convention ML-classifiers; Multi-Layer Perceptron (MLP), Decision Tree (DT), and Super Vector Machines (SVM) and Random Forest (RF) and an accuracy is achieved between 99.3 % and 99.7% respectively, [YKP+23].

In another research, a cyberthreat-detector for unsupervised hunting of anomalous commands (UHAC) proposed an approach that is able to create a feature set and document character matrices. In addition, an auto-encoder that acted as a detector was trained using a custom loss function. It was then observed that this algorithm (feature set) was able to outperform one-class SVM while the UHAC detector was able to identify 84-89% of the anomalies that comprised of 10% of the actual data. From this study, it was evident that there were implications for cyber-security threat hunters, [KAS23].

Another approach for cyber-threat hunting that has a focus on the Kernel audit records has shown how to find information that is related to attack events in a quicker way from large data streams. This approach utilized the construction of knowledge graphs that was able to show correlations among the entities in the audit logs. the constructed knowledge graphs are used in hunting real threats according to [YHD+22].

3.3 Machine Learning for Malware Detection

The proliferation of malware has continuously been seen to pose a significant threat to cybersecurity.It has been observed that to counter this threat, machine learning techniques offer a promising approach. Therefore, in this section, we explore works on machine learning for malware detection in order to explore how these techniques leverage data-driven methodologies to identify and classify malicious software instances.

A study leveraging ML for the detection and classification of malware shows diverse methods and features in traditional ML and deep learning methods. It shows classifiers that rely on more than one feature or modality in detecting malware [GMP20]. A study by [VAS+19] has shown an intelligent malware detection approach that leverages deep learning where classical ML algorithms and deep learning architectures for malware detection are evaluated, classified and categorized with the study being able to act on zero-day malware. Consequently, other-relevant malware detection using machine-learning approach has utilized a Deep learning framework that leverages AutoEncoders with Boltzman machines that actively detect unknown malware [YHD+22]. In addition, malware classifier approaches using Deep Convolution Neural Network (CNN) leveraged a CNN-based architecture that was able to classify malware samples. It utilizes MAlimg and Microsoft Malware-this method was able to achieve better performance than the state-of-the art with 98.5% and 99.97% on accuracy on the two datasets respectively [KRM+18].

In spite of this, other relevant studies have focused on detecting Android malware, where malware has been classified using a deep learning framework referred to

Table 3.1: Summary of Related Work in Predictive Analytics and Cybersecurity

Category	Methods/Models	Datasets	Key Findings
Predictive Analytics in Cyberspace	Machine learning (e.g., Logistic Regression, SVM, Random Forest, Decision Trees) [YOIL+21]	Microsoft Malware Prediction	Leveraged ML and threat intelligence properties to model predictive analytics for cyber threats.
Cyber Threat Hunting	THREATRAPTOR framework for log collection and analysis [GSL+21]	System-level logs	Facilitated efficient cyber-threat hunting by analyzing interactions between system entities.
Cyber Threat Hunting	Knowledge graphs for analyzing audit records [YHD+22]	Audit logs	Demonstrated efficient attack event detection through correlations in large data streams.
Cyber Threat Hunting	AutoEncoder with custom loss function [KAS23]	Custom dataset	Identified 84–89% of anomalies, outperforming one-class SVM, with significant implications for threat hunting.
Machine Learning for Malware Detection	Deep learning (AutoEncoder with Boltzmann machines) [YHD+22]	Not specified	Enabled detection of unknown malware, showing robust performance against zero-day attacks.
Machine Learning for Malware Detection	CNN-based architecture for malware classification [KRM+18]	Malimg, Microsoft Malware	Achieved 98.5% and 99.97% accuracy, outperforming state-of-the-art methods.
Machine Learning for Malware Detection	Droid-NNet framework for Android malware detection [MS19]	Android malware datasets	Outperformed existing techniques using F-beta score, showcasing robustness and effectiveness.

as Droid-NNet, which outperforms the existing cutting-edge ML techniques. This study shows robust and effective approaches using the F-beta score for Droid-NNet after comparing it with a number of algorithms like Linear Regression (LR), SVM, and DT [MS19].

A method that shows approaches that have been used to answer the main problem that has been addressed in this research thesis has been given in the next section.

4 Research Method

This section outlines the systematic approach employed to investigate and address the research problem that was mentioned in Section 1 of this research thesis. It builds upon the insights that were gained from the background and comprehensive literature review presented earlier (see Section 2 and 3). Precisely, it gives the methodological framework that has been adopted for modeling predictive models for cyber-threats detection. The approach that has been leveraged has several phases shown in Figure 4.1, and each designed to aid in achieving the objecctives that were set in this study.

4.1 Research Design

This study uses a predictive analytics design approach based on a quantitative research framework to analyze and predict cyber threats. The leveraged design is exploratory in nature, and it aims to uncover patterns and trends that can be assessed in a cybsersecurity perspective using ML algorithms.

The rationale behind using ML-based methods is based on the ability that ML algorithms have of handling large and complex datasets, where they can uncover hidden relationships that the traditional statistical methods might overlook. It is the authors' opinion that given the dynamic and the changing cyber-security threat landscape ML algorithms offers robust tools that can adapt to this changing landscape.

The research process that is shown in Figure 4.1 begins with data collection and processing in the process labeled 2. This includes handling missing data, encoding categorical variables, and transforming skewed distributions. Exploratory Data Analysis (EDA) is employed to identify patterns and relationships in the data. Next, feature importance is evaluated, enabling the selection of relevant predictors for the machine learning models.

In the Next step labeled 3, we do model training and implementation, using the selected ML models which are explained further on. This step includes splitting the dataset into train and test set, train the model and fine tune the hyperparameters in order to find the best parameters.

In the next phase that is labeled 4 is the evaluation phase, where the performance of the proposed modeling approach is assessed against predefined criteria and benchmarks. We employ appropriate evaluation metrics and methodologies. The final steps the research methodd labeled 5 shows a comprehensive analysis and synthesis of findings.

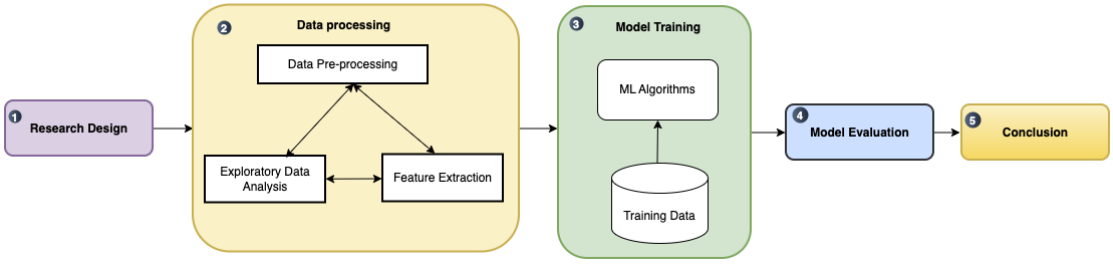

Figure 4.1: Research Method

4.2 Data Processing

In this section a discussion on the data processing approaches is given based on the process labelled 2 in Figure 4.1. In this section we discuss, the dataset Microsoft Malware Prediction, Pre-processing, Exploratory Data Exploratory, and Feature extraction.

4.2.1 Microsoft Malware Prediction Dataset

This study leverages the Microsoft Malware Prediction dataset from Kaggle [Mic24], a dataset that is curated by Microsoft for the purpose of advancing research in malware detection and prediction. The dataset is used in this study as a base to analyze and develop predictive models to identify potential existence of malware, a type of cyber-threat that is increasingly sophisticated and capable of bypassing traditional security measures.

The dataset provides extensive information on various system attributes and behaviors that can potentially indicate malware presence. This dataset includes features such as software configurations, antivirus detections, and device settings, offering a rich resource for exploring patterns that may correlate with malware activity. By utilizing this dataset, the study aims to build predictive models that can classify and forecast the likelihood of malware infection, focusing on identifying critical features that contribute most significantly to accurate prediction.

4.2.1.1 Data Description

A description of the data that has been used in this research thesis is given in this section. The variables in this dataset are shown in Table A.1. Some of the column descriptions that are not available from the data source or self-explanatory have been marked with an "NA". There are total 83 variables in the dataset, and the target variables is 'HasDetections' which is the ground truth and indicates that Malware was detected on the machine.

Figure 4.2 illustrates the distribution of the target variable in the dataset, which is nearly balanced between the two categories: Yes (1) and No (0), with the percentage of 49.5% and 50.5% respectively. This close-to-even distribution shows a well-balanced dataset for the target variable, which is important to avoid bias in predictive modeling.

It is crucial to note that, due to limitations in computational resources, we have opted to use a subset of 10,000 records from the original dataset of 8.9 millions

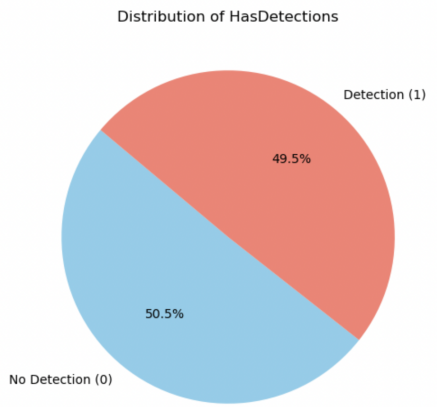

Figure 4.2: Distribution of HasDetections

records. This decision was made to balance processing efficiency with the need to maintain the dataset's representativeness. Even though the subset is smaller, it retains the distribution characteristics of the full dataset, ensuring that our analysis remains robust and reliable.

4.2.2 Pre-processing

The raw dataset has undergone several transformations to prepare it for analysis and modeling. This process aims to address missing values, handle skewness, and ensure consistency and compatibility of the data. In this study, preprocessing of the Microsoft Malware Prediction dataset involves several steps.

Firstly, columns with a high percentage of missing values or skewed distributions are identified and removed to avoid introducing bias into the analysis. Next, missing values are imputed using appropriate strategies, such as replacing missing values in categorical columns with the most frequent value and replacing missing values in numerical columns with the median value. This ensures that the dataset is complete and ready for further analysis. Additionally, variable transformation is performed to transform some variables into bigger group that make it easier for the models to predict, then encoding steps are used to convert the variables into the format that is compatible for training models at later steps. The preprocessing stage plays a crucial role in ensuring the quality and integrity of the dataset, laying the foundation for accurate and reliable analysis and modeling.

4.2.2.1 Handling High Skewness in Data

In the dataset, we identified several columns with high skewness, where one or a few values dominate the distribution of the data. Skewness in this context refers to the imbalanced frequency of certain values within a column, which can create a situation where the column provides little informative value to distinguish different outcomes. By eliminating them, we reduced the risk of overfitting as well as their limited predictive power.

$$\text{Percentage of Most Frequent Value} = \left(\frac{\text{Frequency of Most Frequent Value}}{\text{Total Number of Records}} \right) \times 100 \tag{4.1}$$

To detect high skewness, we calculated the percentage of the most frequent value in each attribute relative to the total number of records as shown in Equation 4.1, also, some of the features addressed when handling skewness are shown in Table 4.1. Columns where a value occupies for more than 95% of the records were flagged as highly skewed and were removed out of the used dataset.

Table 4.1: Snapshot of some features in the dataset about skewness

Feature	**Unique Values**	**Values in the biggest category (%)**	**Type**
Census_ThresholdOptIn	1	100	Numerical
Census_IsFlightingInternal	1	100	Numerical
Census_IsWIMBootEnabled	1	100	Numerical
Census_IsFlightsDisabled	1	100	Numerical
AutoSampleOptIn	1	100	Numerical
...	...	...	...
AvSigVersion	1661	1.23	Categorical
CityIdentifier	3611	1.18	Numerical
Census_FirmwareVersionIdentifier	3558	1.05	Numerical
Census_SystemVolumeTotalCapacity	6687	0.62	Numerical
MachineIdentifier	10000	0.01	Categorical

4.2.2.2 Handling Missing Data

There are several columns with a significant proportion of missing values in the dataset. Missing data can bring challenges in data analysis and modeling, as it may reduce the reliability of the results and make the process of training machine learning models more complicated, a sample of variables with percentage of missing value is shown in Table 4.2. Hence, columns with more than 95% missing values were removed from the dataset to ensure the integrity and effectiveness of our analysis, enhance the model performance and improve model interpretability with less but informative columns.

Moreover, for the remaining missing value in the remaining attributes with less than 95% missing, the employed methodology utilizes the `SimpleImputer` class from the `sklearn.impute` module. Initially, the algorithm identifies categorical and numerical columns within the dataset using the `select_dtypes` method. Subsequently, it proceeds to instantiate separate `SimpleImputer` objects for each data type, with distinct strategies: replacing missing values in categorical columns with the most frequent value and in numerical columns with the median. Through the `fit_transform` method, the imputer is then fitted to the dataset, and missing values are replaced accordingly, ensuring uniform handling of missing data across the dataset. This process results in a refined dataset, devoid of missing values, poised for subsequent analysis and modeling endeavors. After the pre-processing steps, we reduced from 83 variables to 59 variables.

Table 4.2: Snapshot of some features in the dataset about missing data

Feature	**Unique Values**	**Missing value (%)**	**Type**
PuaMode	1	99.97	Categorical
Census_ProcessorClass	3	99.58	Categorical
DefaultBrowsersIdentifier	82	95.28	Numerical
Census_IsFlightingInternal	1	83.09	Numerical
Census_InternalBatteryType	13	71.2	Categorical
...	...	...	...
ProductName	2	0.00	Categorical
Census_HasOpticalDiskDrive	2	0.00	Numerical
Census_ChassisTypeName	23	0.00	Categorical
Census_PowerPlatformRoleName	8	0.00	Categorical
HasDetections	2	0.00	Numerical

4.2.2.3 Data Transformation

There are 6 variables in the dataset that are some versions composed of different numbers and decimal, which leads to too high cardinality, such as EngineVersion, AppVersion, AvSigVersion, Census_OSVersion, OsBuildLab. Usually those contains all the the released versions even with a small change, e.g iOS version 13.1, when a small change happens there is a 13.1.1, 13.1.2, 13.1.3 version released, and ideally the closer two sub versions are, the more similar vulnerabilities they have. Therefore, we decide to group those sub versions into a bigger group such as 13.1 only. Below tables show the original (Table 4.3) and transformed (Table 4.4) form of those variables.

4.2.2.4 Variable Encoding

In this step a discussion in give how the variables have been encoded. The variables have been encoded based on the type of variables. In the dataset that has been leveraged, there are 29 numerical variables (50%), 5 binary variables (6.62%) and 24 categorical variables (41.4%), as shown in Figure 4.3. Label encoding is used for the categorical columns after we cleaned them up, and for Numerical variables we use Standard Scaler to scale them.

Table 4.3: Originn of the 'version' variables

EngineVersion	**AppVersion**	**AvSigVersion**	**Census_OSVersion**	**OsBuildLab**
1.1.15100.1	4.18.1807.18075	1.273.1735.0	10.0.17134.165	17134.amd64fre.rs4.release.180410-1804
1.1.14600.4	4.13.17134.1	1.263.48.0	10.0.17134.1	17134.amd64fre.rs4.release.180410-1804
1.1.15100.1	4.18.1807.18075	1.273.1341.0	10.0.17134.165	17134.amd64fre.rs4.release.180410-1804
1.1.15100.1	4.18.1807.18075	1.273.1527.0	10.0.17134.228	17134.amd64fre.rs4.release.180410-1804
1.1.15100.1	4.18.1807.18075	1.273.1379.0	10.0.17134.191	17134.amd64fre.rs4.release.180410-1804
...	...	...	...	...
1.1.15100.1	4.18.1806.18062	1.273.499.0	10.0.17134.165	17134.amd64fre.rs4.release.180410-1804
1.1.15100.1	4.14.17613.18039	1.273.1311.0	10.0.14393.1198	14393.1198.amd64fre.rs1.release_sec.170427-1353
1.1.15200.1	4.18.1807.18075	1.275.1198.0	10.0.15063.1266	15063.amd64fre.rs2.release.170317-1834
1.1.15200.1	4.18.1807.18075	1.275.995.0	10.0.17134.229	17134.amd64fre.rs4.release.180410-1804
1.1.14901.4	4.14.17639.18041	1.269.277.0	10.0.16299.431	16299.431.amd64fre.rs3.release_svc_escrow.1805...

Table 4.4: Transformation of the 'version' variables

EngineVersion	**AppVersion**	**AvSigVersion**	**Census_OSVersion**	**OsBuildLab**
1.1.15	4.18.18	1.273	10.0.17	amd64fre
1.1.14	4.13.17	1.263	10.0.17	amd64fre
1.1.15	4.18.18	1.273	10.0.17	amd64fre
1.1.15	4.18.18	1.273	10.0.17	amd64fre
1.1.15	4.18.18	1.273	10.0.17	amd64fre
...	...	...	...	...
1.1.15	4.18.18	1.273	10.0.17	amd64fre
1.1.15	4.14.17	1.273	10.0.14	amd64fre
1.1.15	4.18.18	1.275	10.0.15	amd64fre
1.1.15	4.18.18	1.275	10.0.17	amd64fre
1.1.14	4.14.17	1.269	10.0.16	amd64fre

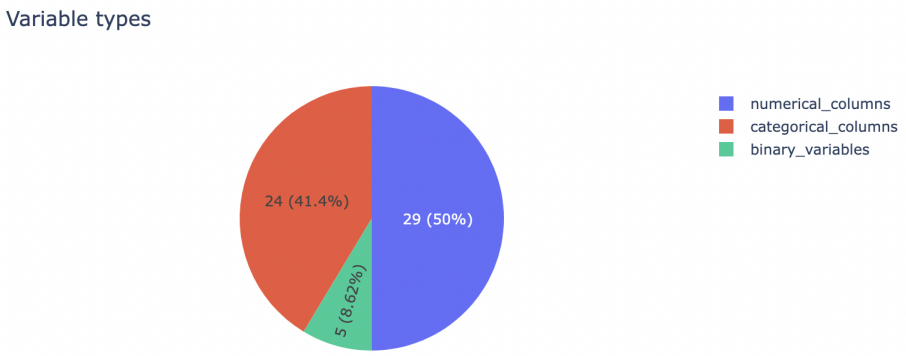

Figure 4.3: Variable types

It is worth mentioning that, label encoding was applied given that this approach is efficient to categorical data with a limited number of distinct categories. This and ensures that there is compatibility with algorithms that process numerical input. On the other hand, Standard Scaler has been used to normalize the data by centering it around the mean and scaling it to have unit variance. This was important for the ML models given that it will prevent the variables that have larger variables from having to dominate during training process but to have it balanced.

4.2.3 Exploratory Data Analysis

In the Exploratory Data Analysis (EDA) phase, we first examined the distribution of categorical variables, as shown in Figure 4.4. It was observed that variables such as `EngineVersion`, `AppVersion`, `AvSigVersion`, and `OsBuildLab` exhibited fewer unique values after preprocessing. However, the distribution of records within these variables was uneven, with certain values, often representing the latest versions, being heavily dominant. For example, the values 1.1.15 for `EngineVersion`, 4.18.18 for `AppVersion`, and 1.275 or 1.273 for `AvSigVersion` were particularly prevalent.

Consequently, some variables, including `Census_OSWAutoUpdateOptionsName`, `Census_OSInstallTypeName`,, and `Census_OSVersion`, displayed more balanced distributions with smaller differences between class frequencies. We also visualized the distribution of binary variables, as shown in Figure 4.5, where most exhibited significant class imbalances. The exception here was `Census_IsSecureBootEnabled`, which displayed a balanced distribution, while other binary variables had one class overwhelmingly outnumbering the other, suggesting the potential for skewed model outcomes.

In addition, we analyzed the correlation between the numerical variables using a heatmap, presented in Figure 4.6. This analysis was able revealed two pairs of highly correlated variables: `Census_OSBuildNumber` and `OsBuild` (with a correlation of 0.95). The oother with a higher correlation include `Census_OSUILocaleIdentifier` and `Census_OSInstallLanguageIdentifier`. The high correlation between these pairs could be because of the following: `Census_OSBuildNumber` and `OsBuild` both represent system build information, while `Census_OSUILocaleIdentifier` and `Census_OSInstallLanguageIdentifier` correspond to locale and language settings in the Operating System (OS). From this, we found the need for a careful feature selection to avoid redundancy and potential issues with multicollinearity in subsequent modeling steps, hence, 2 variables 'OsBuild', 'Census_OSUILocaleIdentifier'were

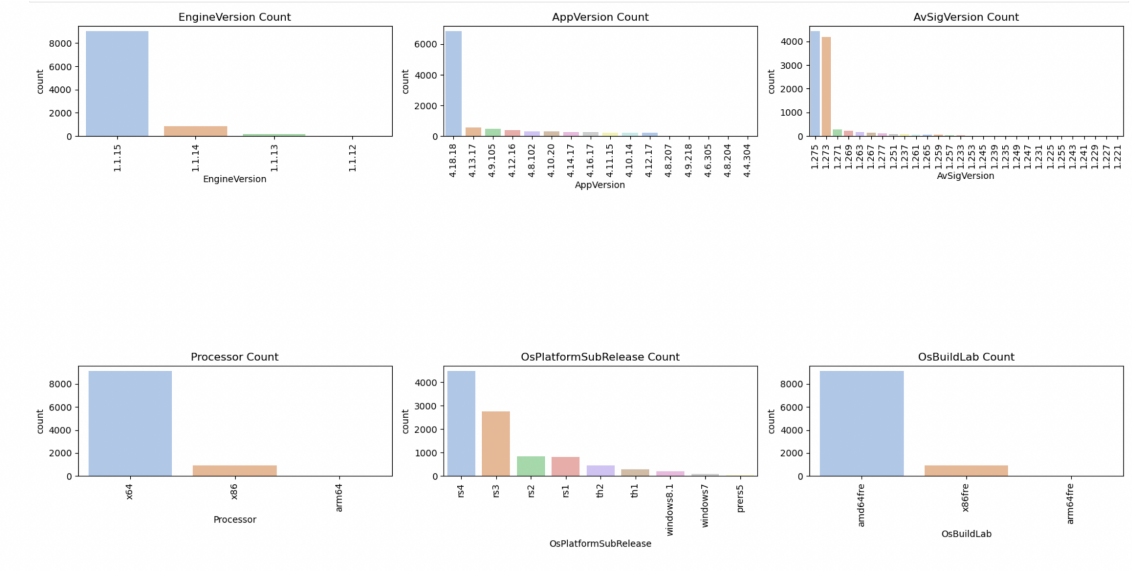

Figure 4.4: Sample Distribution of categorical variables

removed from the dataset. The key features from the EDA have been aslo shown in Table 4.5.

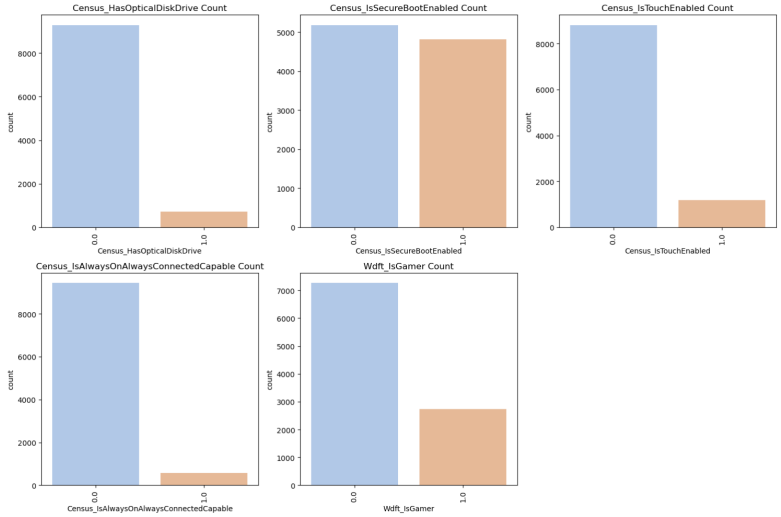

Figure 4.5: Distribution of binary variables

Table 4.5: Key Observations from Exploratory Data Analysis

Variable Type	Observation	Example/Details
Categorical Variables	Uneven distribution, with dominant classes	'EngineVersion': 1.1.15 'AppVersion': 4.18.18, 'AvSigVersion': 1.275, 1.273
	Balanced distribution with small class differences	'Census_OSInstallTypeName', 'Census_OSWAutoUpdateOptionsName', 'Census_OSVersion'
Binary Variables	Balanced distribution	'Census_IsSecureBootEnabled'
	Highly imbalanced distribution	Other binary variables with one class dominating the other
Numerical Variables	High correlation between pairs	'Census_OSBuildNumber' & 'OsBuild' (correlation: 0.95)
	Moderate correlation between pairs	'Census_OSUILocaleIdentifier' & 'Census_OSInstallLanguageIdentifier'

4.2.4 Feature Extraction

Feature extraction is a crucial step in the process of building an effective machine learning model. In this study, we explore the importance of different features in

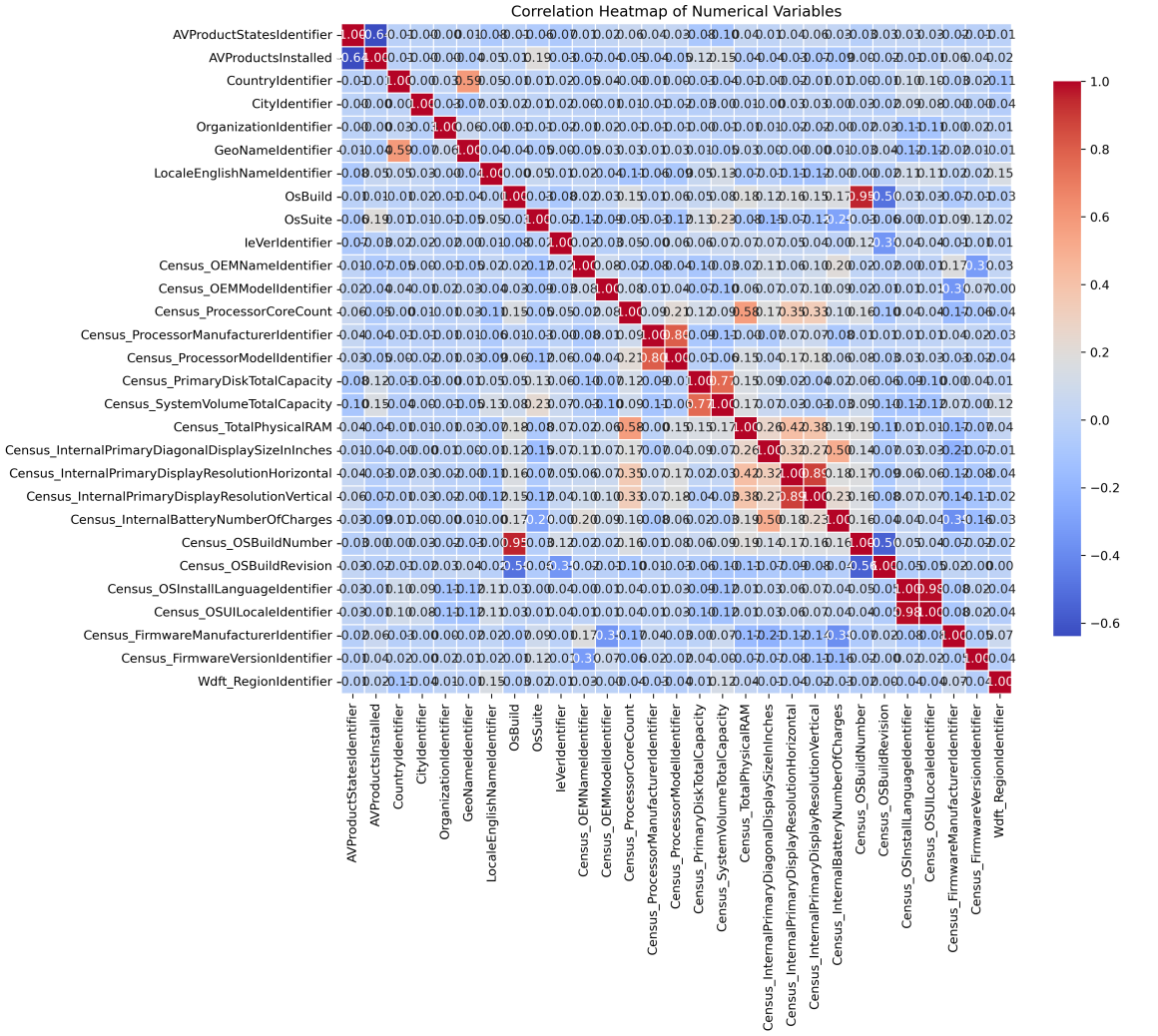

Figure 4.6: Correlation Heatmap

predicting malware presence using the Random Forest classifier. By identifying and understanding the significance of each feature, we can enhance the model's performance and potentially uncover underlying patterns within the dataset that contribute to malware detection.

4.2.4.1 Random Forest and Feature Importance

To determine the most relevant features, we employed a Random Forest classifier with a grid search for hyperparameter tuning. This approach allowed us to systematically explore different combinations of hyperparameters, ensuring the selection of the best-performing model. The grid search was conducted using 5-fold cross-validation, and the best parameters were identified as follows: `max_depth=10`, `min_samples_leaf=2`, `min_samples_split=5`, and `n_estimators=200`. This resulted in the best cross-validation score of 0.6238.

4.2.4.2 Analysis of Feature Importance

Once the optimal model was trained on the entire training dataset, we extracted the feature importances from the Random Forest classifier. Feature importances provide

a measure of the contribution of each feature to the model's predictive power. By analyzing these importances, we can prioritize features that have a higher impact on the model's accuracy and potentially reduce the dimensionality of the dataset for future analyses.

The top 20 feature importances were visualized using a bar plot as shown in Figure 4.7, highlighting the relative importance of each feature. This visualization helps in easily identifying the most influential features, which can guide further investigations and model refinements.

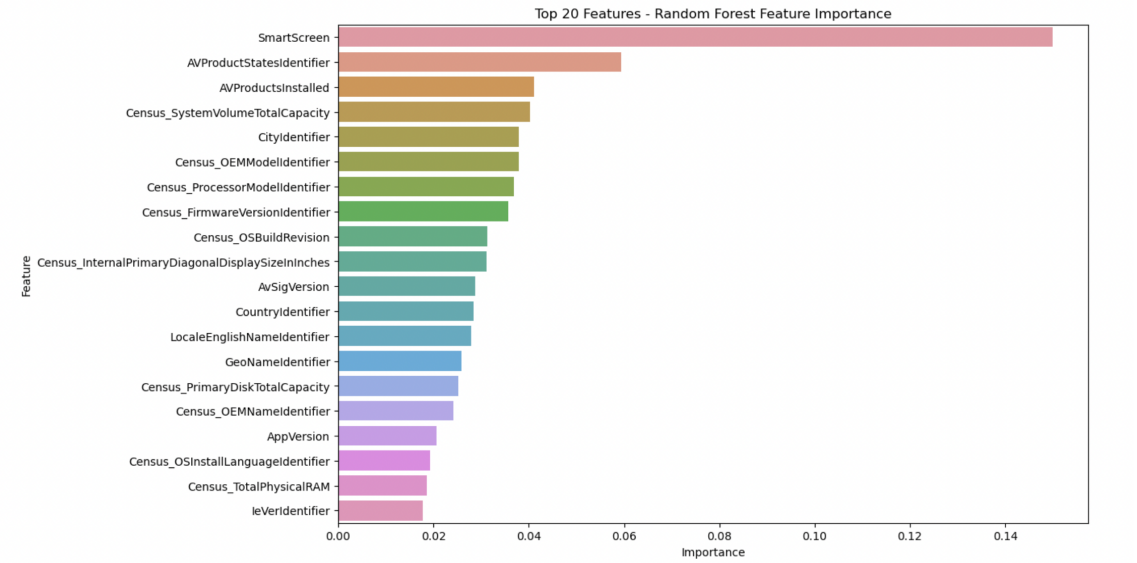

Figure 4.7: Top 20 Features - Random Forest Feature Importance

We can observe from Figure 4.7 that variable 'SmartScreen' has significantly high important score (0.14) compared to the second important variable AVProduct-Stateidentifier (0.06) and the third variable AVProductInstalled (0.04). Microsoft Defender SmartScreen is a feature designed to protect users from malicious websites, harmful applications, and the downloading of potentially dangerous files, hence it is reasonable that this variable has highest importance score. The remaining variables from the forth score onwards do not have significant gap in importance score.

We do some further bivariate analysis to see the relationship between these feature importance and target variable. The variable SmartScreen is the SmartScreen enabled string value, and from Figure 4.8, majority of cases has Malware detected is in class 'ExistNotSet', which means the value exists but is blank.

In Figure 4.9, the next four most important numerical features are shown. Among them, AVProductStatesIdentifier exhibits a significant difference between class 0 and 1 of target variable. However, since the mapping of this identifier is unavailable, we are unable to determine which specific states have the greatest impact on malware detection. Variable 'AvProductsInstalled' shows that if number of Antivirus products installed is 1 then it has higher chance to be attacked by Malwares than the machines have more than 1 Antivirus Product installed. For the remaining 2 variables in top 5, there is no clear difference between class 0 and 1 of Target variable.

It is worth noting that the feature extraction process using the Random Forest classifier allowed us to identify key features that significantly contribute to the prediction of malware. This step not only enhances the model's interpretability but

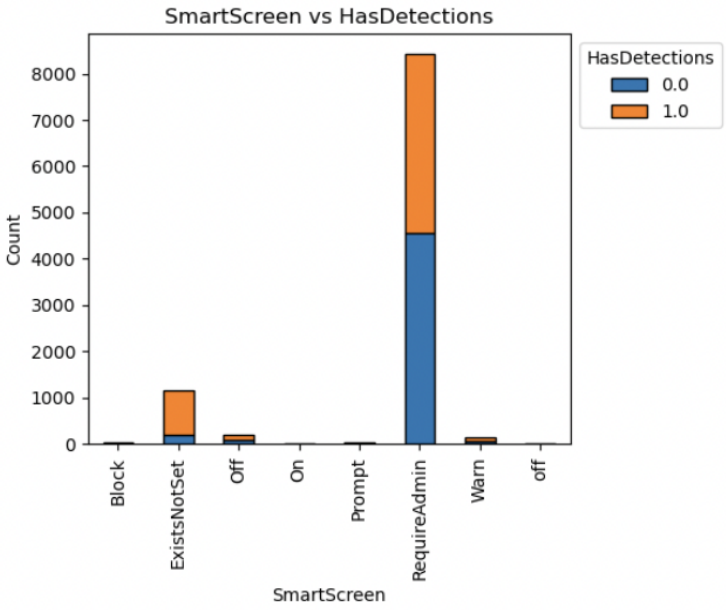

Figure 4.8: SmartScreen vs HasDetections

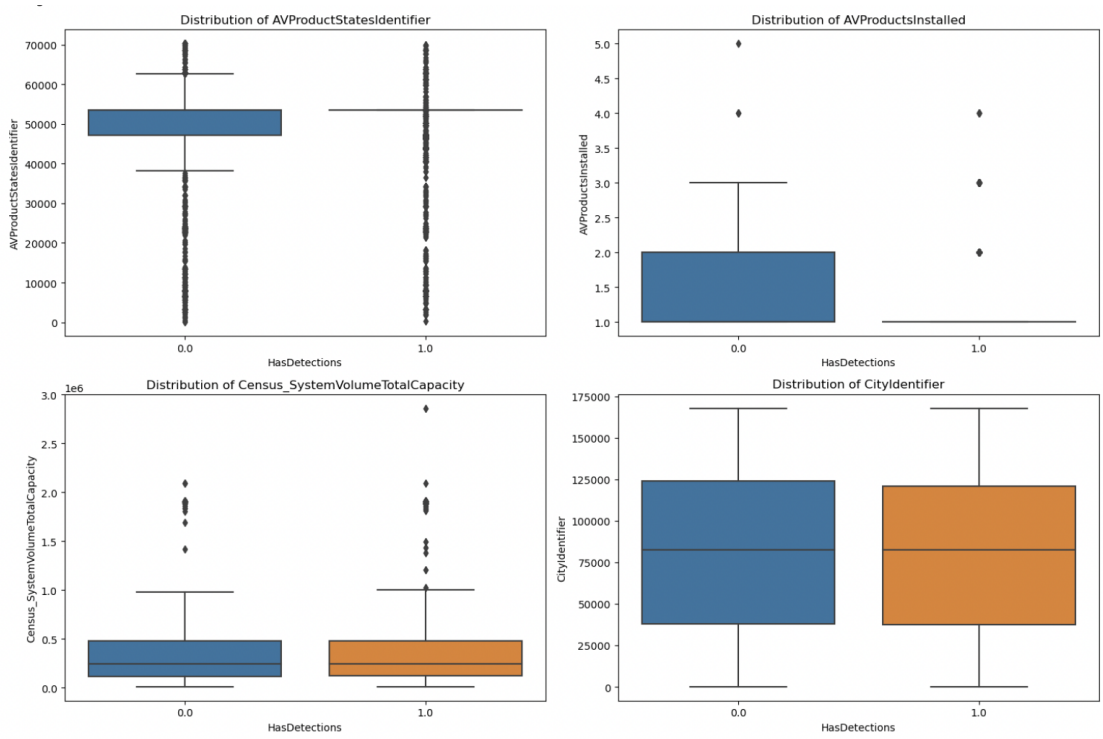

Figure 4.9: Top 5 Feature Important vs HasDetections

also lays the foundation for building more robust and efficient predictive models.

4.2.5 Feature Selection Based on Importance

After identifying the most significant features through the feature importance analysis using the Random Forest classifier, we proceeded to refine our model by focusing only on these important variables with importance score higher than 0.005. This step aimed to reduce the dimensionality of the dataset, potentially enhancing the

model's performance and interpretability, and this threshold resulted in 41 key features being selected out of the original set.

We then created a new training and testing dataset comprising only these 41 important features. The Random Forest classifier was reinitialized with the previously identified optimal hyperparameters: `max_depth=10`, `min_samples_leaf=4`, and `n_estimators=150`. The model was retrained on the reduced feature set to evaluate if the reduced complexity would maintain or improve its predictive power. This selective approach not only simplifies the model but also enhances its computational efficiency by focusing on the most influential features, which are critical for accurate malware prediction.

4.3 Model Development and Evaluation

In order to address the growing complexity and sophistication of cyber threats necessitate robust and effective machine learning models that can efficiently handle large datasets like the Microsoft Malware Prediction Dataset [Mic24], that has been leveraged in this study and and its diverse features. To address this challenge, we explore aa number of algorithms, that includes: Random Forest (RF), Logistic Regression (LR), Quadratic Discriminant Analysis (QDA), Linear Discriminant Analysis (LDA), XGBoost, and Fully Connected Neural Networks (FCNN)

4.3.1 Motivation for Diverse set of Model Selection

RF has been chosen for its robustness coupled with its performance when it comes to dealing with complex, and high-dimensional data. Notably, RF's strength lies in their ability to perform feature selection and ensemble learning, which reduces overfitting [BHA09][PIL06]. Next, LR has been included in this study owing to its simplicity and interpretability. In order to understand how the individual features are able to influence the outcome, LR is capable of giving interpretable coefficients that make it easier to analyze and test hypotheses [CVA06]. On the other hand, QDA and LDA have been been selected because of its discriminative power of classificatying tasks [BG98][DB09]. Next, XGBoost has been chosen in this study given that it has a proven track record of superior performance in a number of ML competitions [CHB+15]. In addition, it has ability of handling large datasets, and also to optimize complex models. It also provide regularization to prevent overfitting makes it a powerful tool in predictive modeling. Finally, the FCNNs has been included in order to capture the complex, and non-linear relationships that exist in the data. Specifically, the FCNNs are useful when the data shows patterns that cannot be easily captured by the traditional linear models. Inspite of that, their capacity to comprehend and represent these non-linearities is able to make makes them indispensable when it comes to more complex data patterns [SVSS15].

The rationale behind the selection of the abovementioned models lies in their unwavering complementary strengths-which collectively are able to provide a comprehensive analysis of the data.

4.3.2 Model Development Process

In the model development process ML algorithms (RF, LR, QDA,LDA and FCNN)) were trained on the preprocessed dataset and fine-tuning their hyperparameters to achieve optimal performance. Following training, each model's performance was rigorously validated using various metrics, ensuring that the chosen models not only fit the training data well but also generalize effectively to unseen data. This approach allowed us to identify the most suitable models for our specific application, balancing accuracy, computational efficiency, and interpretability.

4.3.3 Training and Testing Procedures

To ensure robust performance and generalizability of the ML models, it is crucial to properly train and test them. Initially, we split the dataset into a training set of (80%) and a test set (20%). After this,, we applied the Cross-Validation technioque in order to further divide the training set into multiple subsets, for both training and validation. This approach helps identify the optimal parameters for each model. Once the best parameters were determined, we used them to make predictions on the test set and evaluate the model's performance using various metrics.

4.3.3.1 Hyperparameter Tuning

To optimize the performance of our ML models, we used GridSearchCV for hyperparameter tuning. This method evaluates multiple hyperparameter combinations and selects the best configuration via cross-validation, as is shown in 4.6.

Table 4.6: Hyperparameters for the models

Model	**Parameters**	**Value to choose**
Random Forest	n_estimators	[50, 100, 150, 200]
	max_depth	[None, 10, 20]
	min_samples_split	[2, 5,10]
	min_samples_leaf	[1, 2, 4]
Logistic Regression	C	[0.001, 0.01, 0.1, 1, 10]
	penalty	['l1', 'l2']
	solver	['liblinear', 'saga']
QDA	reg_param	[0.1, 0.5, 1.0]
XGBoost	learning_rate	[0.1, 0.01, 0.001]
	max_depth	[3, 5, 7]
	n_estimators	[50, 100, 200]
	subsample	[0.5, 1.0]
	colsample_bytree	[0.8, 1.0]

For the Random Forest model, we tuned 'n_estimators', 'max_depth', 'min_samples_split', and 'min_samples_leaf' with values such as 50, 100, 150, 200 for 'n_estimators' and None, 10, 20 for 'max_depth'.

For Logistic Regression, we tuned the 'C' parameter (0.001 to 10), 'penalty' (L1 or L2), and 'solver' (liblinear or saga). For QDA model, we adjusted 'reg_param' with values 0.1, 0.5, and 1.0. For the XGBoost model, the parameters `learning_rate`, `max_depth`, `n_estimators`, `subsample`, and `colsample_bytree` were optimized by

testing various values to identify the best configuration for improving model performance and reducing overfitting. For the LDA model we did not perform hyperparameter tuning as it is a linear classifier with default settings that generally perform well without extensive optimization.

For the FCNN, we experimented with three different architectures, varying the number of hidden layers, activation functions, optimizers, learning rates, and regularization strengths as is shown in Table 5.4.

Table 4.7: Summary of FCNN Models

Model	**Hidden Layers**	**Activation**	**Optimizer**	**Learning Rate**	**Regularization (λ)**
Model 1	2 (128, 64)	ReLU	Adam	1×10^{-3}	0.001
Model 2	3 (128, 64, 32)	tanh	SGD	1×10^{-4}	0.01
Model 3	4 (512, 256, 128, 64)	tanh	SGD	1×10^{-2}	0.0005

4.3.4 Performance Metrics

In this study, we employed key performance metrics to evaluate the effectiveness of the classification models: accuracy (Acc), precision ($Prec$), recall (Rec), and the F1 score. These metrics were essential for assessing predictive performance across different dimensions. Accuracy (Acc) is defined as the proportion of correctly classified instances over the total number of instances:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.2}$$

where TP represents true positives, TN represents true negatives, FP represents false positives, and FN represents false negatives. Precision ($Prec$) measures the proportion of true positive predictions among all positive predictions:

$$Prec = \frac{TP}{TP + FP} \tag{4.3}$$

Recall (Rec), also known as sensitivity, quantifies the proportion of true positive predictions among all actual positive instances:

$$Rec = \frac{TP}{TP + FN} \tag{4.4}$$

The F1 score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance:

$$F1 = \frac{2 \times Prec \times Rec}{Prec + Rec} \tag{4.5}$$

These metrics were systematically applied to evaluate and compare the performance of the models, such as its ability to make accurate predictions, minimize false positives, and effectively capture positive instances. By exploring these metrics, we gain a good understanding of the model's strengths and weaknesses, enabling informed decisions in model selection. A detailed discussion on how the performance metrics have been applied is discussed in the next section.

5 Evaluation

In this section we provide an examination of the outcomes obtained from our study based on the method highlighted in Section 4 of this research thesis. This section discusses the outcome and evaluated the effectiveness of the ML models employed in this study. Through an analysis and evaluation, we aim to unveil the strengths and limitations of each ML model, shedding light on their performance across various performance metrics. Based on the outcome of this evaluation process, we not only scrutinize numerical results but also explore their implications for real-world cyber threat scenarios.

5.1 Discussion of the Results

We analyzed the performance of each ML model employed, and then checked their strengths, weaknesses, and overall effectiveness. Through a systematic comparison of model performances, we were able to scrutinize various metrics such as accuracy (*Acc*), precision (*Prec*), recall (*Rec*), the F1 score, and ROC-AUC analysis. Furthermore, we interpret and report these findings.

5.1.1 Comparison of Model Performances

To assess the efficacy of various ML models in cyber threat detection, we conducted a rigorous evaluation of their performances. Table 5.1 presents a comprehensive comparison of key metrics across different models (RF, LR, QDA, LQD and FCNN).

Table 5.1: Performance Metrics Comparison Across Models

Metric	Random Forest	Logistic Regression	QDA	LDA	XGBoost	Neural Network
Accuracy	0.62000	0.612500	0.601000	0.608500	0.624000	0.613000
Precision	0.620084	0.612560	0.604396	0.608485	0.624006	0.618655
Recall	0.620000	0.628968	0.601000	0.608500	0.624000	0.613000
F1 Score	0.619768	0.620656	0.596808	0.608412	0.623897	0.609179
AUC	0.619808	0.612367	0.600190	0.608383	0.623872	0.613799

Table 5.1 highlights variations in accuracy, precision, recall, F1 score, and area under the ROC curve (AUC) across different models. Random Forest and XGBoost exhibit relatively higher accuracy and precision compared to other models, while QDA shows the lowest performance in terms of accuracy and precision. These

findings provide better reflections into the relative strengths and weaknesses of each of the model, guiding the selection of optimal approaches for cyber threat prediction tasks.

5.1.1.1 Accuracy and Precision

Accuracy measures the overall correctness of the model's predictions. while precision quantifies the correctness of positive predictions. We observe that Random Forest and XGBoost models exhibit the highest accuracy and precision scores among the evaluated models, Accuracy comparison has been shown in Figure 5.1 . These findings suggest that both Random Forest and XGBoost classifiers are effective in accurately predicting cyber threats based on the input features. However, it's important to note that while these models perform well in terms of accuracy and precision, other metrics such as recall, F1 score, and AUC should also be considered for a comprehensive evaluation of model performance.

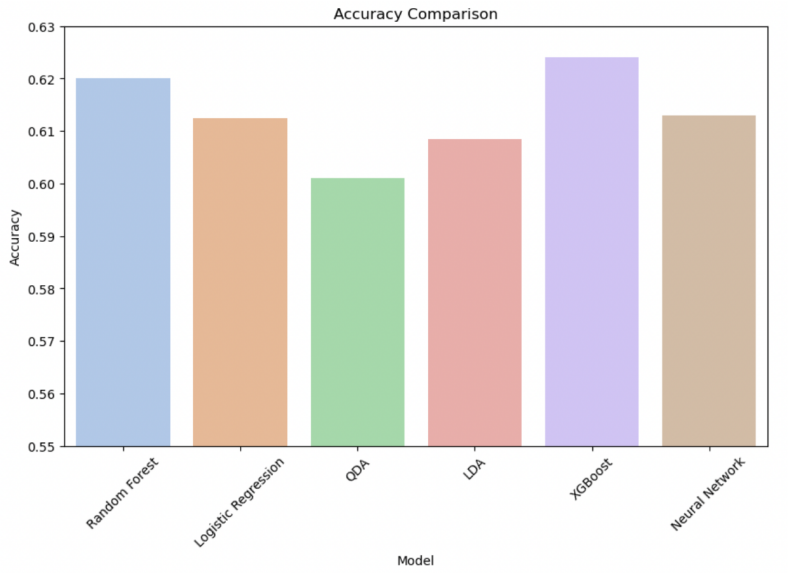

Figure 5.1: Accuracy Comparison

5.1.1.2 Recall and F1 Scores

In reviewing the recall and F1 scores depicted in Figure 5.2, several important observations are seen regarding the performance of each model. Firstly, from Figure 5.2 it's evident that the LR model demonstrates a significantly high recall across the models even though Accuracy and Precision are lower than RF and XGBoost. This implies that LR model effectively identify a significant portion of true positive instances, crucial for detecting potential cyber threats accurately, followed by RF and XGBoost. Conversely, QDA and LDA exhibit slightly lower recall scores, indicating potential shortcomings in capturing positive instances compared to other models.

Secondly, regarding the F1 scores, which provide a balanced assessment of precision and recall, in Figure 5.3 we observe that the 3 models RF, LR and XGBoost maintain relatively high F1 scores compared to the remaining 3 models. This suggests that RF, LR and XGBoost achieve a better balance between minimizing false

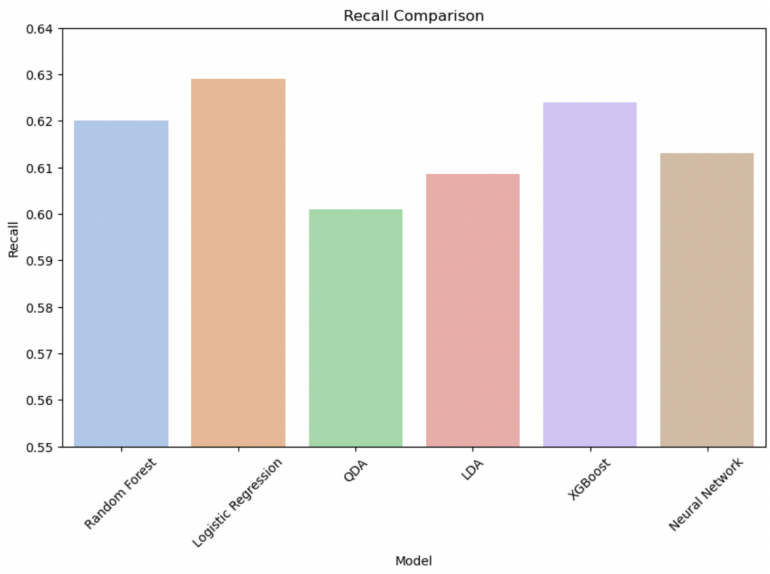

Figure 5.2: Recall Comparison

positives and false negatives, making them more robust choices for cyber threat detection tasks where both types of errors are critical.

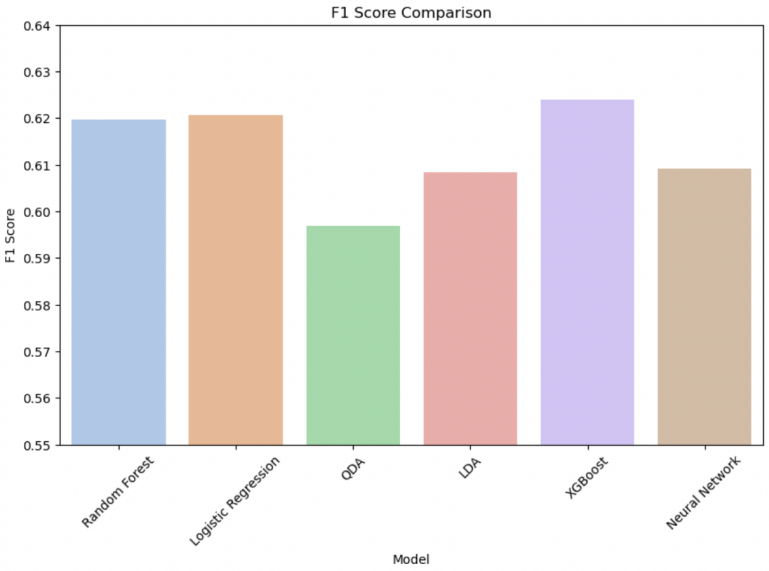

Figure 5.3: F1 Score Comparison

While accuracy and precision offer valuable insights into a model's overall performance, the recall and F1 scores provide a deeper understanding of its effectiveness in identifying true positive instances and maintaining a balance between precision and recall. Thus, considering these metrics collectively enables a more comprehensive evaluation of each model's suitability for cyber threat detection applications.

5.1.1.3 AUC Analysis

The area under the ROC curve (AUC) is a crucial metric for evaluating the performance of classification models, particularly in binary classification tasks like cyber threat detection. A higher AUC indicates that the model has better discriminatory

power, distinguishing between positive and negative instances more effectively.

In our comparison of model performances, depicted in the AUC Comparison plot as is shown in Figure 5.4, it's evident that XGBoost emerges as the top-performing model, closely followed by RF. These two models exhibit notably higher AUC values compared to LR, QDA, LDA, and the Neural Network (NN) model.

The higher AUC values for XGBoost and RF suggest that these models have superior discriminatory abilities in distinguishing between cyber threat and non-threat instances. This implies that XGBoost and RF are better equipped to rank instances correctly, making them preferable choices for cyber threat detection applications where accurately identifying positive instances is critical.

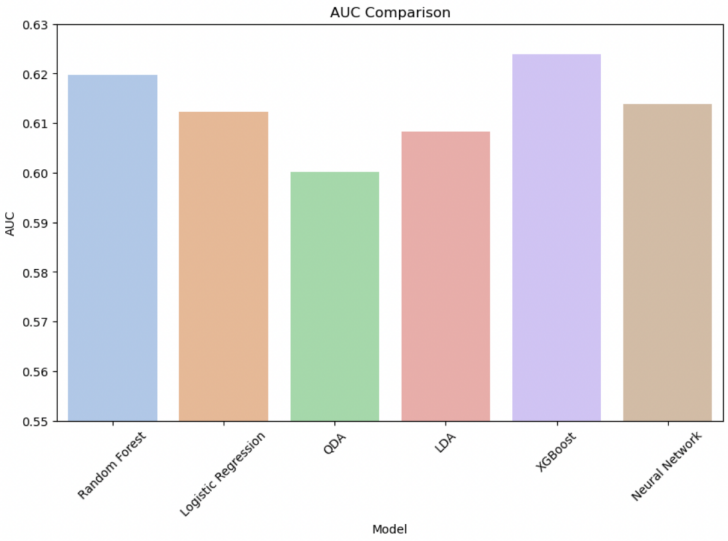

Figure 5.4: AUC Comparison

5.1.1.4 ROC Curve

The Receiver Operating Characteristic (ROC) curve is a graphical representation of the true positive rate (Sensitivity) against the false positive rate (1-Specificity) for different threshold values. It provides a comprehensive view of the trade-off between true positive and false positive rates, allowing us to evaluate the performance of classification models across various threshold values.

In the ROC curve depicted in Figure 5.5, we compare the performance of different models in terms of their AUC (Area Under the Curve) values. A higher AUC indicates better discrimination ability of the model, with values closer to 1 indicating superior performance.

From the ROC curve, we observe that Random Forest (RF) and XGBoost exhibit the highest AUC values, followed by Logistic Regression (LR) and Linear Discriminant Analysis (LDA). Quadratic Discriminant Analysis (QDA) and the Fully Connected Neural Network (FCNN) show relatively lower AUC values compared to the other models.

The ROC curve analysis reaffirms the findings from our previous comparisons, highlighting the effectiveness of ensemble methods like Random Forest and XGBoost in cyber threat detection tasks, as they demonstrate superior discriminatory power in distinguishing between threat and non-threat instances.

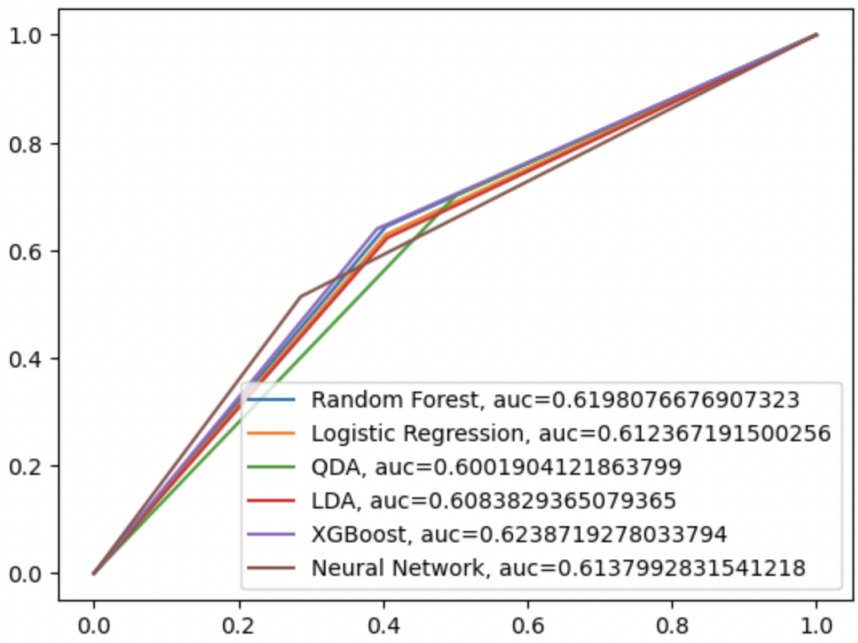

Figure 5.5: ROC Comparison

5.1.2 Evaluation of Predictive Models

In this section, we evaluate the predictive power of the ML models, by splitting the dataset randomly into training and testing sets five additional times. Each model is retrained on the new data splits, and predictions are generated for the corresponding test sets. The average performance metrics across all runs are then computed to confirm that the models generalize effectively to unseen data. Table 5.2 show the average performance metrics for 5 runs of all models.

Table 5.2: Average performance for Evaluation

Metric	**Random Forest**	**Logistic Regression**	**QDA**	**LDA**	**XGBoost**	**Neural Network**
Accuracy	0.6224	0.6141	0.5915	0.6086	0.6075	0.6305
Precision	0.623491	0.614534	0.59553	0.60895	0.608036	0.639032
Recall	0.6224	0.6141	0.5915	0.6086	0.6075	0.6305
F1 Score	0.621571	0.61402	0.585112	0.608313	0.607516	0.627506
AUC	0.622313	0.613628	0.589773	0.608449	0.607559	0.633303

Table 5.2 illustrates the average accuracy of five models, we see Neural Network model has surprisingly high performance than the other models, across all five metrics Accuracy, Precision, Recall, F1, AUC, followed by RF model. We can observe that the Neural Network (NN) show better performance in unseen data, while RF still keeps the good predictive model compared to the first training time. In the opposite, XGBoost shows lower performance in new unseen data.

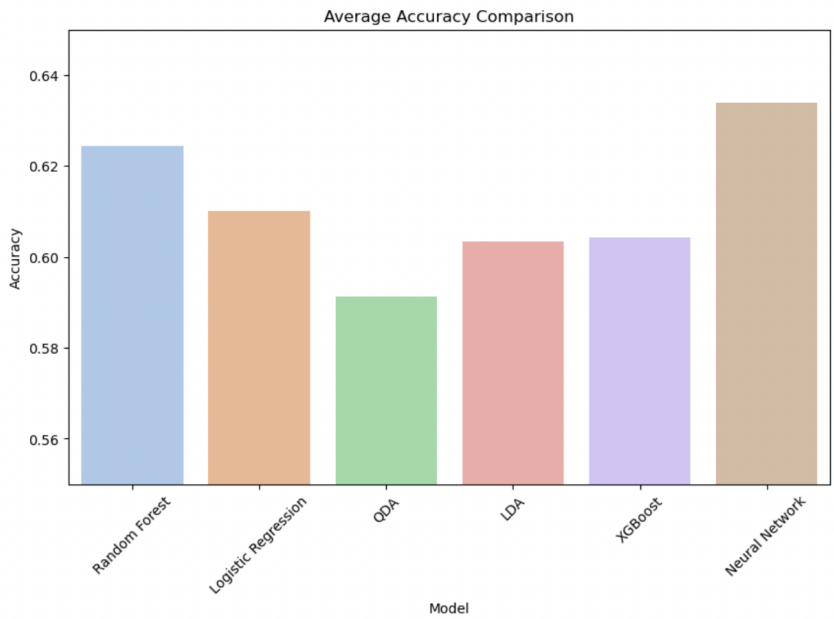

Figure 5.6: Average Accuracy Comparison of the ML models

5.1.3 Interpretation of Findings

The interpretation of the findings from our comprehensive evaluation of various models applied to the cyber threat detection approach has revealed that the Neural Nets and Random Forest model both demonstrate superior overall performance with the highest average accuracy and balanced precision, recall, and F1-scores when we evaluation the model in 5 different set of train-test data. This suggests its reliability and robustness in distinguishing between benign and malicious activities, making it a strong candidate for real-world cyber threat detection systems.

Comparatively, our results align with existing literature that emphasizes the efficacy of ensemble methods like RF and deep learning method such as FCNN. These models not only excelled in metrics like accuracy and precision but also demonstrated resilience against data variability — a key challenge in cybersecurity. In contrast, simpler models like LR and LDA showed moderate performance. The underperformance of the XGBoost model suggests that the Gradient Boosting method might work well at first training dataset but performed worse in new data for this dataset.

5.2 Finalized Predictive Models

Drawing from the results discussed in the preceding sections, we identified the following two predictive models (Random Forest and Fully Connected Neural Network) as the most effective for cyber-threat prediction. These models were selected based on rigorous evaluation, extensive experimentation, and hyperparameter tuning.

5.2.1 Random Forest Model

The RF model was selected because of its robustness nature and the ability it has in handling a imbalanced datasets effectively. After employing GridSearchCV, the op-

timal hyperparameters shown in Table 5.3 were identified. These parameters allowed the model to balance performance and generalization, minimizing overfitting.

Table 5.3: Best Hyperparameters for Random Forest Model

Parameters	**Value**
n_estimators	150
max_depth	10
min_samples_leaf	4

Furthermore, the RF model demonstrated high performance across various evaluation metrics, including accuracy, precision, recall, and F1 score in the average accuracy as is shown in Table 5.2.

5.2.2 Fully Connected Neural Network (FCNN) Model

The Fully Connected Neural Network (FCNN) was also a candidate for the finalized predictive model, owing to its flexibility in capturing the complex patterns that exist in the data. Table 5.4 summarizes the design parameters of the selected FCNN model.

Table 5.4: Summary of FCNN Models

Hidden Layers	**Activation**	**Optimizer**	**Learning Rate**	**Regularization (λ)**
4 (512, 256, 128, 64)	tanh	SGD	1×10^{-2}	0.0005

In addition, the selected FCNN model design is illustrated in Figure 5.7. The network comprises four hidden layers, with layer sizes seen to progressively decrease to encourage feature extraction and dimensionality reduction. A `tanh` activation function shown in Table 5.4 was chosen for its ability to handle both positive and negative values effectively, enabling better gradient flow. Also, the Stochastic Gradient Descent (SGD) optimizer was used in conjunction with a learning rate of 1×10^{-2} and a regularization parameter (λ) of 0.0005 to mitigate overfitting.

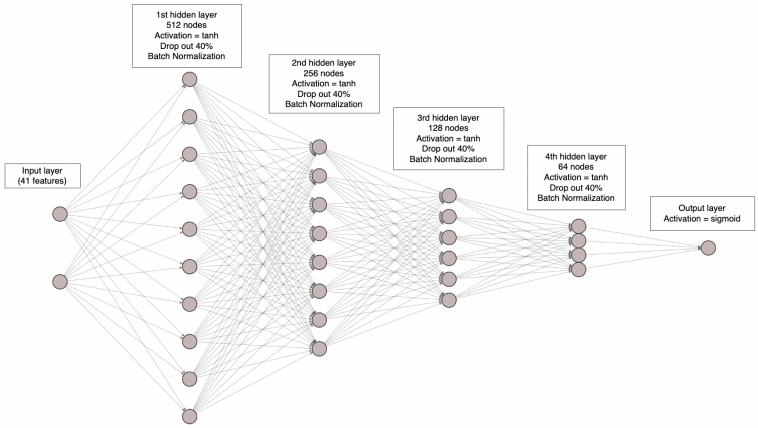

Figure 5.7: Illustration of the selected FCNN model design

5.2.3 Comparison and Practical Use

It has been observed that both the RF and FCNN models offer unique strengths, making them valuable algorithms for cyber threat detection. The RF model is easier to interpret and computationally efficient, making it suitable for scenarios that requires rapid predictions or resource-constrained environments. The model's ability to generate feature importance allows users to gain insights about the key factors impacting predictions, which can be useful in detection strategies and communicating findings to non-technical stakeholders.

On the other hand, the FCNN model is seen to be more versatile, particularly in handling datasets with high complexity. The FCNN's capacity to generalize across diverse and dynamic datasets enhances its reliability, especially in detecting sophisticated cyber threats that evade traditional detection mechanisms.

Together, these two models (RF and FCNN) provide a robust and efficient framework for accurate and reliable cyber threat prediction. The RF model can serve as a fast, lightweight baseline for initial detection, while the FCNN model can provide a deeper layer of prediction for complex or high-risk scenarios. This dual-model approach not only improves detection accuracy but also offers flexibility in adapting to various needs.

6 Conclusion

This section gives concluding remarks to the investigation on cyber threat prediction and analysis using machine learning algorithms. It summarizes the major findings, reviews the initially set research objectives and the research questions (See Section 1), discusses the main contributions and suggests future research. Based on the mentioned contributions and the prevailing limitations of this study, the authors of this thesis aim to give a clear standpoint of how the ML models that have been leveraged can be used in cyber threat predictive analysis, and also to identify potential areas for improvement that may need further investigation.

6.1 Review of Research Objectives

The research objectives that were previously outlined at the beginning of this research study are revisited in this section. Furthermore, we assess the extent to which the initially set objectives have been achieved. The main objectives of this research were to:

- **Obj 1: To explore the use of data analysis and machine learning techniques for analyzing cyber-threat data**

  This objective was addressed through the examination of various machine learning models. This included Random Forest (RF), Logistic Regression (LR), QDA, LDA, XGBoost, and Neural Networks (NN) on Microsoft malware dataset. Observations showed that these techniques can be applied to analyze and interpret cyber-threat data effectively. It is the authors opinion that by leveraging this approach, this research gave some insights into the prevalence of cyber threats and how they can be detected and classified.

- **Obj 2: To develop predictive models for anticipating emerging cyber-threats , evaluate and compare their accuracy and reliability:** To achieve this objective, the authors developed and tested multiple predictive models in order to determine how effective they would be in forecasting potential cyber threats. A number of evaluation criteria was used as well as fine-tuning the models hyperparameters using Cross Validation to find the best parameters for the models.

- **Obj 3: To assess the performance of these models to new, unseen data split, ensuring they perform well in real-world scenarios:** In order to meet this objective the authors evaluate the predictive models in new

unseen data split to make sure the models still perform well for new, unseen data.

Table 6.1: Objectives and Focus Areas

Objective	**Focus**	**Where Addressed**
1	Use of data analysis and ML to analyze cyber-threat data	Section 4
2	Development of predictive models for anticipating emerging cyber-threats, evaluate and compare their accuracy and reliability	Section 4, 5
3	To asses the performance of these models to new, unseen data split, ensuring they perform well in real-world scenarios	Section 5

6.2 Review of Research Questions

We revisit the research questions that guided this study. We begin by evaluating the outcome of the research questions based on the findings that we obtained from the preceding sections, where the main focus was on analöysis and predictioncyber threats We examined each research question based on the effectiveness of the predictive models that were leveraged, the accuracy of the predictive models and how reliable they may be. We give a summary on how these research questions were addressed.

Table 6.2: Review of Research Questions

Research Question	**Summary of Findings**	**Where RQ is Answered**
RQ 1	Try different Predictive ML-based methods to effectively analyzed cyber-threats.	Section 4.2
RQ 2	Predictive models were run 5 additional times in newly split train-test set to evaluate their performance in new, unseen data.	Section 5.1.2
RQ 3	The key features importance that can affect prediction and analysis of cyber threats has been explored	Section 4.1.4

6.3 Discussions of the Main Contributions

We address the and explore overarching objectives and research questions that were identified in Section 1 of this thesis as part of the contributions.

While it is pertinent to highlight that this research thesis significantly focuses on the intersection of the field of Data science and cybersecurity , we emphasize that the scope of the contribution can be assessed in depth, however, the main contributions in this thesis are discussed as follows:

- Exploring diverse ML models, like RF, LR, QDA, LDA, XGBoost, and Neural Networks, have shown how effective they can be in the analysis and prediction of cyber threat. These ML models that were leveraged provided insights into the prevalence and how chances of cyber-threat detection can be increased.

- The approach of developing and evaluating ML predictive models focused on predicting the presence of cyber-threats as is highlighted in (Obj 2) has shown a number of aspects. These aspects in the long run were assessed to be factors that are able to influence the accuracy and reliability aspects. Other aspects like feature selection, the complexity of the model, and the quality of the data was also identified to be critical in enhancing the predictive capabilities of ML models like RF and the FCNN.

- To emphasize the effectiveness of the predictive models that can be utilized as a framework or a guide for the development of robust and effective predictive systems that are or could be tailored for different sector.

6.4 Modules not Implemented

As part of the process of designing this research, a number of advanced modules were identified, to be potential additions to the potential enhancements to the cyber-threat detection systems, however these modules were not able to be implemented owing to a variety of constraints. These modules include real-time data streaming integration from multiple source systems, advanced anomaly detection algorithms, and extensive data analysis beyond basic metrics. While the implemented aspects of ML predictive model components provide a robust foundation for cyber-threat detection, adding and integrating these additional modules could have a significant enhancement of the systems' capabilities, this providing more detailed and real-time threat analysis approach that can help future threat section purposes.

6.5 Future Work

Identifying and developing a robust cyber-threat detection system needs precise approaches and the approach that we have used in this research thesis has been able to lay a solid foundation, however, our research has also identified avenues for future research work that can be used to enhance the propositions that have been suggested in this research thesis as follows:

- Integrating streaming real-time data that originates from multiple source systems can be explored to the greater extent of discovering real time cyber threats. In short, the real-time data integration, would not only pave way for an immediate detection and response to emerging threats, but it will also improve the security posture. To implement this, it would require developing scalable data pipelines and designing more efficient stream processing techniques that can align to the data pipelines.

- There is a need for developing advanced anomaly detection algorithms and also integrating them to the ML predictive models. Developing these algorithms

could be beneficial since they could utilize unsupervised learning methods, which is useful in the detection of previously unseen cyber threats. It is the authors' opinion that this could be particularly beneficial in the evolving cyber-threat landscape where we witness new attack vectors and constantly emerging zero-day threats.

- In spite of that, it would be beneficial to develop an extensive data analysis module that would help in the identification of the nature and origins of cyber-threats. This approach is envisaged to go beyond the basic performance metrics to perform advanced statistical analyses, visualization techniques, and trend analyses. This, would in the long run help in understanding the underlying patterns and behaviors of real-time and emerging cyber threats,in order to identify more suitable mitigation strategies.

- It would be pertinent to expand the dataset to include diverse range of cyber-threats and also to incorporate the extra features could improve the predictive accuracy of the models. For this to be effective, it would involve the collaboration of various cybersecurity bodies in order to obtain more representative data that can be understood well.

6.6 Final Conclusion

The study that has been conducted in this research thesis had an aim of addressing the problem of cyber threats leveraged machine learning approaches as a step towards detecting threats . This study explored the effectiveness of various predictive models like Random Forest, Logistic Regression, QDA, LDA, XGBoost, and the Neural Networks. This was also able to provide a better understanding of the capabilities of the identified approaches in a real-world context.

The main objectives that were identified in this research thesis have successfully been achieved through an evaluation and comparison of ML model performances. ML models like RF and FCNN have shown a higher accuracy and reliability for purposes for predicting cyber threats.

However, the study has also portrayed other areas that can be explored further. This include, the need for real-time data integration, advanced anomaly detection algorithms, and extensive data analysis that can help in the understanding of novel cyber-threats. The avenues for future work that can enhance is seen to be promising and with continued improvement which in the long run will lead in the detection and mitigation of cyber-based threats in different sector.

Ultimately, this research thesis has provided a contribution to the body of knowledge in the field of cybersecurity . It has provided a foundation for future research and potential practical applications. This has been achieved by way of addressing the evolving cyber-threat landscape based on intelligent analytical techniques. The ideas, insights gained from this study and the propositions that have been fronted, and the recommendations for future enhancements have laid a roadmap and a core foundation for continued advancements and exploration in this area of study.

Bibliography

[AHT17] Uchenna P Daniel Ani, Hongmei He, and Ashutosh Tiwari. Review of cybersecurity issues in industrial critical infrastructure: manufacturing in perspective. *Journal of Cyber Security Technology*, 1(1):32–74, 2017.

[BG98] Suresh Balakrishnama and Aravind Ganapathiraju. Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*, 18(1998):1–8, 1998.

[BHA09] Simon Bernard, Laurent Heutte, and Sebastien Adam. On the selection of decision trees in random forests. In *2009 International joint conference on neural networks*, pages 302–307. IEEE, 2009.

[BHBK06] Robert E Banfield, Lawrence O Hall, Kevin W Bowyer, and W Philip Kegelmeyer. A comparison of decision tree ensemble creation techniques. *IEEE transactions on pattern analysis and machine intelligence*, 29(1):173–180, 2006.

[BS16] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25:197–227, 2016.

[CHB+15] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[CVA06] Qi Cheng, Pramod K Varshney, and Manoj K Arora. Logistic regression for feature selection and soft classification of remote sensing data. *IEEE Geoscience and Remote Sensing Letters*, 3(4):491–494, 2006.

[DB09] Sarah J Dixon and Richard G Brereton. Comparison of performance of five common classifiers represented as boundary methods: Euclidean distance to centroids, linear discriminant analysis, quadratic discriminant analysis, learning vector quantization and support vector machines, as dependent on data structure. *Chemometrics and Intelligent Laboratory Systems*, 95(1):1–17, 2009.

[DQRT15] Diego Didona, Francesco Quaglia, Paolo Romano, and Ennio Torre. Enhancing performance prediction robustness by combining analytical modeling and machine learning. In *Proceedings of the 6th ACM/SPEC international conference on performance engineering*, pages 145–156, 2015.

[FCT20] Stefanos Fafalios, Pavlos Charonyktakis, and Ioannis Tsamardinos. Gradient boosting trees. *Gnosis Data Analysis PC*, 1, 2020.

[GMP20] Daniel Gibert, Carles Mateu, and Jordi Planes. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications*, 153:102526, 2020.

[GSL+21] Peng Gao, Fei Shao, Xiaoyuan Liu, Xusheng Xiao, Zheng Qin, Fengyuan Xu, Prateek Mittal, Sanjeev R Kulkarni, and Dawn Song. Enabling efficient cyber threat hunting with cyber threat intelligence. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 193–204. IEEE, 2021.

[Hel07] Tomas Hellström. Critical infrastructure and systemic vulnerability: Towards a planning framework. *Safety science*, 45(3):415–430, 2007.

[HMH+17] Shamsul Huda, Suruz Miah, Mohammad Mehedi Hassan, Rafiqul Islam, John Yearwood, Majed Alrubaian, and Ahmad Almogren. Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data. *Information Sciences*, 379:211–228, 2017.

[KAS23] Varol O Kayhan, Manish Agrawal, and Shivendu Shivendu. Cyber threat detection: Unsupervised hunting of anomalous commands (uhac). *Decision Support Systems*, 168:113928, 2023.

[KRM+18] Mahmoud Kalash, Mrigank Rochan, Noman Mohammed, Neil DB Bruce, Yang Wang, and Farkhund Iqbal. Malware classification with deep convolutional neural networks. In *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)*, pages 1–5. IEEE, 2018.

[LRM20] Valliappa Lakshmanan, Sara Robinson, and Michael Munn. *Machine learning design patterns*. O'Reilly Media, 2020.

[Mic24] Microsoft. Microsoft malware prediction. can you predict if a machine will soon be hit with malware? *Kaggle.com*, 0(0):., 2024.

[MJ18] Vasileios Mavroeidis and Audun Jøsang. Data-driven threat hunting using sysmon. In *Proceedings of the 2nd international conference on cryptography, security and privacy*, pages 82–88, 2018.

[MLA+24] Arash Mahboubi, Khanh Luong, Hamed Aboutorab, Hang Thanh Bui, Geoff Jarrad, Mohammed Bahutair, Seyit Camtepe, Ganna Pogrebna, Ejaz Ahmed, Bazara Barry, et al. Evolving techniques in cyber threat hunting: A systematic review. *Journal of Network and Computer Applications*, page 104004, 2024.

[MS19] Mohammad Masum and Hossain Shahriar. Droid-nnet: Deep learning neural network for android malware detection. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 5789–5793. IEEE, 2019.

[NBL+14] Jason RC Nurse, Oliver Buckley, Philip A Legg, Michael Goldsmith, Sadie Creese, Gordon RT Wright, and Monica Whitty. Understanding insider threat: A framework for characterising attacks. In *2014 IEEE security and privacy workshops*, pages 214–228. IEEE, 2014.

[NGK+06] Scott A Neslin, Sunil Gupta, Wagner Kamakura, Junxiang Lu, and Charlotte H Mason. Defection detection: Measuring and understanding the predictive accuracy of customer churn models. *Journal of marketing research*, 43(2):204–211, 2006.

[NK21] Ahmed Nassar and Mostafa Kamal. Machine learning and big data analytics for cybersecurity threat detection: A holistic review of techniques and case studies. *Journal of Artificial Intelligence and Machine Learning in Management*, 5(1):51–63, 2021.

[PAMM21] Venkata Reddy Palleti, Sridhar Adepu, Vishrut Kumar Mishra, and Aditya Mathur. Cascading effects of cyber-attacks on interconnected critical infrastructure. *Cybersecurity*, 4:1–19, 2021.

[PIL06] Anantha M Prasad, Louis R Iverson, and Andy Liaw. Newer classification and regression tree techniques: bagging and random forests for ecological prediction. *Ecosystems*, 9:181–199, 2006.

[PTRC07] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.

[RM18] Robert S Radvanovsky and Allan McDougall. *Critical infrastructure: homeland security and emergency preparedness*. crc press, 2018.

[RTP+23] Hugo Riggs, Shahid Tufail, Imtiaz Parvez, Mohd Tariq, Mohammed Aquib Khan, Asham Amir, Kedari Vineetha Vuda, and Arif I Sarwat. Impact, vulnerabilities, and mitigation strategies for cyber-secure critical infrastructure. *Sensors*, 23(8):4060, 2023.

[Sar21] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3):160, 2021.

[Sea22] Jim Seaman. Cyber threat prediction and modelling. In *Artificial Intelligence and National Security*, pages 113–156. Springer, 2022.

[SMT09] Carolin Strobl, James Malley, and Gerhard Tutz. An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological methods*, 14(4):323, 2009.

[SVSS15] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4580–4584. Ieee, 2015.

[SZZZ19] Moting Su, Zongyi Zhang, Ye Zhu, and Donglan Zha. Data-driven natural gas spot price forecasting with least squares regression boosting algorithm. *Energies*, 12(6):1094, 2019.

[TSYQ05] E Ke Tang, Ponnuthurai N Suganthan, Xin Yao, and A Kai Qin. Linear dimensionality reduction using relevance weighted lda. *Pattern recognition*, 38(4):485–493, 2005.

[Uli07] Mihaela Ulieru. Design for resilience of networked critical infrastructures. In *2007 Inaugural IEEE-IES Digital EcoSystems and Technologies Conference*, pages 540–545. IEEE, 2007.

[VAS+19] R Vinayakumar, Mamoun Alazab, KP Soman, Prabaharan Poornachandran, and Sitalakshmi Venkatraman. Robust intelligent malware detection using deep learning. *IEEE access*, 7:46717–46738, 2019.

[VBF21] Stefan Varga, Joel Brynielsson, and Ulrik Franke. Cyber-threat perception and risk management in the swedish financial sector. *Computers & security*, 105:102239, 2021.

[YHD+22] Fengyu Yang, Yanni Han, Ying Ding, Qian Tan, and Zhen Xu. A flexible approach for cyber threat hunting based on kernel audit records. *Cybersecurity*, 5(1):11, 2022.

[YHV+11] Xiaowei Yu, Juha Hyyppä, Mikko Vastaranta, Markus Holopainen, and Risto Viitala. Predicting individual tree attributes from airborne laser point clouds based on the random forests technique. *ISPRS Journal of Photogrammetry and remote sensing*, 66(1):28–37, 2011.

[YKP+23] Abbas Yazdinejad, Mostafa Kazemi, Reza M Parizi, Ali Dehghantanha, and Hadis Karimipour. An ensemble deep learning model for cyber threat hunting in industrial internet of things. *Digital Communications and Networks*, 9(1):101–110, 2023.

[YOIL+21] Abel Yeboah-Ofori, Shareeful Islam, Sin Wee Lee, Zia Ush Shamszaman, Khan Muhammad, Meteb Altaf, and Mabrook S Al-Rakhami. Cyber threat predictive analytics for improving cyber supply chain security. *IEEE Access*, 9:94318–94337, 2021.

A (Appendix)

Table A.1: Variables list of dataset Microsoft Malware Prediction

Feature	**Description**
MachineIdentifier	Individual machine ID
ProductName	Defender state information e.g. win8defender
EngineVersion	Defender state information e.g. 1.1.12603.0
AppVersion	Defender state information e.g. 4.9.10586.0
AvSigVersion	Defender state information e.g. 1.217.1014.0
IsBeta	Defender state information e.g. false
RtpStateBitfield	NA
IsSxsPassiveMode	NA
DefaultBrowsersIdentifier	ID for the machine's default browser
AVProductStatesIdentifier	ID for the specific configuration of a user's antivirus software
AVProductsInstalled	NA
AVProductsEnabled	NA
HasTpm	True if machine has tpm
CountryIdentifier	ID for the country the machine is located in
CityIdentifier	ID for the city the machine is located in
OrganizationIdentifier	ID for the organization the machine belongs in
GeoNameIdentifier	ID for the geographic region a machine is located in
LocaleEnglishNameIdentifier	English name of Locale ID of the current user
Platform	Calculates platform name (of OS related properties and processor property)
Processor	This is the process architecture of the installed operating system
OsVer	Version of the current operating system
OsBuild	Build of the current operating system
OsSuite	Product suite mask for the current operating system.
OsPlatformSubRelease	Returns the OS Platform sub-release
OsBuildLab	Build lab that generated the current OS.
SkuEdition	To use the Product Type defined in the MSDN to map to a 'SKU-Edition' name.
IsProtected	This is a calculated field derived from the Spynet Report's AV Products field.
AutoSampleOptIn	This is the SubmitSamplesConsent value passed in from the service
PuaMode	Pua Enabled mode from the service
SMode	This field is set to true when the device is known to be in 'S Mode'
IeVerIdentifier	NA
SmartScreen	This is the SmartScreen enabled string value from registry.
Firewall	This attribute is true (1) for Windows 8.1 and above if windows firewall is enabled.
UacLuaenable	This attribute reports whether or not the "administrator in Admin Approval Mode" user type is disabled or enabled in UAC.
Census_MDC2FormFactor	A grouping based on a combination of Device Census level hardware characteristics.
Census_DeviceFamily	AKA DeviceClass.
Census_OEMNameIdentifier	NA
Census_OEMModelIdentifier	NA
Census_ProcessorCoreCount	Number of logical cores in the processor
Census_ProcessorManufacturerIdentifier	NA
Census_ProcessorModelIdentifier	NA
Census_ProcessorClass	A classification of processors into high/medium/low. Initially used for Pricing Level SKU. No longer maintained and updated
Census_PrimaryDiskTotalCapacity	Amount of disk space on primary disk of the machine in MB
Census_PrimaryDiskTypeName	Friendly name of Primary Disk Type - HDD or SSD
Census_SystemVolumeTotalCapacity	The size of the partition that the System volume is installed on in MB
Census_HasOpticalDiskDrive	True indicates that the machine has an optical disk drive (CD/DVD)
Census_TotalPhysicalRAM	Retrieves the physical RAM in MB
Census_ChassisTypeName	Retrieves a numeric representation of what type of chassis the machine has. A value of 0 means xx
Census_InternalPrimaryDiagonalDisplaySizeInInches	Retrieves the physical diagonal length in inches of the primary display
Census_InternalPrimaryDisplayResolutionHorizontal	Retrieves the number of pixels in the horizontal direction of the internal display.
Census_InternalPrimaryDisplayResolutionVertical	Retrieves the number of pixels in the vertical direction of the internal display
Census_PowerPlatformRoleName	Indicates the OEM preferred power management profile. This value helps identify the basic form factor of the device
Census_InternalBatteryType	NA
Census_InternalBatteryNumberOfCharges	NA
Census_OSVersion	Numeric OS version Example - 10.0.10130.0
Census_OSArchitecture	Architecture on which the OS is based. Derived from OSVersionFull. Example - amd64
Census_OSBranch	Branch of the OS extracted from the OsVersionFull.
Census_OSBuildNumber	OS Build number extracted from the OsVersionFull. Example - OsBuildNumber = 10512 or 10240
Census_OSBuildRevision	OS Build revision extracted from the OsVersionFull. Example - OsBuildRevision = 1000 or 16458
Census_OSEdition	Edition of the current OS.
Census_OSSkuName	OS edition friendly name (currently Windows only)
Census_OSInstallTypeName	Friendly description of what install was used on the machine i.e. clean
Census_OSInstallLanguageIdentifier	NA
Census_OSUILocaleIdentifier	NA
Census_OSWUAutoUpdateOptionsName	Friendly name of the WindowsUpdate auto-update settings on the machine.
Census_IsPortableOperatingSystem	Indicates whether OS is booted up and running via Windows-To-Go on a USB stick.
Census_GenuineStateName	Friendly name of OSGenuineStateID. 0 = Genuine
Census_ActivationChannel	Retail license key or Volume license key for a machine.
Census_IsFlightingInternal	NA
Census_IsFlightsDisabled	Indicates if the machine is participating in flighting.
Census_FlightRing	The ring that the device user would like to receive flights for.
Census_ThresholdOptIn	NA
Census_FirmwareManufacturerIdentifier	NA
Census_FirmwareVersionIdentifier	NA
Census_IsSecureBootEnabled	Indicates if Secure Boot mode is enabled.
Census_IsWIMBootEnabled	NA
Census_IsVirtualDevice	Identifies a Virtual Machine (machine learning model)
Census_IsTouchEnabled	Is this a touch device ?
Census_IsPenCapable	Is the device capable of pen input ?
Census_IsAlwaysOnAlwaysConnectedCapable	Retreives information about whether the battery enables the device to be AlwaysOn-AlwaysConnected.
Wdft_IsGamer	Indicates whether the device is a gamer device or not based on its hardware combination.
Wdft_RegionIdentifier	NA