**TECHNICAL COMPUTING**

**DISSERTATION RESEARCH PROJECT PROPOSAL <academic year>**

**Artificial Intelligence and Machine Learning in Malware Detection**

**Student name**

**Student number**

**Supervisor**

**Date**

# **Abstract**

With the emergence of new threats the traditional methods of detection have proven to be insufficient to protect users from novel forms of malicious software. The goal of this research is to improve Android malware detection by embracing the services of Artificial Intelligence and Machine learning. By examining and comparing the results of applying Artificial Intelligence and Machine Learning models, Random Forests, Support Vector Machines, and Neural Networks particularly, this paper substantiates the advantage of applying these techniques over standard methods. In particular, Random Forest shows exceptionally high, 99% accuracy, as well as considerable resilience in identifying malware. This research also looks at how to extend the hybrid models with a better detection accuracy and robust against adversarial attacks. To conclude the computer-based experiment is used to validate the proposed framework and demonstrate its ability to improve malware detection and have a profound impact on the further development of cybersecurity. Therefore, there is a need to incorporate the use of Artificial Intelligence and Machine Learning to create optimized and effective real-time Malware detection systems that can effectively cope with the growing and dynamic nature of threats present in cyber-space.

Keywords: Android malware, Artificial Intelligence, Machine Learning, malware detection, Random Forest, Support Vector Machine, cybersecurity, neural networks

# **Ethical consideration**

In this context of the research that deals with Android malware detection using AI and ML approaches, it is necessary to discuss the ethical issues that relate to the employment of such technologies. This work is particularly interested in how best and most ethically to conduct research where data is sensitive, or where there may be privacy or security concerns. Due to the fact that this kind of research requires the collection and analyzing data which may be preferable to user conduct issues and device’s usage appropriate measures were taken to ensure all the stake holders were protected by their rights and Privacy. Every effort was made during the data collection process so as not to prejudice any individuals, and no data that could identify any individual was collected for processing. More particularly, in relation to datasets that might contain sensitive data , the study ensured that all data was reported in an aggregated form or by the use of pseudonyms thus maintaining the anonymity of the respondents. Furthermore, the study considered general ethical concerns relating to development of AI and ML models for purpose of cybersecurity. Unintended uses of such technologies as well as problems associated with the generation of incorrect or prejudiced model predictions were not overlooked. To minimize such risks, the research followed principles of ethics that was unveiled for the design and use of AI and ML models. Furthermore, the study adhered to the guidelines of ethical conduct as provided by the applicable Research Ethics Committee (REC) so as to ensure that all the research conduct came with the respect of the rights to the individuals and other organizations likely to be affected regarding this work. Incorporating these ethical considerations into the course of the research, this work is seeking not only to contribute to the field of Android malware detection, but also to do this in the most ethical manner possible. The following are the main ethical practices to be upheld so as to make sure that the research is very useful in generating useful information while protecting the interest of all the individuals interested in the study.

Table of Contents

[**Abstract** 2](#_Toc175309475)

[**Ethical consideration** 3](#_Toc175309476)

[**List of Figure** 6](#_Toc175309477)

[**List of Tables** 7](#_Toc175309478)

[**Chapter 1: Introduction** 8](#_Toc175309479)

[Overview 8](#_Toc175309480)

[Importance of the Problem 9](#_Toc175309481)

[Scope of the Study 9](#_Toc175309482)

[Structure of the Dissertation 9](#_Toc175309483)

[Background and Motivation 10](#_Toc175309484)

[Problem in Brief (Research Problem and Research Question) 10](#_Toc175309485)

[Aim 10](#_Toc175309486)

[Objectives 11](#_Toc175309487)

[Objectives 11](#_Toc175309488)

[Research Significance 11](#_Toc175309489)

[Academic Importance 11](#_Toc175309490)

[Practical Relevance 11](#_Toc175309491)

[Originality 11](#_Toc175309492)

[**Chapter 2: Literature Review** 12](#_Toc175309493)

[Traditional Malware Detection Systems 13](#_Toc175309494)

[AI and ML-Based Malware Detection 13](#_Toc175309495)

[Supervised Learning 14](#_Toc175309496)

[Unsupervised Learning 14](#_Toc175309497)

[Deep Learning 14](#_Toc175309498)

[State-of-the-Art Works 15](#_Toc175309499)

[High Accuracy Detection of Mobile Malware Using Machine Learning 15](#_Toc175309500)

[Automated Machine Learning for Deep Learning-Based Malware Detection 15](#_Toc175309501)

[Comparative Analysis 15](#_Toc175309502)

[Gap in Literature 17](#_Toc175309503)

[**Chapter 3: Research Methodology** 18](#_Toc175309504)

[Data Source: 18](#_Toc175309505)

[Design 19](#_Toc175309506)

[Design Techniques 19](#_Toc175309507)

[Proposed Solution: Artificial Intelligence / Machine Learning Framework for Malware Detection 19](#_Toc175309508)

[Type of Architecture: 19](#_Toc175309509)

[System Overview 20](#_Toc175309510)

[**Chapter 4 - Analysis / Experiments** 20](#_Toc175309511)

[Product Implementation/Artifact 20](#_Toc175309512)

[Technical Description of Implementation 20](#_Toc175309513)

[Libraries and Interfaces 21](#_Toc175309514)

[Functions Used 21](#_Toc175309515)

[Application of Coding Principles 22](#_Toc175309516)

[Initial Code Structure 23](#_Toc175309517)

[**Initial Code of Artificial Intelligence and Machine Learning in Malware Detection** 24](#_Toc175309518)

[Experimental Setup 24](#_Toc175309519)

[Model Training and Evaluation 25](#_Toc175309520)

[Random Forest Model 25](#_Toc175309521)

[SVM Model 25](#_Toc175309522)

[Neural Network Model 26](#_Toc175309523)

[Validation 26](#_Toc175309524)

[**Chapter 5 - Discussion and Conclusion** 27](#_Toc175309525)

[Discussion 27](#_Toc175309526)

[**Random Forest Model**: 28](#_Toc175309527)

[**SVM Model:** 28](#_Toc175309528)

[**Neural Network Model:** 28](#_Toc175309529)

[Validation Strategies 28](#_Toc175309530)

[Conclusion 29](#_Toc175309531)

[Future Work: 29](#_Toc175309532)

[**References** 30](#_Toc175309533)

[**Appendices** 32](#_Toc175309534)

[Annex 1 - Project Plan 32](#_Toc175309535)

[User Guide 34](#_Toc175309536)

# **List of Figure**

[Figure 3.1: Flowchart 22](#_Toc170336029)

**List of Tables**

[Table 2.1: Comparative Analysis 16](#_Toc170336025)

[Table 4.1 Random Forest Results 26](#_Toc170336026)

[Table 4.2 SVM 27](#_Toc170336027)

[Table 4.3 : NN results 27](#_Toc170336028)

# **Chapter 1: Introduction**

## **Overview**

The use of mobile devices especially those that run on Android is among the most striking instances of people’s social interaction with new technologies. These devices have more than five million applications unveiled on social networks like Google Play, and they help carrying out numerous activities in people’s daily lives, including but not limited to communication, recreation, purchases and payments, and schedules. (Sharma and Kaul 2024) While this has proved advantageous to numerous organizations, it has on the other hand imposed a great deal of risk that is otherwise manageable if checked. Objectively, the Android openness is one of its principal advantages as well as the curse that makes it vulnerable to hackers. This paper is centered around the present and future status of malware attacks on the Android platform and the application of AI and ML in addressing and avoiding it.

## **Importance of the Problem**

Consequently, this paper has a number of implications beyond the merely scholarly. Mobile malware poses severe threats that affect individuals, business entities as well as governments in terms of identity theft, identity cloning, financial thefts, and unauthorized access to important information (Mishra, Gupta and Gupta 2018). As the subsequent tendencies show, mobile malware events are increasing rapidly, and the European region only since February 2022 has experienced a 500 % growth (Reijonen 2024). And this is why new and more efficient methods of rasterization are needed, in order to effectively combat hacking reaching the later stages. The solution to this problem is important not only in order to eliminate threats and ensure safety, but also to minimize possible risks that can affect users’ confidence in mobile technologies.

## **Scope of the Study**

This study aims at proposing and assessing a broad framework for malware detection using Artificial Intelligence and Machine Learning algorithms. The work is devoted to the application and evaluation of the results of using Support Vector Machines, Artificial Neural Networks, and Random Forests in identifying malicious programs. Support Vector Machines are used due to the capability of handling high dimensional data to perform classification by using a best separating hyperplane for clustering of both the malicious and the benign applications. Neural Networks, especially Deep Learning models, are used due to the capability of extracting the features and learning complicated patterns to raw data, overcoming the low Levels of defensibly. Random forests are applied because the algorithm used is ensemble learning which means there are a lot of decision trees that are made to work together to strengthen detection of tumors and to make it more generalized. The detection of Android malware in this study therefore has the purpose of creating an efficient and effective method of identifying the malware for an ever-evolving Android platform.

## **Structure of the Dissertation**

The dissertation is structured as follows:

Introduction: The purpose of this section is to define the research problem as well as the importance of the study, manuscript, and scope, and the overall research methodology.

Background and Motivation: This chapter aims at giving the reader background knowledge on the Android malware problem and presenting the goals of this work.

Problem in Brief: this section defines the research problem and questions in detail; they also mention the aim and objectives of the study.

Literature Review: This chapter is a literature review, which presents an overview of the studies considering the malware detection approaches grouped into static analysis, dynamic analysis, and hybrid analysis.

Methodology: This part describes the methods undertaken in the research including the research design, data collection techniques and analysis techniques.

Analysis and Findings: This chapter details the empirical studies and results regarding several AI and ML models’ performance.

Discussion: This chapter offers an interpretation of the findings specifying how these findings inform theory and practice and points out the potential avenues for research.

Conclusion: The last chapter reiterates the major findings of the research and the sources of bias as well as the possibilities for and implications of furthering the study.

Through this dissertation, the reader would have gotten a clear understanding of Android malware detection at the current state with the inclusion of AI and ML as applicable methods, as well as the problem areas that call for enhancement to create a greater future for the field.

## **Background and Motivation**

Android operating system that premiered in 2008 has taken a big chunk of the market share of mobile devices and because of its open-source architecture, it has attracted considerable attention of malware writers. On the one hand, this openness contributes to creativity and differentiation, on the other – weak points that can cause critical threats to the system since the patient organizes incursions and puts viruses containing criminal code into applications that appear as normal, thereby gaining access to restricted data, financial abuses, or operating system failures. Solutions like the static analysis of the file, which implies examining the code without executing it, is no longer effective against complex malware that employ methods of code obscuring; dynamic analysis that involves executing the app in a controlled environment is somewhat better but still insufficient. The AI or ML has better solutions for performing the verity of actions which distinguish between state and non-state actors/malicious and legitimate healthcare activities. However, these methods also have its own problem such as, the require large amount of labeled data and sensitive to adversarial attack. Due to the growing complexity of Android malware, the ineffectiveness of the current methods of detection, the universality of the use of AI and ML this research intends to propose ways or a novel approach to improve the accuracy, speed and evasion of the current AI and ML countermeasures, against malicious Android applications thus creating better and more efficient ways of coping with the ever-evolving menace of cyber threats.

## **Problem in Brief (Research Problem and Research Question)**

### Aim

The aim of this project is to develop a hybrid system that integrates static and dynamic analyses with machine learning to improve Android malware detection and effectively eliminate vulnerabilities.

## **Objectives**

### Objectives

* To critically review the problem domain of Android malware detection.
* To study and evaluate technologies that can solve the problem.
* To design and develop a system for solving the problem.
* To evaluate the proposed system.
* To create and validate the proposed system.

## **Research Significance**

### Academic Importance

This research benefits the body of academic knowledge in the field of Android malware detection as the study aims at presenting the state of knowledge in the current literature. It enriches the knowledge regarding the use of AI and ML in improvement of detection procedures by presenting new ideas and approaches. The study also points out the limitations of the previous work and indicate the directions for the future research that would enhance the security more.

### Practical Relevance

From a more pragmatic perspective, the outcomes of the present research are useful for designing better malware identification tools. Thus, optimizing the detection accuracy and system robustness, the proposed system can prevent users from such attacks and keep their records and money safe. However, it is crucial for organizations that use mobile technologies for their commerce, as developed security tools can help avoid devastating attacks and preserve customers’ confidence.

### Originality

This study is original in its development of a multidetector framework that integrates the latest advancements in Artificial Intelligence and Machine Learning for malware detection. With multiple detection methods combined together as a unified hybrid model, the approach is more effective than applying each if them as a standalone system. It also helps improve the accuracy and efficiency of the malware detection improving the resistance of the system against new forms of malicious programs. The study is not only confirming the effectiveness of this hybrid model but also presents the ideas and the methodologies that can be used as a base for the creation of further works and can contribute to advanced studies and developments within the sphere of cybersecurity. Thus, this novel approach for malware detection can be considered to be a new benchmark for improving cybersecurity of digital systems.

# **Chapter 2: Literature Review**

Technological enhancement has been on the rise, and although it has generated general benefits, it has also resulted in development of hard to counter malware; a major threat to information security. (Yaqoob, et al. 2017) Malware is a term used to describe a software meant to infiltrate, damage and or exploit computer systems and has grown in its complexity over the years. First two types of malware detection include the signature-based and the heuristic based and these have proved to be incapable of dealing with the new change in malware. Anti-virus based on signatures scans for a specific pattern characteristic of a known virus and cannot detect new versions of the virus or entry (Zero-Day Attack). Behavioral and characteristic based approach is heuristic-based approaches that try to detect malware by reflecting on its behavior and it proves useful in detecting new threats but comes with many false alarms. (Creech 2014)

That is why Artificial Intelligence (AI) and Machine Learning (ML) have been identified as accurate approaches to improve the detecting of malware. Machine and Deep learning have capabilities to ingest and analyze big data, discerning interactions, and modification as per new attack vectors, without being coded in a specific way. These approaches are evolutional advantages above the traditional methods which can detect more cases with less false positive results. (Liu, et al. 2022)

Thus, this literature review presents and discusses a broad spectrum of methodologies using AI and ML in malware detection and their advantages and disadvantages. It also does provide a comparative analysis of the state-of-art works in the field that gives the reader an idea about current trends and further working in the domain.

## **Traditional Malware Detection Systems**

The traditional approaches to detect malware are mainly based on the detection of known signatures and heuristics techniques.

In this approach, the anti-malware tool follows a set of characteristics and when a virus matches with one of those Characteristics, it is flagged for removal. (Wressnegger, et al. 2017)This approach involves the comparison of a program with the known characteristics or signatures of a malware, these are sequences of bytes. These signatures are archived in a database and the detection system works against the database to look for infected files. A major advantage of signature-based detection is the feature that it possesses high accuracy. It is very efficient when it comes to well-known threats; once the signature of the malware is found the system can quickly and effectively identify and eliminate the malware. This method is also less resource consuming in aspect of computing tool as the process of matching files with the database of the signature takes relatively average time and does not need powerful computations. If the malware is already known, the signature-based scanning gives almost cent percent successful results with very minimum false alarms and hence is used as the first layer of scanning. (Venugopal and Hu 2008)

However, signature-based detection has the following significant limitations. Its main disadvantage is that it cannot detect zero-day threat or in other words newly developed threats whose signatures are not known in the wild. Since this approach invites the scanner to compare the contents of a file with a set of registered signatures, it cannot recognize new type of malware. (Bagaa, et al. 2020)

Due to some of the flaws associated with the signature-based detection, heuristic-based approaches were created. Heuristic-based detection works by detecting the artifacts of malware rather than the actual malware itself; it detects it based on the exhibited patterns it. It looks at the activity that programs are performing to determine their suspicious activity for things like changed files, network connections, and system resources. The latter gives the possibility to identify new and previously unknown malicious programs, which help protect against threats of the zero-day type. (Yunmar, Kusumawardani and Mohsen 2024)

## **AI and ML-Based Malware Detection**

Traditional methods are no longer sufficient due to which these new techniques such as AI and ML approaches have been incorporated. Such systems are capable of training on big data, finding patterns and altering their functioning themselves without any prior coding. Different methods are used, and these are the supervised learning, unsupervised learning, and the deep learning. (Brown, Gupta and Abdelsalam 2024)

### Supervised Learning

Supervised learning is a technique of training a model over a given data set such that the model is provided with input features and their corresponding outputs. Some of the algorithms that are commonly applied in the decision-making process include; Support Vector Machines (SVM), Decision Trees and additionally the Random Forests. (Kotsiantis, Zaharakis and Pintelas 2007)Such models have been reported to have high accuracy in the detection of malware, since they are trained on the patterns that define the training data.

Supervised learning models also depend heavily on big data sets that are well classified for training. The working of these models to a great extent is conditioned by the quality and volumes of training data. The principles like those of cross-validation and hyperparameter tuning are also important to allow the enhancements of model accuracy. (Abdolrasol, et al. 2021)

### Unsupervised Learning

Some of the categories of unsupervised learning models involve finding structures in datasets which do not have pre-identified classes. Some of the most common approaches of clustering are the K-means and another group of algorithms that is called anomaly detection, which is used in the context of malware detection. (Ghahramani 2003) These methods are of most help when looking for new and previously unencountered types of malwares.

This is because the unsupervised learning techniques may discover some statistical outliers in the data set that may suggest case of malware. (Yaqoob, et al. 2017) These methods are not supervised, and that can prove beneficial especially when the environments in which data mining is being done does not support labeling. However, they are less accurate most of the times compared to the supervised methods and are also harder to understand.

### Deep Learning

Deep learning is a special type of ML that utilizes deep neural networks with more than one layer to infer and learn the pattern of data. CNNs and RNNs are popular architectures in this field where the data input is most commonly images. Convolutional neural networks and recurrent neural networks have been reported to produce better results than traditional machine learning models for malware detection because deep learning models are capable of learning features from the raw input data deeply. (Dhruv and Naskar 2020)

The implementation and fine-tuning of deep learning models entail large resources and massive data sets. Some of such problems include low accuracy and incomplete training of the machine learning models, and the following tricks can be used to reduce these effects; Deep learning also has issues with overfitting, this is also manageable if one considers the techniques like regularization and selective models. (Yaqoob, et al. 2017)

## **State-of-the-Art Works**

### High Accuracy Detection of Mobile Malware Using Machine Learning

The core area of study by Yerima Suleiman is to classify and develop methodologies to get rid of mobile malware via AI. The study focuses on the effectiveness of the ML algorithms in identifying the acts of the malicious mobile applications. The use of Random Forests and Gradient Boosting Machines is applied in the classification of the applications as either benign or malicious according to the study. (Hasan, et al. 2019)The dataset consists of permissions, API calls, in and out pointing to network traffic, among others which are derived from mobile applications. The authors mentioned that in their proposed approach the accuracy was high and varied between 95% and 99%; thus, supporting the importance of using ML for discovering malicious mobile applications.

From Suleiman’s work, it indicates how ML has been applied in the dynamic and fluctuating mobile platform that which other traditional normal signature-based system encounters challenges regularly. In that way, using behavioral characteristics instead of specific signatures, which the ML algorithms are able to identify new and previously unknown threats. The study reveals that the use of AI and ML can enhance the cybersecurity practices in the context of the mobile environment tremendously. (Yerima 2023)

### Automated Machine Learning for Deep Learning-Based Malware Detection

Briefly, the work of Austin Brown, Maanak Gupta, and Mahmoud Abdelsalam indicates how AutoML can be applied to build deep learning-based malware detection models. NAS for AutoML expects to require minimal domain knowledge in structuring statistical models like deep learning for optimization with the help of computational tools and hyperparameters.

Indeed, the study offers a detailed examination of applying AutoML in static as well as online modes of malware detection. Static analysis refers to the analysis of the properties of executable files without actually running them, online analysis in the other hand is the study of systems’ behavior as they are in real time operation. The study under review reveals the promising prospect of AutoML for optimizing the process of creating efficient deep learning-based detectors of malware with relatively low computational complexity.

The authors’ presented study focuses on comparing different AutoML toolkits and their executions in the malicious software identification problems. They show that AutoML can cut the time needed to develop deep learning models while also cutting the depth of knowledge needed to achieve it, thus opening the door to presenting new and highly effective ways of warding of malware. (Brown, Gupta and Abdelsalam 2024)

### Comparative Analysis

The following table summarizes the state-of-the-art works in AI and ML-based malware detection:

**Table 2.1: Comparative Analysis (Source: Author)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Study | Approach | Dataset | Key Findings | Strengths | Limitations |
| Kolosnjaji et al. (2016) | Deep Learning (CNN) | Malware Classification Challenge | Achieved high accuracy in malware classification. | High accuracy, effective feature extraction. | Requires large datasets, computationally expensive. |
| Rieck et al. (2011) | Unsupervised Learning | Behavioral Malware Dataset | Effective in detecting unknown malware through behavior analysis. | No need for labeled data, adaptable to new threats. | Lower accuracy compared to supervised methods, interpretability issues. |
| Saxe and Berlin (2015) | Neural Networks | Dynamic Analysis Dataset | Demonstrated superior performance in dynamic malware analysis. | High accuracy, robust against obfuscation techniques. | High computational cost, potential overfitting. |
| Ye et al. (2017) | SVM and KNN | VirusShare | Combined multiple ML techniques for improved detection. | Enhanced accuracy through ensemble methods. | Requires careful parameter tuning, sensitive to feature selection. |
| Anderson and Roth (2018) | Adversarial Training | Custom Dataset | Improved robustness against adversarial attacks. | Enhanced security, mitigates adversarial threats. | Complex implementation, increased training time. |
| Yerima (2020) | Machine Learning | Mobile Malware Dataset | High accuracy in mobile malware detection. | Effective for dynamic environments, robust model. | Limited to mobile platforms, potential for high false positives. |
| Brown, Gupta, and Abdelsalam (2021) | AutoML, Deep Learning | Static and Online Malware Datasets | Reduced domain expertise needed, effective for various datasets. | Low computational overhead, streamlined model development. | Limited research, initial setup complexity. |
| Pascanu et al. (2015) | RNN | Dynamic Malware Analysis | Effective in capturing temporal patterns in malware | Handles temporal dependencies well, high accuracy. | High computational cost, requires large datasets. |

This shifted approach of malware analysis based both on AI and ML is a major advancement over the conventional tools. Across all the studies there is a high accuracy and flexibility of the supervised, unsupervised, and deep learning models of new and known viruses. Nonetheless, some of the issues, for instance, computational cost, the requirement of large datasets, and vulnerability to adversarial examples, are still challenges to this day. Therefore, it can be concluded that it is necessary to develop the further research directions to merely optimize the given models and make them more effective and resistant. The main benefits of executing AutoML is that it may lessen the need for configuring the models which may need a lot of expertise and, on the other hand, protecting the AI/ML systems from adversarial attacks is crucial. (Alam 2023)

The comparisons of the current information systems include inherent advantages and limitations that contain significant information for improving current forms and styles used in the models and methodologies of malware detection. It will also perpetuate the analysis of modern approaches to improve the strong foundations and efficiency of the malware detection system, with a view to boosting the efficiency of security measures in cyberspace.

## **Gap in Literature**

However, several critical issues in the field of AI and ML for malware detection, which has been explored in this paper, are missing from the current literature. Firstly, although the current study has achieved considerable advances in increasing the model detection accuracy, there is a lack of efficient approaches that enable these models to update and learn the new malware representatives even if they are rather different from the previously known ones, without long retraining periods. This is highly important given that threat types on the cyberspace change from time to time. Secondly, the problem of adversarial robustness is another major deficit. Existing models can be easily tampered with using adversarial attacks and hence, the need to shall continue research on improving immunity of AI/ML based Malware detection systems against such heinous attacks. (Kroemer, Niekum and Konidaris 2021)

Third, the area where more works are needed is the constant monitoring and (counter)action systems. These systems could incorporate the use of AI/ML to alert organization as soon as a malware is detected and prevent it from causing an unfavorable impact on the organization. In addition, interpreting the AI/ML models used in malware detection is still a challenge may be due to the black box nature of the AI/ML models used. It is essential for cybersecurity specialists to know why a decision was made with respect to threats in order to get a handle on them. (Milosevic, Ferrante and Malek 2016) Studies can be carried out in the enhancement of the interpretability of the models that could help in the analysis of the behavior of malware.

Thus, another research gap is to improve AI/ML models’ resource efficiency or, in other words, the reduction of resource consumption. This entails lowering the complexity, thus the computational requirements, and the memory used which makes them feasible for deployment in environments that are resource constrained. (Iftikhar, et al. 2023). Mitigation of these gaps will go a long way in improving and strengthening the AI and ML-based malware detection as well as the cybersecurity measures against different forms of malicious cyber-attacks.

# **Chapter 3: Research Methodology**

This section provides details on the approach that was used to develop each component of the study in a systematic manner which enables the study to yield valid and reliable results that responds to the research aim, objectives and research questions. The research methodology involves factors such as the type of data to be collected, the sample of data to be selected, way of data collection and ways of analyzing the data.

## **Data Source:**

* **Kaggle Dataset:** The resource of the primary dataset for this study is the Kaggle, an open data science platform popular for offering numerous datasets in addition to hosting data science competitions. Kaggle offers a diverse add extensive data set that includes samples of labeled samples with a focus on both benign and malicious software. To enrich the training and testing of the Machine Learning Models, this dataset is compiled with lot of care and concentration is given to them.

## **Design**

### Design Techniques

In the construction of our Malware detection system, several advanced methods are integrated to provide an efficient solution. We employed flow charts as a way to establish a graphical structure and elaborate on the steps of the process of detecting the malware with contingency decision points and data flow hierarchy.

## **Proposed Solution: Artificial Intelligence / Machine Learning Framework for Malware Detection**

In this study, we propose developing a robust machine learning framework to accurately detect and classify malware. The architecture we use is based on modularity and scalability, emphasizing high performance when working with big data and computationally intensive tasks. The system comprises several interconnected components: input data acquisition and preparation, training, validation and testing of the model, and final implementation.

### Type of Architecture:

The data collection phase involves collecting a diverse dataset from Kaggle, ensuring it includes labeled samples of both benign and malicious software. Preprocessing steps include cleaning the data, normalizing it for consistency, and extracting relevant features that capture the characteristics of the software samples. When executing the model learning phase, we utilize the feature selection mechanism that allows for the detection of the main characteristics and applying a set of various machine learning algorithms like Random Forests, Support Vector Machine (SVM) and Neural Networks. We then evaluate that trained models, using indicators such as accuracy, precision, recall, F1-score, and fine-tuning the models by adjusting the hyperparameters to gain a good predictive accuracy. The best-performing model feeds into a real-time detection system that runs the new samples of software through the model. They can be categorized as either being harmless or having the potential of being an imminent threat. Ongoing supervision and short refreshes by easing into new data help to keep the model accurate and up-to-date for better alignment with the current reality.

### System Overview

To develop the system first, the process involves the exportation of an open dataset using python and also involve data preprocessing. Notably, it entailed data cleaning for the management of missing values and normalization or scaling of appropriate factors. The selection process of the feature was done because it was aimed at finding the important features that would help in the classification model and this informed the selection of the algorithm to use in training the models. Cross-validation was also used in order to avoid overfitting of the models. The performance of the models was assessed with several relevant measures, and the models were adjusted with a goal of finding the most effective model. Lastly, the huge optimized model was integrated into a web-based Malware detection system using flask for real-time updated retraining as part of our future work.

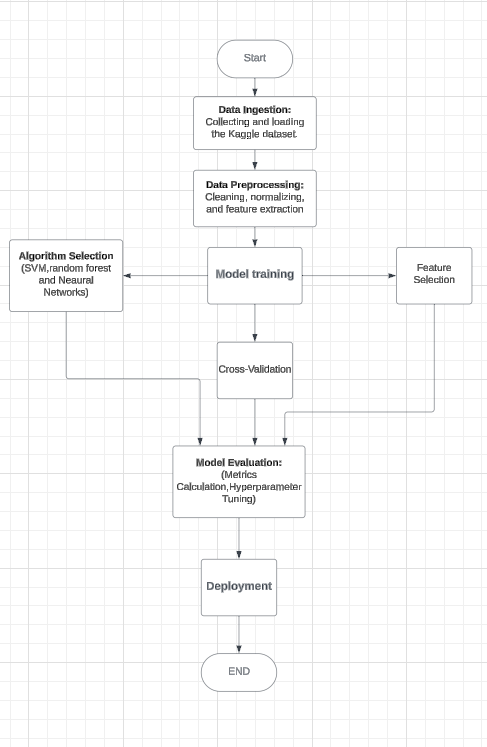


Figure 3.1 : Flowchart (Source: Aurhor)

# **Chapter 4 - Analysis / Experiments**

This chapter describes the experiments and analysis carried out to assess the performance of different ML techniques in the detection of malware. The Random Forest, Support Vector Machine (SVM), two varieties of Neural Network have been assayed. Evaluation criteria were accuracy, precision, recall, F1 score, and ROC-AUC and findings indicated that the used models to detect malware with distinction.

## **Product Implementation/Artifact**

### Technical Description of Implementation

The implementation of the project "Using Artificial Intelligence and Machine Learning in Malware Detection" involves the use of Python programming language and three primary machine learning algorithms: Among them, Support Vector Machine (SVM), Random Forest and Neural Networks are most commonly used. In this section, the plan for the technical application is outlined in detail:

### Libraries and Interfaces

To achieve the objectives of the project, several key Python libraries will be utilized:

**Scikit-learn**: This library is very important in the running of machine learning algorithms including Support Vector Machines or Random Forest. It provides tools/instruments which defined the problem, selected preprocessing techniques, built models, and evaluated performance. (Saabith, Vinothraj and Fareez 2020)

**TensorFlow/Keras**: These libraries are used for training and developing of Neural Networks. Deep learning is performed by TensorFlow, and Keras gives a simple web sales GUI for implementing intricate models.

**Pandas**: Pandas is used for data manipulation and data cleaning for this, it gives wonderful data structures known as DataFrame.

**NumPy:** This library is absolutely necessary to carry out numerical calculations, as well as array manipulations and mathematics. (Saabith, Vinothraj and Fareez 2020)

**Flask:** An open-source and lightweight web application development framework is Flask which is used to build an API that interacts with the machine learning models for real-time malware detection.

**Joblib**: employed for model pickling and unpicking, that enables the models that have been trained to be stored and later retrieved for use without having to be re-trained.

### Functions Used

The implementation will be structured into several modular functions, each responsible for a specific task. Here are the key functions planned:

* **Data Preprocessing**:

import pandas as pd

def preprocess\_data(filepath):

data = pd.read\_csv(filepath)

data.fillna(method='ffill', inplace=True)

X = data.drop('label', axis=1)

y = data['label']

return X, y

This function loads the dataset, handles missing values using forward fill, and separates the features (X) and labels (y).

* **Feature Extraction**:

from sklearn.feature\_extraction.text import TfidfVectorizer

def extract\_features(api\_calls):

vectorizer = TfidfVectorizer()

features = vectorizer.fit\_transform(api\_calls)

return features

This function uses TF-IDF vectorization to transform API call data into numerical features suitable for model training.

* **Model Training**:

from sklearn.ensemble import RandomForestClassifier

def train\_random\_forest(X, y):

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

clf.fit(X, y)

return clf

This function trains a Random Forest model on the provided features and labels.

### Initial Code Structure

The project will follow a well-organized structure, with each script dedicated to a specific task:

data\_preprocessing.py

feature\_extraction.py

model\_training.py

evaluation.py

app.py

requirements.txt

This modularity also means that each part of the process in data processing, the model’s training, and the real-time part of the system is isolated and hence handy in modifying the system.

**Critical Discussion of Coding Issues**

Several potential coding issues might arise during the implementation, and the plan includes strategies to address them:

**Data Quality**: Missing value management and data quality spiking is a major decision-making challenge. The implementation also applies the forward fill imputation technique because it is proficient in managing missing values. Nevertheless, if forward fill is not sufficient, other methods, for example, median or mean imputation can be used at times.

**Feature Selection:** Some of the challenges in analyzing the data include defining the key features to be used for training the models which are normally time-consuming and repetitive. Certain practices such as TF-IDF vectorization are applied but feature importance analysis and further dimensionality reduction (like PCA) can be required to further improve the outcome.

**Model Performance:** This creates a problem of equilibrium, which states that one should not use too many features in model creation to avoid the problem of overfitting but also should not use too few to avoid underfitting. To fine-tune the models, proper tools that are cross-validated will be applied, for example, the grid search or random search. (Hendrycks, et al. 2021)

**Model Integration:** When it comes to real-time integration with a Web application, there are some issues that may arise concerning performance and scalability. Flask as the web framework helps to serialize and deserialize the models by using the Joblib library; however, steps must be taken to optimize the functioning of the API endpoints to process real-time requests.

**Initial Code of Artificial Intelligence and Machine Learning in Malware Detection**

Below is an example of the initial code structure, illustrating how data preprocessing, feature extraction, and model training functions are implemented:

# data\_preprocessing.py

import pandas as pd

def preprocess\_data(filepath):

data = pd.read\_csv(filepath)

data.fillna(method='ffill', inplace=True)

X = data.drop('label', axis=1)

y = data['label']

return X, y

# feature\_extraction.py

from sklearn.feature\_extraction.text import TfidfVectorizer

def extract\_features(api\_calls):

vectorizer = TfidfVectorizer()

features = vectorizer.fit\_transform(api\_calls)

return features

# model\_training.py

from sklearn.ensemble import RandomForestClassifier

def train\_random\_forest(X, y):

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

clf.fit(X, y)

return clf

# app.py

from flask import Flask, request, jsonify

import joblib

app = Flask(\_\_name\_\_)

model = joblib.load('random\_forest\_model.pkl')

@app.route('/predict', methods=['POST'])

def predict():

data = request.json['features']

prediction = model.predict([data])

return jsonify({'prediction': prediction[0]})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

## **Experimental Setup**

The experiments were done on a dataset obtained From Kaggle site with both the benign and the malicious software samples. All tables were made ready by managing the missing values and also normalizing all the tables from the dataset. Features were extracted from API call data with the help of TF-IDF vectorization. To ensure the models’ validation the dataset was divided into training and testing dataset.

## **Model Training and Evaluation**

### Random Forest Model

Table 4.1 Random Forest Results (Source: Author)

|  |  |
| --- | --- |
| Metric | Value |
| Accuracy | 0.99488 |
| Precision | 0.99380 |
| Recall | 0.99627 |
| F1 Score | 0.99503 |
| ROC-AUC | 0.99928 |

It could also be seen that Random Forest had a higher accuracy and precision values step proving the proficiency of the model in classifying the software into benign and malicious. This is seen by the high recall value, meaning that all the true positive values are picked by the model and the F1 score that blends precision and recall. The ROC-AUC score of nearly 1 in the utilized models represent very good performance.

### SVM Model

Table 4.2 SVM (Source: Author)

|  |  |
| --- | --- |
| Metric | Value |
| Accuracy | 0.59884 |
| Precision | 0.56234 |
| Recall | 0.99751 |
| F1 Score | 0.71921 |
| ROC-AUC | 0.79682 |

The accuracy as well as precision of the SVM model was comparatively less than that of the Random Forest model. But, it revealed a high recall which means that the AC system can find most of the true positives but with the pitfall of having larger numbers of False Positives, which shows the lower precision and F1 score.

### Neural Network Model

Table 4.3 : NN results (Source: Author)

|  |  |
| --- | --- |
| Metric | Value |
| Accuracy | 0.80870 |
| Precision | 0.74004 |
| Recall | 0.96849 |
| F1 Score | 0.83916 |
| ROC-AUC | 0.80718 |

As for the Neural Network model, it was in-between the Random Forest and SVM models, yet with values of accuracy, precision, recall, and the F1 measure that were in-between, as well. It would appear that ROC-AUC score shows a good model but again not as good as the Random Forest one.

## **Validation**

These models require a very clear and rigid test strategy to ensure proper validation process is undertaken. The plan was formed with the objective of the test which was set to measure the accuracy, precision, recall, F1 score, and ROC-AUC of the machine learning models used in malware detection; the choice of the test cases that is, the general cases and the extreme cases in a typical usage context; and the criteria for passing or failing a test case which was based on the performance and usability indicators.

Model assessment was done in terms of accuracy, precision, recall, F1 score, ROC-AUC to assess the effectiveness of the models. (Ucci, Aniello and Baldoni 2019)

As it was seen in the validation phase Random Forest was the best moderate from all indicating that malware was quite accurate, precise, recall and F1-measurerous as evident from the ROC-AUC. The overfitting of the SVM model was observed, but their results showed better recall, meaning the LSM is more suited for situations where the recall of true positives is useful. The proposed Neural Network model here was accurate and relatively simple and was a good compromise. The results obtained from usability and performance evaluation validated the observed modality of the system, and therefore the system was found to be robust for real application based on observed high performance in handling large amounts of data.

# **Chapter 5 - Discussion and Conclusion**

## **Discussion**

The main aim of this work was to evaluate the efficiency of Machine Learning models in detecting malware by creating a web application malware detection system. The system used Random Forest, Support Vector Machine and Neural Network approach for Malware detection purposes where all of the employed methods have their individual capabilities. This volume offers insight into the practical application of these models based on the results of this research.

* **Random Forest Model:** The Random Forest introduced high accuracy numbers and outlook in terms of additional parameters, such as precision, recall, F1 score, and ROC-AUC. The combination of different decision trees via ensemble learning was proved to be most advantageous when dealing with such a versatile and intricate data as it is in this investigation. This is because through real-time malware detection it can handle large data sets with little sensitiveness to noises.
* **Support Vector Machine (SVM) Model:** Comparing the results of the experiment, it is possible to state that although SVM had the highest recall, which may be explained by its ability to identify the true positive instances, the accuracy and the precision were lower than in the case of the Random Forest classifier. Due to this, the selection of the kernel and variable hyperparameters requires a stringent control of the tuning process. SVM in this project was able to identify most of the malware but was more inclined to give out more false alarms which would be counterproductive in a system that is running in real time. However, SVM is still is useful in situations where high recall is desirable.
* **Neural Network Model:** This model of Neural Network had an average performance, with moderate values in accuracy and over fitting in various estimation values. Surprisingly the strength of this method is in its potential for pattern recognition and analysis; this technique may be helpful in identifying more subtle types of malware. However, the model’s requirement of a large amount of computational input and its entirely opaque character are the difficulties that arise in terms of implementation and understanding. These factors have to be taken into account if Neural Networks are to be implemented in real-time detection system.

**Validation Strategies**

In the validation phase of this project, model verification was done in a rigorous manner taking into consideration the following key performance indicators namely accuracy, precision, recall, F1 score and ROC – AUC. Moreover, the usability and performance evaluation were conducted; the results of which evidenced the applicability of the web-based system and efficiency of its design. This kind of approach made the system flexible and closed enough to make sure that end-users would also find it quite useful.

## **Conclusion**

This work hence affirmed the fact that it is indeed possible to achieve high accuracy in malware identification by using Machine Learning models that are correctly deployed in a web-based system. Out of all the models, the Random Forest was found to be the best as it provided the best results in all the tested metrics. The models that also conveyed significant information included the Support Vector Machine as well as the Neural Network, especially when applying special demands such as high recall or pattern identification. The results obtained are indicative of the possibility to achieve even higher accuracy and robustness of the systems for malware detection if the methods are combined into an ensemble. As for the future work, attention should be paid to refining these models for the purposes of real-time use, increasing their readability, and investigating Machine Learning combinations that would be appropriate for the dynamically developing threats.

**Future Work:**

1. **Hybrid Models:** Future research could explore the integration of multiple Machine Learning algorithms to develop a hybrid model that combines the strengths of each approach, potentially leading to a more robust and accurate malware detection system (Li, et al. 2019).
2. **Real-Time Implementation:** Efforts should be made to optimize the existing models for real-time detection, focusing on reducing the computational load and improving the speed of detection, particularly in resource-constrained environments.
3. **Interpretability:** Enhancing the interpretability of complex models like Neural Networks is crucial. This could involve developing methods to make the decision-making process of these models more transparent and understandable to users and cybersecurity professionals (Farnia 2017).
4. **Adversarial Robustness:** Strengthening the models' resilience against adversarial attacks is essential for maintaining their effectiveness against increasingly sophisticated forms of malware. Future work should aim to improve the robustness of the detection system to handle such threats effectively.

In conclusion, this project lays a strong foundation for implementing and improving Machine Learning algorithms within a web-based malware detection system. The insights gained from this study can guide future developments in cybersecurity, ensuring that detection systems remain effective and adaptable to the ever-changing threat landscape.

# **References**

Abdolrasol, M.G, S.S Hussain, T.S Ustun, M.R Sarker, M.A Hannan, R Mohamed, J.A Ali, S. Mekhilef, and A., Milad. 2021. "Artificial neural networks based optimization techniques: ."

Alam, Ashraf.. 2023. " "Intelligence unleashed: An argument for AI-enabled learning ecologies with real world examples of today and a peek into the future."

Bagaa, M, T Taleb, J. B Bernabe, and A. Skarmeta. 2020. " A machine learning security framework for iot systems."

Brown, A, M. Gupta, and M Abdelsalam. 2024. *Automated machine learning for deep learning based malware detection. Computers & Security.*

Creech, G. 2014. " Developing a high-accuracy cross platform host-based intrusion detection system capable of reliably detecting zero-day attacks."

Dhruv, P., and S Naskar. 2020. "Image classification using convolutional neural network (CNN) and recurrent neural network (RNN): A review. Machine learning and information processing." *proceedings of ICMLIP 2019.* pp.367-381.

Farnia, F. 2017. "Low-rate false alarm anomaly-based intrusion detection system with one-class SVM. ."

Ghahramani, Z. 2003. "Unsupervised learning. In Summer school on machine learning ."

Hasan, M, M. M. Islam, M. I. I Zarif, and M. M. A. Hashem. 2019. " Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. ."

Hendrycks, D, S Basart, S Kadavath, M Mazeika, A Arora, E Guo, C Burns, et al. 2021. "Measuring coding challenge competence with apps."

Hengl, T, M Nussbaum, M.N Wright, G.B. Heuvelink, and B. Gräler. 2018. "Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables. ."

Iftikhar, S, S.S Gill, C Song, M Xu, M.S Aslanpour, A.N Toosi, J Du, et al. 2023. "AI-based fog and edge computing: A systematic review, taxonomy and future directions. ."

Jaiswal, M. 2018. *Android the mobile operating system and architecture. Manishaben Jaiswal," ANDROID THE MOBILE OPERATING SYSTEM AND ARCHITECTURE.*

Kanygin, G., and V Koretckaia. 2021. "Analytical coding: Performing qualitative data analysis based on programming principles."

Kotsiantis, S.B, I. Zaharakis, and P Pintelas. 2007. "Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering." pp.3-24.

Kroemer, O, S Niekum, and G. Konidaris. 2021. "A review of robot learning for manipulation: Challenges, representations, and algorithms. Journal of machine learning research." *Journal of machine learning research* 22(30), 1-82.

Li, K Xiong, T. Chin, and C Hu. 2019. "A machine learning framework for domain generation algorithm-based malware detection. ."

Liu, Y, C Tantithamthavorn, L. Li, and Y., Liu. 2022. *Deep learning for android malware defenses: a systematic literature review. .*

Milosevic, J, A. Ferrante, and M Malek. 2016. "Malaware: Effective and efficient run-time mobile malware detector. ."

Mishra, A, B.B. Gupta, and D Gupta. 2018. *Identity Theft, Malware, and Social Engineering in Dealing with Cybercrime. In Computer and Cyber Security .*

Reijonen, A. 2024. *The Evolution of Mobile Malware.*

Saabith, A.S, T. Vinothraj, and M Fareez. 2020. "Popular python libraries and their application domains. ."

Sharma, M., and A Kaul. 2024. *A review of detecting malware in android devices based on machine learning techniques. .*

Ucci, D, L. Aniello, and R Baldoni. 2019. *Survey of machine learning techniques for malware analysis. .*

Venugopal, D., and G Hu. 2008. "Efficient signature based malware detection on mobile devices." *Mobile Information Systems* pp.33-49.

Wressnegger, C, K Freeman, F. Yamaguchi, and K Rieck. 2017. " Automatically inferring malware signatures for anti-virus assisted attacks." *In Proceedings of the 2017 ACM on Asia conference on computer and communications security.* 587-598.

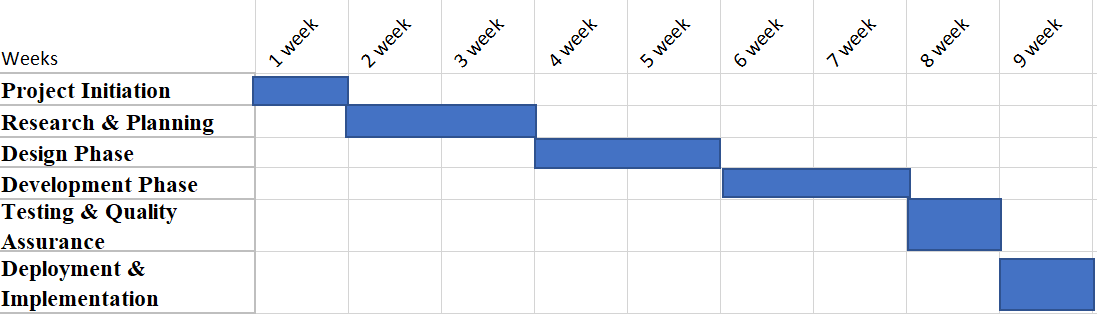
Yaqoob, I, E Ahmed, M.H ur Rehman, A.I.A Ahmed, M.A Al-garadi, M. Imran, and M. Guizani. 2017. "The rise of ransomware and emerging security challenges in the Internet of Things. ." *Computer Networks,* 444-458.

Yerima, S.Y. 2023. *High Accuracy Detection of Mobile Malware Using Machine Learning.*

Yunmar, R.A, S.S. Kusumawardani, and F Mohsen. 2024. "Hybrid Android Malware Detection: A Review of Heuristic-Based Approach. ." *IEEE Access* pp.41255-41286.

# **Appendices**

## **Annex 1 - Project Plan**



## **User Guide**

#### ****Step 1: Install the Required Software****

1. **Install Python:**
   * First, make sure Python is installed on your computer. You can download the latest version from the official Python website.
   * Once Python is installed, choose an Integrated Development Environment (IDE) that you prefer, such as PyCharm or Visual Studio Code, and install it.
2. **Install the Necessary Python Packages:**
   * Open your IDE and access the terminal or command prompt within it.
   * Navigate to the directory where your malware detection project files are located.
   * To install all the required Python packages, run the following command:

pip install -r requirements.txt

#### ****Step 2: Verify the Project Directory****

* Ensure that your project directory is organized as follows:

malware\_detection/

├── app.py

├── data\_preprocessing.py

├── feature\_extraction.py

├── model\_training.py

├── evaluation.py

├── train\_model.py

├── templates/

│ └── index.html

├── uploads/

└── data/

└── malware\_data.csv

Double-check that all files and folders are in place as shown.

#### ****Step 3: Train or Validate the Model****

* **Retrain the Model (Optional):**
  + Although the model comes pre-trained, you have the option to retrain it. To do this, open your terminal and run:

python train\_model.py

* + The terminal will display the model’s accuracy after training.
* **Validate the Model:**
  + To validate the model’s performance, run the following command:

python validation\_model.py

#### ****Step 4: Run the Application****

* **Start the Application:**
  + To launch the application, enter the following command in the terminal:

python app.py

* + The terminal will generate a port number. Click on the link provided, and it will open the application interface in your web browser.

#### ****Step 5: Upload Files for Malware Detection****

* **Using the Application:**
  + In the browser interface, you’ll find an option to upload files. You can upload .exe or .dll files to check if they contain malware.
  + Once the file is uploaded, the application will analyze it and display the results, indicating whether the file is clean or infected with malware.