National College of Ireland

Evaluating the Sensitivity of Machine Learning Algorithms to Training Data Size in OS X and Memory Malware Detection

MSc Research Project
MSc in Data Analytics

Devika Tamidala
Student ID: x23189428

School of Computing
National College of Ireland

Supervisor: Vikas Tomer

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Devika Tamidala
Student ID:	X23189428
Programme:	MSc in Data Analytics **Year:** 2024
Module:	Research Project
Supervisor:	Vikas Tomer
Submission Due Date:	12/12/2024
Project Title:	Evaluating the Sensitivity of Machine Learning Algorithms to Training Data Size in OS X and Memory Malware Detection
Word Count:	8733 **Page Count:** 23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Devika Tamidala

Date: 12/12/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Evaluating the Sensitivity of Machine Learning Algorithms to Training Data Size in OS X and Memory Malware Detection

Devika Tamidala
x23189428

Abstract

Malware detection is an important factor in cybersecurity as the number of complex attacks on OS X and memory-based systems continues to rise. Due to the increasing use of ML techniques, the effect of training data size on detection accuracy and time complexity is still an open issue. This work focuses on the problem of choosing reliable ML models for malware detection in scenarios with limited resources, especially training data. Three Machine Learning algorithms, namely, Logistic Regression (LR), K-Nearest Neighbors (KNN), and Gaussian Naive Bayes (GNB) have been considered in the present research, for performance assessment on two popular benchmark datasets of OS X and memory malware, namely the OS X Malware Dataset and CIC-MalMem-2022. Thus, sensitivity to the proportion of training data (10%, 20%, 50%, 80%, and 90%) is estimated, as well as accuracy, precision, recall, F1-score, and time to train each model. The findings show that memory malware detection has the lowest sensitivity to data size, while OS X malware detection is more sensitive, with LR giving the best results at larger datasets. The research also reveals that GNB is the most computationally efficient for both types of malwares. This research aims at identifying appropriate algorithms for real-time analysis and efficient use of resources in the detection of malware.

1. Introduction

1.1 Background

Malware detection is a fundamental part of cybersecurity that aims at detecting and preventing any malicious software that attacks computers. This, especially the memory related threats to the OS X systems call for improved and efficient measures in detecting these threats (Talukder et al.; 2020). The conventional or traditional methods of detection which include signature based detection are efficient in identifying known threats but are ineffective in identifying or detecting new or disguised threats. This shortcoming has resulted in the enhanced utilization of machine learning (ML) techniques because these models can process data to find trends that are associated with cyber threats (Gharghasheh et al.; 2022). To achieve their intended results, most of the ML algorithms require vast amounts of accurate and relevant data. But such type of data is not easily available in real life problems owing to factors such as privacy, cost, and time. Consequently, the role of the size of training data on the performance of the ML algorithms comes into focus. Understanding this relationship will help in the formulation of efficient detection techniques that can be used despite constraints in data to improve on the overall cybersecurity of organizations with scarce resources (Mijwil; 2020)

1.2 Aim

This research looks into the crucial cyber security problem of malware detection by comprehensively investigating the impact of training data size on the performance of various machine learning algorithms for OS X and memory malware detection. This research intends to determine the relationship between the size of training data and the sensitivity of the various algorithms or how much the chosen algorithm will be affected by the size of the training data and, in turn, the overall detection rate by systematically altering the proportions of the training data and testing different algorithms. This research will help know the algorithms that can tolerate such deviations in data, and understanding the best trade-offs between computational cost and performance, and inform better resource-limited cyber security solutions, thus improving real-world malware identification.

1.3 Objectives

- To collect and pre-process OS X and memory malware datasets to ensure data quality and consistency.
- To split each dataset into training, validation, and testing sets based on predefined proportions (10%, 20%, 50%, 80%, and 90%).
- To create multiple sub-datasets with different training data proportions: 10%, 20%, 50%, 80%, and 90% and ensure proportional representation of malware and benign samples in each sub-dataset.
- To implement a diverse set of machine learning algorithms, including Logistic Regression, K-Nearest Neighbors and Gaussian Naive Bayes for OS X and memory malware detection.
- To train each algorithm on each sub-dataset and evaluate its performance on a test set using metrics such as accuracy, precision, recall, and F1-score as well as time required for training, testing and validation.
- To compare the performance variability of different algorithms across varying training data sizes within and between datasets.
- To identify algorithms that demonstrate consistent robustness to fluctuations in training data size.

1.4 Research Question

1. How does the size of training data impact the effectiveness of detecting OS X and memory malware?
2. Which machine learning algorithm demonstrates the highest level of robustness to variations in training data size for OS X and memory malware detection?
3. What is the minimum proportion of training data required for each algorithm to achieve acceptable performance in OS X and memory malware detection?
4. How do the computational requirements (training time, memory usage) of each algorithm change as the training data size increases for both OS X and memory malware datasets?

1.5 Motivation

The increasing and more frequent attacks on computer systems, and the increasing complexity of malware which also include those that attack OS X and other complex memory systems, requires better and more efficient methods of detecting the malware. Standard methods of detecting worm-type malware using signature-based approaches are no longer sufficient given that malware is continually evolving and hiding (Prachi et al.; 2022). Hence

there is a need to get a break through by using the ML technique. However, one of the factors that greatly affect the performance of any given ML algorithm is the size and quality of the training sets. Most organizations face difficulties in obtaining big data because of issues on privacy, cost, and time taken to collect data (Yücel et al.; 2020). To meet these challenges, this work formally analysing the dependency of various ML algorithms on the size of the training data for malware detection. This research seeks to improve the malware detection process through examining the approaches that these algorithms use to ensure that they have high detection rates even when the data is scarce. The results can be useful for the improvement of cybersecurity products especially in the conditions when there is limited possibility to get access to large data sets (Botacin et al.; 2020).

1.6 Structure of the study

This paper is organized to provide a clear and coherent flow of research outcomes and conclusions. It begins with the **Introduction**, which contains the background of the study, purpose of the research, research questions and objectives, and motivation of the study. Then, the **Literature Survey** section expands on previous work in malware detection using machine learning approaches, and issues with training data differences. The **Research Methodology** section provides information on the data sets used, data cleaning, the research design and the ML techniques used for this research. Subsequently, the **Design and Implementation Specifications** section describes the practical aspects of model implementation. The **Evaluation** section contains results of the experiment, which discusses the efficiency of each algorithm and the results of their training depending on the data set size. The **Discussion and Conclusion** part of the work discusses these findings in detail, as well as their prospects and applications in the sphere of cybersecurity. Also, the constraints of the study and the recommendations for further research is also discussed in the last chapter.

2. Literature survey

Malware detection has greatly evolved especially using of machine learning (ML) for detecting new and complex types of malwares that are targeting operating systems and memory systems. This has been due to the understandings of the fact that conventional signature-based detection methods are lack of ability to deal with advanced threats.

Memory forensics has fast become an important field in digital investigation where various machine learning methods have been used in identifying and analyzing the features of malware. Sihwail et al. (2021) implemented a novel approach of using memory image to analyze and mine malware behaviors with classification accuracy of 98% using SVM classifiers. Continuing with this, Dener et al. (2022) adopted the Apache Spark's Pyspark platform to compare a number of algorithms on the CIC-MalMem-2022 dataset, in which LR reached a stunning 99.97% in malware detection. Ramesh et al. (2024) continued this work by proposing a Hybrid Random Forest and Naive Bayes (HRFNB) model that identified malware through memory analysis with a detection rate of 99.89%.

Various issues related to OS X malware detection have been identified as specific research problems. Gharghasheh and Hadayeghparast (2022) compared supervised machine learning techniques and proposed a novel idea of using library system calls as another feature set that increased the detection rate by 4%. Chen and Wulff (2022) likewise focused on the detection of macOS malware and found that Decision Tree algorithms achieved the best accuracy of 92.78%. Thaeler et al. (2023) further enriched this study by extracting metadata and suspicious strings, which raised the number of feature sets from 984 to 1693 and obtained Random Forest F1 scores of 0.8-0.9.

The advanced detection techniques have, therefore, incorporated deep learning and computer vision techniques. Lightweight malware detection in IoT environments was presented in a hybrid model that incorporated CNN with Bi-LSTM by Shafin et al. (2023). In the recent research work done by Xing et al. (2022), the DL model which is an autoencoder was employed with grayscale images of malware and it produced an outstanding performance of 96%. Venkatraman et al. (2019) have presented a hybrid DL process for detecting suspicious system behavior through image processing of approximately 75,000 malware samples.

Ensemble and hybrid techniques have been identified as especially useful in increasing the reliability of detection. Damaševičius et al. (2021) developed an ensemble classification method integrating the NNs and the ML models with the overall accuracy of 99.9% and low FP and FN rates. In the current study, Azeez et al. (2021) utilized a stacked ensemble learning approach which is divided into two stages, and RF got an accuracy of 99.24%. Singh and Bist (2020) applied Stacked Ensemble Classifiers (STENC) with the accuracy of 96.72% and class balancing the data by SMOTE methods.

There are also works on new feature selection and computational optimization techniques. Fang et al. (2024) presented a Deep Q-learning based Feature Selection Architecture (DQFSA) where more than 99% accuracy was achieved using 11 features out of a pool of 103. Euh et al. (2020) proved that it is possible to have low-dimensional features with high detection capability and minimal storage and training time, with XGBoost attaining 97% classification rate using the Window Entropy Map (WEM).

The current literature shows the lack of a clear understanding of how training data size impact algorithm performance across different platforms like OS X and memory systems. Although there are many articles that concern ML for malware detection, the analysis of the algorithms' dependence on the training data size is still rather scarce. The proposed research is intended to offer specific and actionable recommendations to improve the effectiveness and flexibility of malware detection by comparing the results of various ML algorithms based on training data size in resource-scarce settings.

This literature review aims to review current research on machine learning in malware detection and critically evaluate the current antecedent research thereby establishing the need for this research in enhancing cybersecurity techniques.

Table 1 Summary Table of Recent Studies

Authors	Year	Dataset Used	Methodologies Used	Metrics	Values	Limitations	Future Work
Sihwail et al.	2021	Custom Dataset (2502 malicious, 966 benign)	Extracted memory characteristics, applied binary vector feature transformation, trained and tested SVM classifier achieving 98% accuracy.	Accuracy, Sensitivity, False Positive Rate (FPR)	Accuracy: 98%, Sensitivity: 5%, FPR: 1.24%	Limited comparison to other malware techniques	Develop a larger memory-based dataset
Dener et al.	2022	CIC-MalMem-2022	Used PySpark on Google Colab with CIC-MalMem-2022 dataset, applied LR, RF, DT, GB, MLP, DFFNN, and	Accuracy, Precision	LR Accuracy: 99.97%, GB Accuracy:	Limited to binary classification, no analysis on	Expand dataset for detecting more malware

			LSTM, achieving 99.97% accuracy (LR).		99.94%, NB Precision: 98.41%	malware variants	types
Shafin et al.	2023	CIC-MalMem-2022	Proposed a hybrid CNN-BiLSTM model, tested on CIC-MalMem-2022 dataset for IoT malware detection, outperforming ML-based models.	Accuracy	Outperforms existing models on detecting obfuscated malware	Focuses primarily on IoT devices	Implement model in more diverse IoT devices
Bozkir et al.	2021	Custom Dataset (4294 samples)	Converted memory dumps into RGB images, used GIST+HOG descriptors, applied UMAP for manifold learning, achieved 96.39% accuracy (SMO).	Accuracy	Accuracy: 96.39% (SVM with GIST+HOG), UMAP improves accuracy by 20% (varies by model)	Limited to specific dataset and analysis on a standard desktop computer	Broaden dataset, test on more systems for real-world use
Shah et al.	2022	Memory dumps	Employed CLAHE and wavelet transforms for feature extraction from memory dumps, applied SVM, RF, DT, XGBoost, achieving 97.01% accuracy.	Accuracy, Precision, Recall, F1-Score	Accuracy: 97.01%, Precision: 97.36%, F1-Score: 96.36%	Limited variety of malware used, dependency on feature selection techniques	Investigate deep learning techniques for further malware detection
Gharghasheh & Hadayeghparast	2022	Custom Mac OS X dataset	Utilized library system calls as features, compared ML algorithms (DT, SVM, KNN, Ensemble, LR), achieving 94.7% accuracy with KNN.	ROC Curve	Accuracy: 94.7% (KNN)	Small dataset, requires constant updates for malware signatures	Expand dataset, explore new features to enhance detection
Chen & Wulff	2022	Custom Mac OS X dataset	Used macOS malware samples, evaluated DT, SVM, GNB, SGD, and LR; DT achieved 92.78% accuracy.	Accuracy	Accuracy: 92.78% (DT), SGD Accuracy: 91.77%	Limited variety of malware used, small dataset	Extend analysis to other algorithms and datasets
Sihwail et al.	2019	VirusTotal & Das Malwerk (1200	Integrated memory forensics with dynamic analysis, used API call attributes, applied SVM,	Accuracy, False Positive Rate	Accuracy: 98.5%, FPR: 1.7%	Dataset size limited to Windows	Integrate registry/network features,

		malware samples)	achieving 98.5% accuracy.	(FPR)		7 files	improve sandbox defense mechanisms
Carrier et al.	2022	MalMem Analysis-2022	Developed VolMemLyzer with stacked ensemble model using MalMemAnalysis-2022 dataset, achieving 99% accuracy with NB, DT, RF ensemble.	Accuracy	Accuracy: 99%	Controlled settings, focused on only three malware types	Expand to more malware types, test in real-world environments
Al-Qudah et al.	2023	MalMem Analysis-2022	Combined OCSVM with PCA for malware detection, applied to MALMEMANALYSIS-2022 dataset, achieving 99.4% accuracy.	Accuracy	Accuracy: 99.4%	Dependence on specific dataset	Broaden dataset and test model in dynamic contexts
Zhang et al.	2023	PE files (In-memory)	Used CNN with memory forensics on PE files, analyzing binary chunks, achieving 97.48% detection accuracy.	Accuracy	Accuracy: 97.48% (using 4096-byte fragments)	False positives if malicious code doesn't execute	Improve detection of dynamic harmful behavior
Xing et al.	2022	Android apps (10,000 benign, 13,000 malicious)	Proposed an AE-2 autoencoder for feature extraction, compared with CNN and traditional ML methods, achieving 96% accuracy.	Accuracy, F1-Score	AE-2 Accuracy: 96%, F1-Score: 96%	Data pre-processing inefficiencies	Improve pre-processing techniques, enhance robustness
Euh et al.	2020	Custom (20,000 benign, 20,000 malware)	Evaluated tree-based ensemble models using WEM and API features, applied AdaBoost, XGBoost, RF, achieving 97% accuracy with XGBoost.	Accuracy, AUC-PRC	Accuracy: 97%, AUC-PRC: 0.96+	High computational cost for certain features	Improve prediction algorithms, combine multiple feature types
Venkatraman et al.	2019	75,000 malware samples	Used grayscale image representation of malware binaries, trained deep learning models achieving high accuracy using Adam optimizer.	Accuracy, Training Loss	Fairly high accuracy, improved malware classification	Limited to large datasets	Explore other deep learning architectures, apply to real-time detection systems
Damaševičius et al.	2021	ClaMP Dataset	Proposed an ensemble classifier combining DNN and CNN with classical ML models, achieving 99.9%	Accuracy, Precision, F1-Score,	Accuracy: 99.9%, FPR: 0%, FNR: 0.2%	Dataset-specific models, no clear selection	Refine model architecture, add XAI for explainabilit

			accuracy with ExtraTrees.	AUC		criterion	y
Azeez et al.	2021	PE Dataset	Developed a two-stage classification using stacked CNN and ML classifiers (NB, DT, RF, etc.), ExtraTrees performed best.	Accuracy, FPR, FNR	Accuracy: 99.24%, FPR: 2.13%, FNR: 0.31%	Reliance on supervised learning, limited to known malware variants	Develop unsupervised ensemble learning techniques, add XAI
Fang et al.	2024	Custom Dataset	Used Deep Q-learning Feature Selection Algorithm (DQFSA), applied KNN, SVM, and RF, achieved 99% accuracy with a smaller feature set.	Accuracy, Feature Selection	Accuracy: 99%, Smaller feature set accuracy: 96%	Dataset-specific performance, limited generalization	Apply DQFSA to other feature selection tasks

3. Research Methodology

This section describes how the impact of training data size will be assessed regarding the ML algorithms for OS X and memory malware detection. It describes the choice of models and methods of data preparation, as well as measures for evaluating the accuracy, precision, recall and efficiency of computations. The objective of the research is to determine effective algorithms that deliver high results irrespective of the data size and context of detection.

3.1 Rationale for methods and Evaluation

The selection of the methods and metrics for this research is informed by the need to holistically examine the influence of training data on the performance of the ML algorithms in the classification of OS X and memory malware. The Logistic Regression model is chosen as the first model for prediction, due to its simplicity, interpretability and great performance in binary classification problems (Carrier et al.; 2022). K-Nearest Neighbors (KNN) is included in the models because it is non-parametric and can identify local patterns, thereby providing information on the performance of distance-based algorithms as the data size increases (Abualhaj et al.; 2024). GNB is chosen for its efficiency and probabilistic nature and for its ability to handle high dimensional feature spaces which is pertinent to malware datasets and to increase the variety of classifiers employed for the comparison (Khalil and Abu; 2023).

By designing the sub-datasets with different training data proportions (10%, 20%, 50%, 80% and 90%), the study is able to compare the behavior of each algorithm under different data constraints. This is similar to practical scenarios where the training data can be inadequate or excessive giving credibility to the study. This project uses the common metrics including accuracy, precision, recall, and F1-score, especially suitable in cases where the data is imbalanced as is often the case with malware detection. Besides, it is crucial to consider other computational factors such as training time and memory consumption to determine models that can perform well under limited resources.

To increase the validity of the research, performance is compared across both OS X and memory malware datasets, which are different from each other. This cross-dataset

comparison is useful in the prediction of algorithms that are less sensitive to variations in malware detection schemes.

3.2 Design Specification

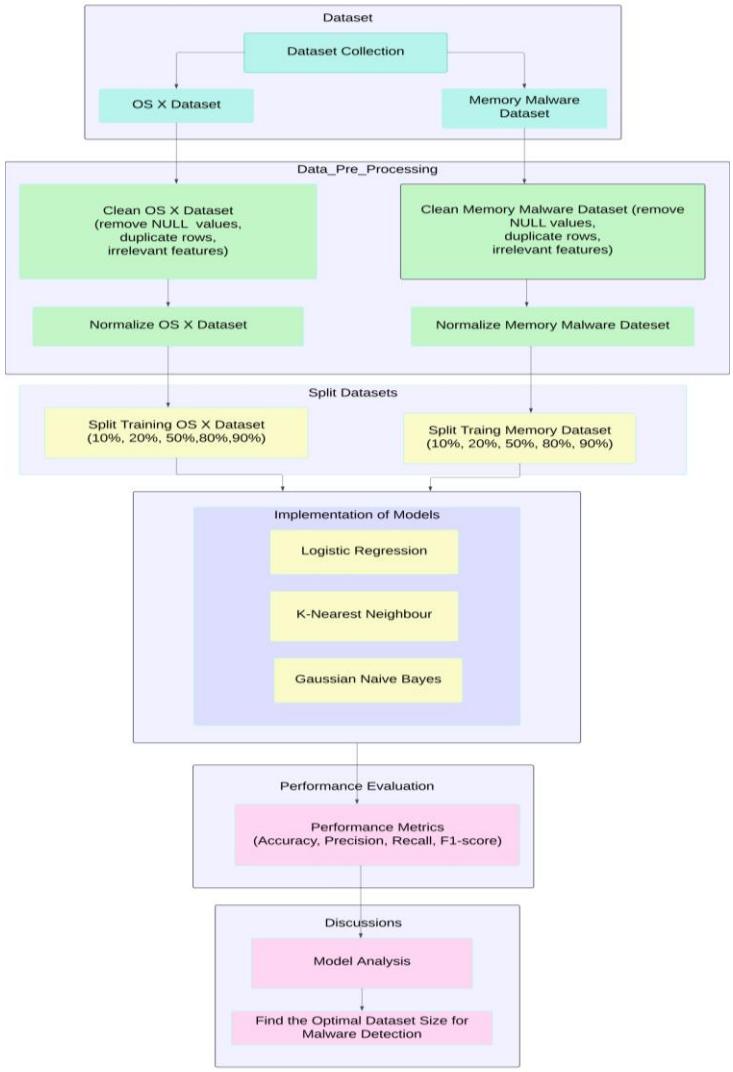


Figure 1: Architecture Diagram

In order to answer the research questions, this section outlines the procedure for an experiment focusing on the impact of the training set size on the performance of the ML algorithms for classification of OS X and memory malware samples. This approach begins with the dataset collection where the OS X malware dataset will be collected from Cyber Science Lab and the memory malware dataset from University of New Brunswick. Data pre-processing will be done on the data so as to enhance its quality and comparability by removing of duplicates as well as missing data, and selecting the required features. Subsequently, the data will be split for training, validation and testing at 10%, 20%, 50%, 80% and 90% respectively. More specifically, several sub-datasets will be produced, based on various training data proportions. The ratio of malicious and benign samples will be kept the same in all of the four datasets. Next, Logistic Regression, K-Nearest Neighbors, Gaussian Naive Bayes and other supervised algorithms will be applied to the problem of OS

X and memory malware detection. These models have to be created in Python using the Scikit-learn library. Then each Algorithm will be trained on each of the sub dataset and the performance of each algorithm on the test set will be evaluated by accuracy, precision, recall and F1 Score. The time spent during the training, testing, and validation of each algorithm will also be considered for detecting OS X and memory malware. Therefore, this analysis aims at identifying the algorithms that are expected to work nearly optimal with respect to the training data size.

3.3 Research Resources

3.3.1 Software Tools

- Python: A powerful programming language for data analysis, machine learning and data visualization.
- Scikit-learn: A widely used library for ML to design many algorithms.
- Pandas and NumPy: For data manipulation and numerical computations
- Matplotlib and Seaborn: For data visualization.
- Jupyter Notebook: For interactive data analysis and experimentation.

3.3.2 Hardware

- CPU: Intel Core i7 or AMD Ryzen 7 (or higher for demanding tasks)
- GPU: NVIDIA GeForce RTX 30 Series or AMD Radeon RX 6000 Series (or equivalent)
- RAM: 16GB or more
- Storage: SSD for fast data access
- Operating System: Linux-based (Ubuntu, Debian)

3.3.3 Datasets

- OS X Malware Datasets
- Memory malware datasets

3.4 Data Pre-Processing

The initial preparations for both set of malware data, the OS X and Memory datasets, began with the loading of the data, inspecting the class balance and erasing of unneeded columns. In particular, the OS X dataset contained uninformed values in LoadDYLIB, missing LoadDYLIB values, and duplicate rows. The Memory dataset had the Category column dropped and the labels were numerical encoded as (Benign=0, Malware=1). The correlation analysis demonstrated dataset-specific patterns: OS X features were generally negatively correlated with the target class, except for Segments, while the Memory dataset had both positive and negative feature correlation. The OS X dataset was also balanced in this project using SMOTE. These two preprocessed datasets were saved as two CSV files for further analysis.

3.5 Proposed Methodology

For this methodology, three ML models which include LR, KNN, and GNB are used in malware detection; each model used undergoes hyperparameter tuning. Logistic Regression, a simple and easy to interpret linear model for binary classification, is chosen for the analysis.

It is very useful for tasks such as malware detection, for which the aiml is to classify the object into the 2 categories: benign or malicious (Chaganti, Ravi and Pham; 2022). The initial settings are set to default with the liblinear solver as it is suitable for binary classification. Other hyperparameters such as C (regularization strength), penalty and max iteration are tuned using RandomizedSearchCV to improve the performance of the model by considering different combinations. The parameter C regulates overfitting, and the parameter penalty controls the type of regularization, which can be L1, L2 or elastic net.

This research chooses KNN algorithm for classification since it uses proximity as its basis for classification and it is useful in problems that have intricate relations between features such as detecting malware (Dolesi et al.; 2024). KNN does not make any assumption on the data distribution and has the capability to capture non-linear decision surfaces, which is very much useful in identifying different types of malwares. The KNN classifier is trained with default parameters and then the hyperparameters like number of neighbors, weights assigned and distance measure is set is set using RandomizedSearchCV. The basic yet effective concept of KNN that classifies data depend on the most common class of the nearest neighbors fits well to this task.

KNN classifier is chosen because it is suitable for classification problems where features are assumed to be independent. For instance, some features of benign and malicious samples can be well captured by the GNB model for malware detection. The GNB model is then started with the default settings for the priors and all features are considered to be conditionally independent. The main hyperparameter, var_smoothing, is introduced to avoid numerical instability when the variance is very small for some features in the data set. The classifier that are chosen has a probabilistic approach, which means that it classifies with less computational effort, thus being appropriate for this task (Kimmell; 2022).

All three models are trained with hyperparameters using 2-fold cross validation to avoid overfitting to a specific training set. RandomizedSearchCV is used to perform search for hyperparameters by sampling parameters from certain distribution in order to get the best results for each model. These models were selected based on their effectiveness to the binary classification problem of malware detection, whereby Logistic Regression is selected for its ability to provide an interpretation, KNN for its ability to handle non-linear boundaries and GNB for working with probabilistic classification. All the models are further trained and validated to improve the classification between the benign and malicious samples.

3.6 Model Training, Evaluation, and Comparison

The models such as LR, KNN and GNB—are fine-tuned using RandomizedSearchCV that further helps to further improve the effectiveness of the model by choosing the best set of hyperparameters from the training data. This step of optimization helps in enhancing the overall performance of the models in terms of prediction accuracy while being trained. In order to enhance the ability of distinguishing between malware and benign samples, the proposed methodology fine-tunes the following hyperparameters for each model to achieve a reliable model training process.

Performance is observed in terms of Training Time and Memory Usage utilizing the time and memory_profiler libraries respectively during the training as well as testing phase. The following metrics are used to determine the computational complexity needed for training of each model. Whenever the models are trained, they are evaluated on a new and unseen dataset at Testing Time and that time is noted down. At the end of testing, a classification report that touches on Accuracy, Precision, Recall and F1-Score is produced. Accuracy evaluates the overall accuracy of the model while precision and recall are crucial for

detecting true FPs and true FNs. F1-Score is beneficial in measurement when precision and recall are both important and when there is a significant class imbalance.
All of the models are validated on a different validation set for each model, to check that the model is able to perform better on new data and to minimize the risk of overfitting. This step is important to validate the performance of the model and its capacity to produce accurate results on new data. To assess the computational performance, the Training Time, Testing Time and Validation Time are recorded for each architecture. These time metrics give an insight of the time taken by each algorithm and the resources needed by the system in training, testing and validation. This makes it possible to determine the time taken to train the models and the accuracy of the models on new data that they haven't been trained on.
The last comparison of the models is based on Classification Performance where each model is comparing with other model using accuracy, precision, recall and F1-Score to determine how well they can differentiate between the benign and malware samples. Further, model performance is improved by Hyperparameter Optimization via RandomizedSearchCV, and Resource Efficiency is compared by analysing time taken to train the model and the memory used during training. These comparisons provide useful information about the ability of each model in the detection of memory malware.
Lastly, the hyperparameters that were found to give the best results for each model after training, testing and validation are presented. The classification results and curves of all models on the testing and validation sets indicate the models' ability to detect malware with the emphasis on the trade-offs among false positives and false negatives. This detailed evaluation allows identifying the best model for the malware detection task by comparison of the performance indicators and computational complexity.

4. Design And Implementation Specifications

Malware detection is necessary for cybersecurity enhancement. It identifies potential hazards to data and systems and implements protective measures. In two distinct scenarios: OS X malware and memory malware, this research examines the influence of training data size on the efficacy of ML algorithms for malware detection. To accomplish this, two distinct datasets were employed, each of which concentrated on a distinct form of malware. The subsequent sections will initially investigate the OS X malware dataset, followed by the memory malware dataset with experiments on both datasets.

4.1 Collection of OS X Malware Data

The OS X malware dataset, which is gathered from the Cyber Science Lab, comprises 613 samples and 16 features. It comprises 152 malware samples (labelled 1) and 461 benign samples (labelled -1), a class imbalance that is indicative of real-world situations. This dataset functions as a basis for comprehending the distinctive attributes of OS X malware and the obstacles associated with its detection.The dataset was purged of three columns (name, strsize, and DYLIBnames) due to their irrelevant nature or the presence of non-numeric, identifier-like data ('name'). For the classification of malware (DYLIBnames and strsize). Following the removal of these columns, the dataset comprises 13 columns that are predominantly numerical in nature: 12 integers and 1 float ('LoadDYLIB'). It is evident from the information that the dataset contains 613 entries, with only one missing value in 'LoadDYLIB' The data is consistent and prepared for further preprocessing and analysis, as there are no additional missing values, as indicated by the summary information.

4.1.1 Removing Null Values and Duplicate Rows

One missing value in the 'LoadDYLIB' column is initially detected. To eliminate this lacking value, the corresponding row is dropped. The duplicate entries were verified after the null values were addressed revealing 99 duplicate entries. These duplicates were eliminated, and the dataset's index is reset to preserve its order. A new column, 'index', is introduced to store the initial index values when the index is reset. As a consequence, the dataframe now contains 14 columns. Clean and consistent data for analysis is guaranteed as a result of the reduction of the dataset to 513 rows and 14 columns following these preprocessing steps.

4.1.2 Correlation Analysis of features in the OS X Malware Data

According to the heatmap, the target feature 'class' is negatively correlated with all features except 'Segments', suggesting an inverse relationship. The positive correlation observed in 'Segments' indicates its potential significance in the detection of malware.

4.1.3 Visualising the Key Features in OS X malware data

In the 'class' column of the OS X malware dataset, the pie chart illustrates the percentage of each class. It indicates that 24.8% of the data is classified as malware (class '1'), while the majority, 75.2%, is classified as non-malware (class '-1'). In modelling, the necessity of managing class distribution is underscored by this imbalance.

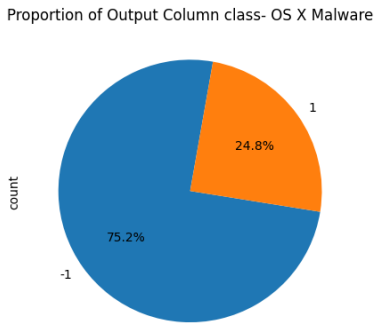


Figure 2: Pie Chart showing distribution of classes in OS X malware data

The OS X malware dataset's frequency of each class category across various segments is illustrated in the stacked bar diagram. The preponderance of occurrences are classified as Segment '3' in both the malware (class '1') and non-malware (class '-1') categories.

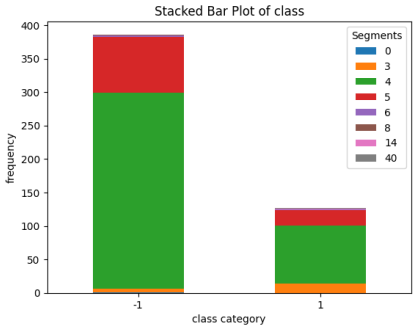


Figure 3: Stacked Bar Plot of frequency of each class across various segments

This suggests that Segment '3' is present in both classes, which could imply its importance in the differentiation between malware and non-malware. The plot assists in the identification of trends in the distribution of segments within each class, thereby offering insight into the potential importance of the feature.

4.1.4 OS X Malware Data Balancing

The dataset is balanced by oversampling the minority class (malware, class '1') using SMOTE, which resulted in 386 instances of each class ('1' for malware and '-1' for non-malware). This process guarantees an equitable distribution of both classes, thereby enhancing model performance by addressing class imbalance.

4.2 Collection of Memory Malware Data

The Memory Malware dataset (Obfuscated-MalMem2022.csv) is collected from the University of New Brunswick (UNB). It contains 57 columns and 58,596 rows, which correspond to the system's detailed activity features. The target column, 'Class', is composed of two balanced classes: Benign (29,298 instances) and Malware (29,298 instances). Since the Class column serves as the target variable, the Category column is removed to make the data simpler. The info() method confirmed that there are no missing values in the 56 columns of the dataset. Data types include 15 float, 40 integer, and 1 object (target) features.

4.2.1 Preprocessing: Null Check, Handling Duplicates, and Encoding

No null values were identified and the dataset is considered full. The dataset is then stripped down to 58,027 rows and 57 columns after 569 duplicate rows were found and eliminated. The index is reset to ensure uniformity. To further ensure compatibility with ML models, the categorical values "Benign" and "Malware" in the 'Class' column were transformed to numerical labels: 0 for "Benign" and 1 for "Malware".

4.2.2 Correlation Analysis of features in the Memory Malware Data

Annotations and a cool colour palette are used to build a heatmap that visualises the dataset's correlation matrix. All 55 features in the accompanying heatmap show a positive or negative correlation with the output feature "Class," making it a tremendously large heatmap. This is helpful for comprehending the effect of feature relationships on malware categorisation.

4.2.3 Visualising the Key Features

This pie chart from the Memory Malware dataset shows how the 'Class' column is distributed. Instances of benign content make for 50.4% of the dataset, whereas instances of malware content account for 49.6%. This equilibrium allows for fair assessment and training of ML models.

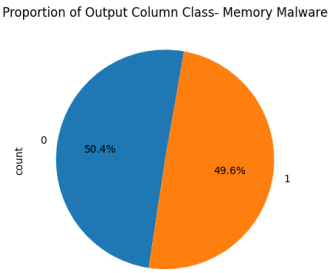


Figure 4: Pie Chart showing distribution of classes in Memory malware data

The stacked bar plot illustrates the frequency of malware and non-malware instances at varying callbacks.ngeneric values. There is a commonality in this feature, as both classes predominantly have a callbacks.ngeneric value of '3'.

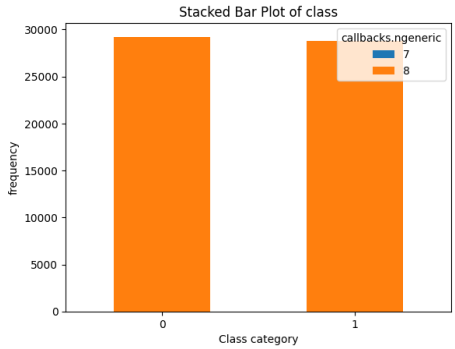


Figure 5: Stacked Bar Plot of frequency of each class across various callbackss.ngeneric

The plot indicates that this shared attribute may function as a critical indicator for classification, despite the presence of variations in other categories. The distribution is more easily comprehended, and patterns that are pertinent to malware detection are more readily identified as a result of the proportionate representation across classes.

5. Evaluation

After preprocessing, the OS X malware dataset comprises 772 samples and 13 features, with the binary classes being converted from {1, -1} to {1, 0}. The memory malware dataset comprises 55 features and 58,027 samples, with binary classes predefined as {0, 1}. Both datasets were subjected to feature selection and cleansing to guarantee their fitness for the purpose of training machine learning models.

5.1 Data Splitting and Sub-Dataset Creation

Based on predetermined training data proportions of 10%, 20%, 50%, 80%, and 90%, both datasets were divided into training, validation, and testing sets. After allotting training data, the remaining samples were equitably allocated to validation and testing. Each subset maintained a proportional representation of benign and malware samples.

Table 2: Splitted data samples using predefined training proportions

Training Data Proportion	OS X Malware (samples)			Memory Malware(samples)		
	Train	Test	Val.	Train	Test	Val.
10%	77	348	347	5802	26113	26112
20%	154	309	309	11605	23211	23211
50%	386	193	193	29013	14507	14507
80%	617	78	77	46421	5803	5803
90%	694	39	39	52224	2902	2901

5.2 Model Implementation and Hyperparameter Tuning

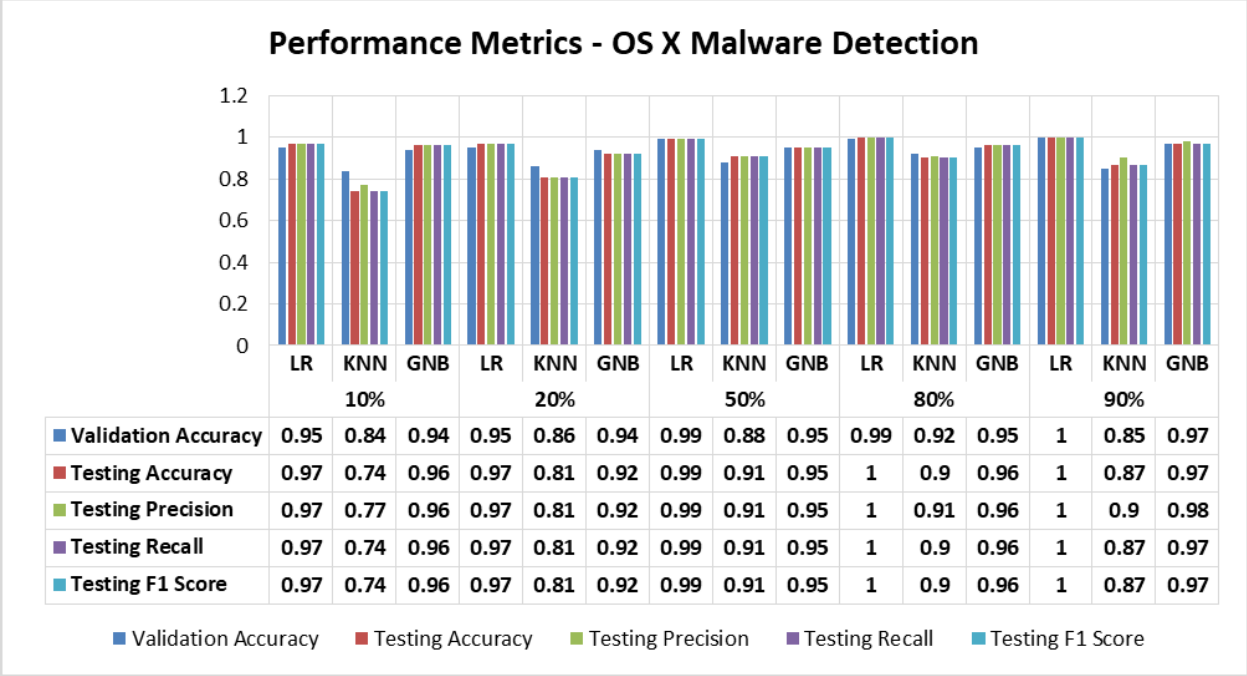
For both datasets, three machine learning algorithms—LR, KNN, and GNB—are employed. Hyperparameter optimization is accomplished through the utilisation of RandomizedSearchCV with 2-fold cross-validation. Each model is trained on all sub-datasets and assessed using metrics such as precision, recall, F1-score, and computational time for training, testing, and validation.

5.3 Performance Across Training Data Proportions in OS X Malware Detection

The results reveal that the amount of training data greatly affects LR. As the size of the training set increases, there is a noticeable enhancement in accuracy and performance. While LR gets a testing accuracy of 0.97 with 10% training data, it reaches perfection with 1.00 accuracy, precision, recall, and F1 score when the training data is increased to 90%. Accordingly, it appears that LR's ability to generalise and accurately detect malware is greatly enhanced with more training data samples. With its impressive performance, LR stands out as an algorithm that could be used to detect malware on OS X.

The sensitivity of KNN to data size is, in contrast, more moderate. The performance of KNN is significantly worse for lower training proportions, such as 10% and 20%, with testing accuracy hanging around 0.74 and 0.81, respectively. The testing accuracy of KNN is 0.91 with 50% training data and 0.87 with 90% training data, demonstrating a steady improvement in performance as the training data grows. Nevertheless, KNN is never able to achieve the same level of precision as LR. Regardless, KNN is still a viable bet for OS X malware detection due to its increased accuracy with bigger training datasets.

Table 3: Performance metrics of ML Algorithms in OS X malware detection

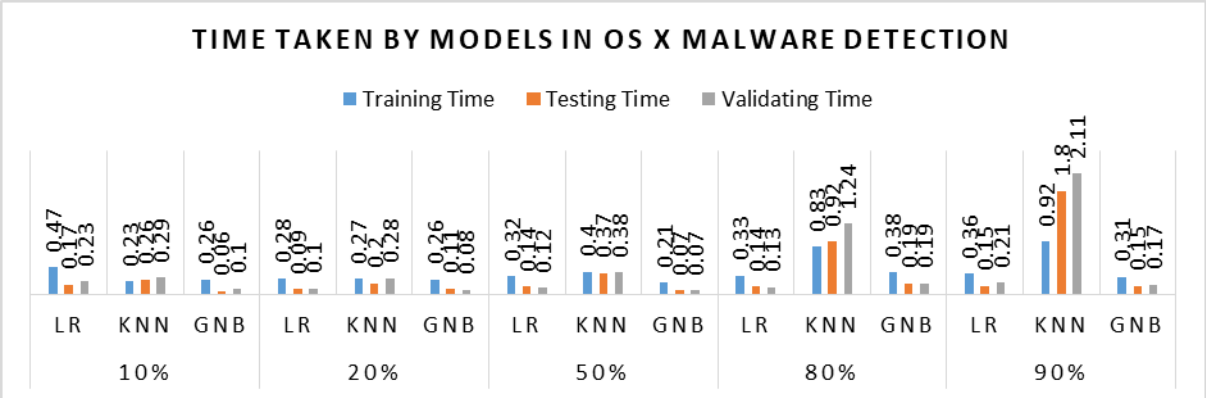


Regardless of the quantity of training data, GNB's performance remains consistent. Even with smaller datasets (10% training data), GNB consistently exhibits accuracy (around 0.95 to 0.96), demonstrating its stability and capacity to perform effectively with fewer data points. Since labelled training data may be scarce in resource-constrained settings, GNB is an excellent choice. Despite the fact that GNB's accuracy does not surpass 0.97, its reliability and speed render it a formidable contender for the detection of OS X malware, particularly when computational efficiency is a critical factor.

5.3.1 Computational Efficiency during OS X Malware Detection

Based on the results, LR is the quickest of the three. Its training time is 0.47 seconds with 10% training data and only slightly increases at 90%. Similarly, its testing time is relatively low and remains consistent throughout data sizes.

Figure 10: Computational Efficiency of ML Algorithms in OS X malware detection



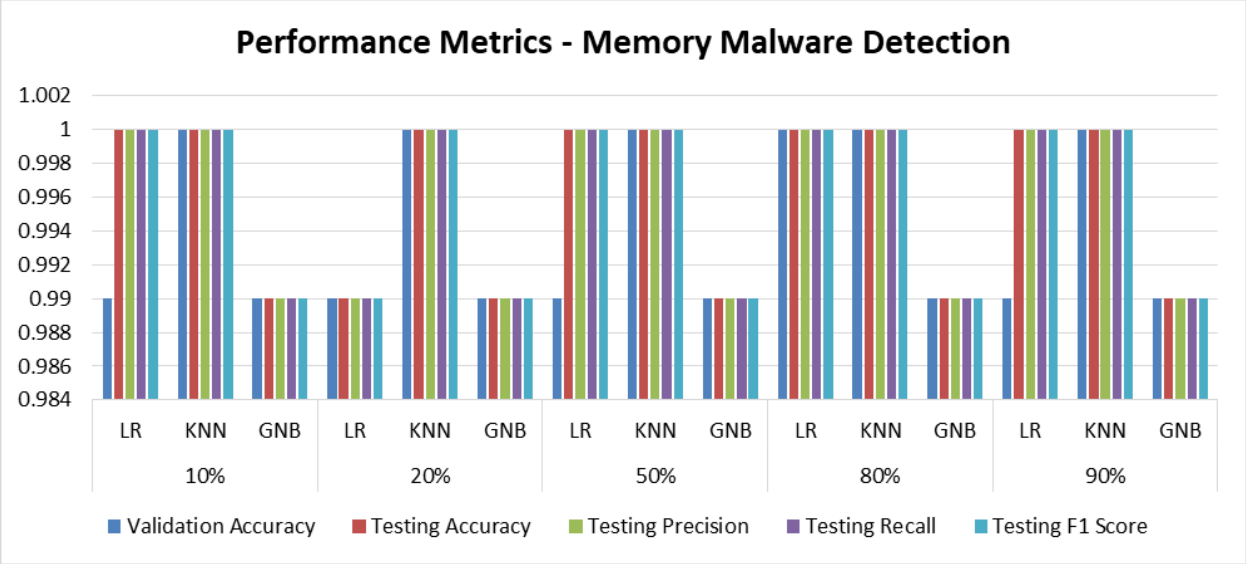
Alternatively, KNN's training, testing, and validation periods are significantly longer, particularly when dealing with larger training samples. For example, KNN's testing time

jumps to 1.80 seconds when 90% of the data is taken for training, demonstrating its computational inefficiency when dealing with large datasets. With training and testing times that are far lower than KNN and only somewhat higher than LR, GNB maintains its position as the most efficient algorithm. Real-time malware detection in OS X systems is a top priority, and GNB is a perfect fit because of its quick training times, particularly with all proportions of training data.

5.4 Performance Across Training Data Proportions in Memory Malware Detection

According to the findings, LR reliably delivers excellent performance with respect to all data proportions. With only 10% training data, LR achieves a testing accuracy of 1.00 while keeping precision, recall, and F1-score at ideal levels as the data amount increases. Because of this, it seems that LR can detect memory malware very well, even with little training data.

Table 4: Performance metrics of ML Algorithms in memory malware detection

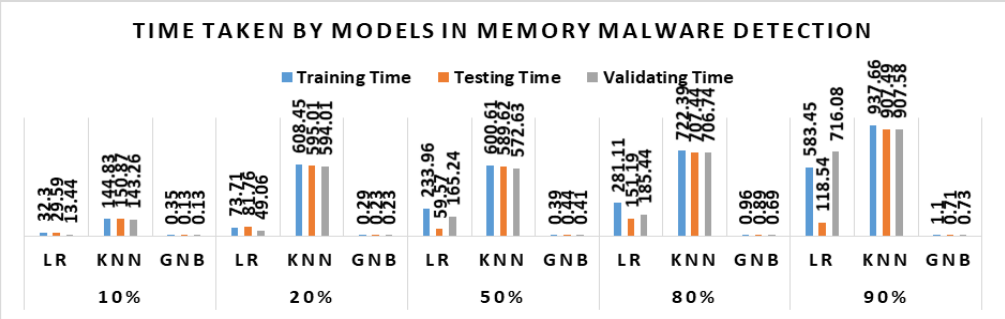


With validation accuracy starting at 0.99 at 10% and reaching 1.00 with increasing proportions, the training set size directly correlates to the amount of data provided, demonstrating that LR is able to generalise effectively. When considering memory malware identification, LR is a trustworthy technique that consistently performs well on both small and large data sizes. Additionally, KNN exhibits flawless detection across all proportions of the training data, with testing accuracy, precision, recall, and F1-score all coming in at 1.00 across the board. Based on these results, KNN seems to be a great option for identifying memory malware because of how well it captures patterns in the data. However, GNB falls short of the flawless performance achieved by LR and KNN, while it still achieves respectable results. A testing accuracy of 0.99 is achieved by GNB at 10% training data, and this accuracy is consistent across all proportions. Despite being marginally less accurate than LR and KNN, GNB is still a dependable classifier with an accuracy, recall, and F1 of 0.99. Nonetheless, GNB shows promise as a viable alternative for memory malware detection due to its consistent performance across different data volumes.

5.4.1 Computational Efficiency during Memory Malware Detection

Training times for LR are reasonable and scale up as the training data amount increases; for example, at 10% training data, it takes 32.30 seconds, and at 90%, it rises to 583.45 seconds. In comparison to KNN, which incurs substantially greater computational costs, LR continues to be comparatively efficient, even with this boost.

Figure 11: Computational Efficiency of ML Algorithms in memory malware detection



The computational expense of KNN becomes apparent at 90% training data; the time required for training increases to 937.66 seconds from 144.83 seconds at 10%. At 10% training data, GNB's training times are as low as 0.35 seconds, and at 90%, they only marginally increase to 1.10 seconds, making it the most computationally efficient. Since speed and minimal resource consumption are paramount in cybersecurity applications, GNB is a perfect fit.

6. Discussion and Conclusion

6.1 Critical Analysis of Sensitivity of ML Algorithms to Training Data Sizes in Malware Detection

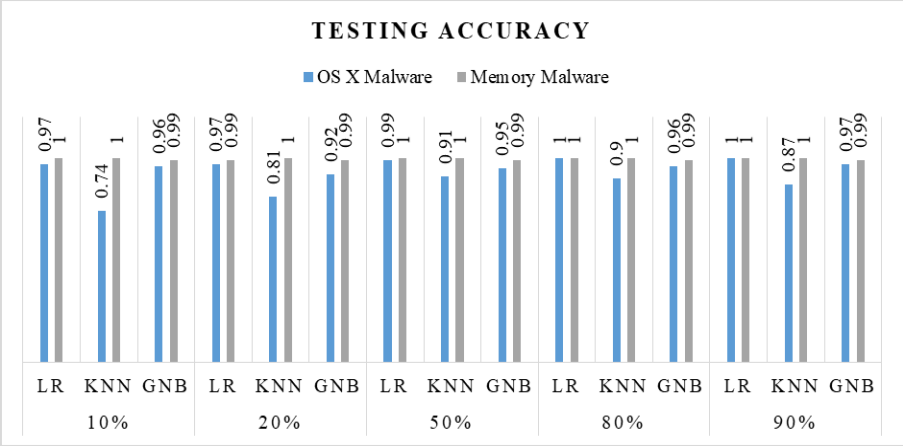
Investigating the effect of training data size on ML algorithm performance for OS X and memory malware detection reveals that training data proportions impact both types of malware detection in different ways. Regardless of the quantity of training data, all three algorithms (LR, KNN, and GNB) in memory malware detection consistently attain testing accuracies between 99% and 100%. Given that all models kept their high detection rates over different amounts of training data, it shows that the memory malware detection task is substantially less sensitive to changes in training data size. The chosen models are extremely capable of detecting memory-based malware, as evidenced by the robust accuracy, even with smaller training proportions.

On the other hand, there is considerable diversity in algorithm performance for OS X malware detection when varying training data sizes. When using 80% or more of the training data, Logistic Regression (LR) achieves 100% accuracy, demonstrating consistent good performance. The accuracy decreases to approximately 97% at reduced training data proportions (10% and 20%), but performance improves as the data size increases.

K-Nearest Neighbours (KNN) demonstrates considerable accuracy swings when it comes to detecting OS X malware. Its performance noticeably drops when compared to other models, especially at smaller training data proportions (10% with 74% accuracy). Gaussian Naive

Bayes (GNB) demonstrates a minor decline in accuracy when the training data is smaller, such as 20% with 92% accuracy, despite its relatively good performance.

Figure 12: Testing Accuracy of ML models in OS X and memory malware detection



The two scenarios illustrate that LR is the most efficient approach in terms of computing speed, although training time grows with bigger datasets, particularly for memory malware detection (where it goes from 32.30 seconds at 10% training data to 583.45 seconds at 90%). Training times for KNN range from 144.83 seconds at 10% training data to 937.66 seconds at 90%, indicating that it is computationally expensive. This becomes a substantial load for larger datasets. For real-time malware detection in both OS X and memory malware scenarios, GNB is the best option because it is computationally efficient and training and testing timeframes are minimally affected.

In broad terms, the computational complexity of memory malware detection is higher than that of OS X malware detection, despite the fact that the accuracy of memory malware detection is consistently robust across all algorithms and training data sizes. The computational demands of models, particularly LR and KNN, are underscored by the substantial increase in time required as the size of the training data increases. The efficacy of OS X malware detection is more sensitive to the proportions of training data, with LR exhibiting the most favourable balance between accuracy and efficiency at higher data volumes. GNB continues to be the most computationally efficient option, providing a balance between speed and accuracy that renders it the optimal choice for real-time cybersecurity solutions and resource-limited environments.

Optimum Model Selection: Due to its consistent accuracy and computational efficiency, GNB is the most effective model for detecting memory malware. The optimal choice for OS X malware detection is LR, as it achieves 100% accuracy with larger training data sizes while maintaining reasonable computational costs.

6.2Addressing Research Questions

1. How does the size of training data impact the effectiveness of detecting OS X and memory malware?

The accuracy of OS X malware detection was substantially enhanced by the size of the training data, particularly for LR. On the other hand, memory malware detection

demonstrated minimal sensitivity to the size of the training data, achieving consistently high accuracy (99–100%) across all models. GNB in particular demonstrated exceptional computational efficiency in real-time, resource-limited scenarios.

2. Which machine learning algorithm demonstrates the highest level of robustness to variations in training data size for OS X and memory malware detection?

OS X malware detection is most robust when using Logistic Regression (LR), which maintains a high level of accuracy (97%–100%) across a variety of training sizes. For the detection of memory malware, all algorithms (LR, KNN, and GNB) exhibit consistent performance (99%-100%). However, GNB is the preferable option due to its computational efficiency, which renders it ideal for resource-limited settings.

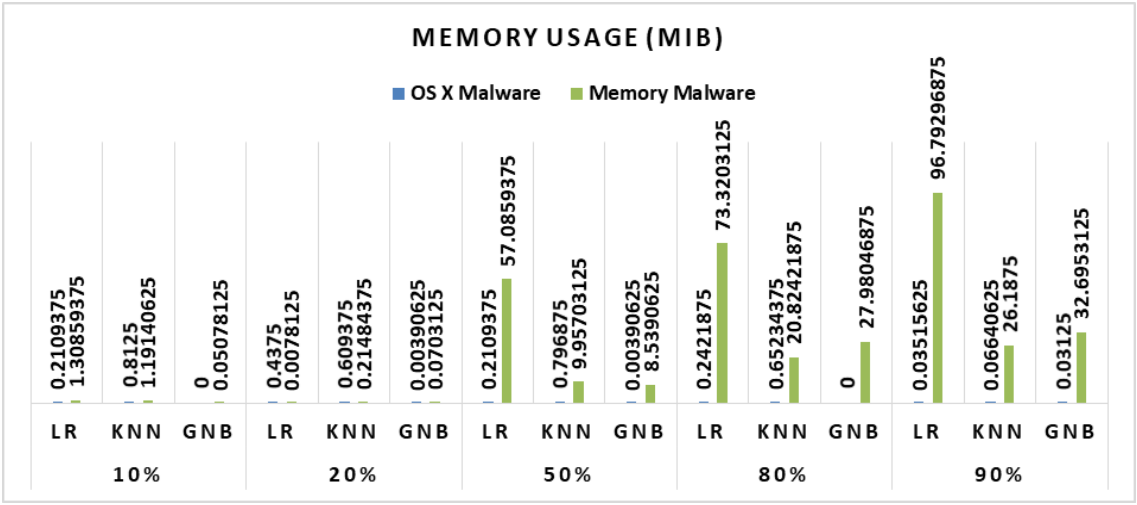
3. What is the minimum proportion of training data required for each algorithm to achieve acceptable performance in OS X and memory malware detection?

Logistic Regression and Gaussian Naive Bayes were able to obtain a 97% and 96% accuracy rate, respectively, for OS X malware detection with only 10% training data. Nevertheless, KNN necessitated a minimum of 50% of the training data to achieve a 91% stability in its performance. To detect memory malware, all algorithms achieved ≥99% accuracy with 10% training data.

4. How do the computational requirements (training time, memory usage) of each algorithm change as the training data size increases for both OS X and memory malware datasets?

All algorithms on the OS X and memory malware datasets experience an increase in computing needs for training time and memory usage as the training data size increases.

Table 5: Training time and Memory usage



As the amount of training data increases, the computational demands for detecting OS X malware and memory malware also rise. The table above shows that with increase in data size the memory usage also increases.

6.3 Conclusion

Malware detection plays an important role in cybersecurity, as this is the process of identifying and eliminating threats that are present in the shape of malicious software. As the number of cyber threats, especially those that exploit memory in OS X systems, has been on the rise, there is a need to develop better and faster methods of detecting malware. This work was set to explore how the size of the training data will influence the performance of various machine learning algorithms in the detection of OS X and memory malware. Here the attempt was made to assess how these algorithms behave when the training dataset proportions are altered and which of the models is most effective in terms of resource-limited real-word environments.

This project investigates the performance of three classification algorithms: LR, KNN and GNB in terms of their performance on varying training dataset sizes and computational time. The analysis revealed that, for memory malware detection, model performance was not as impacted by changes in training data size, and all models maintained very high accuracy rates of 99-100%. On the other hand, the performance of OS X malware detection was influenced by data size; LR provided the optimal performance at a larger set, while KNN offered unstable results.

The research outcomes show that GNB has better computational capability and therefore is the most suitable for detection of memory malware, especially in environments with limited resources. Using LR, the best results were obtained for OS X malware detection with the highest accuracy and sufficient training data. Based on the results, GNB and LR were the most stable models, with GNB providing the best trade-off between accuracy and time complexity for real-time use.

In conclusion, this research revealed the importance of the training data set size and offered guidelines for choosing the most suitable algorithms for real-world malware detection.

6.4 Future Enhancements

To facilitate a more thorough assessment of algorithm performance across various malware categories, future research could concentrate on expanding the dataset to encompass a broader range of malware types beyond OS X and memory malware. Furthermore, the investigation of more sophisticated methods for machine learning, such as neural network models (e.g., CNN or RNN), could offer evidence regarding whether these types of models outperform conventional algorithms like LR, KNN, and GNB in malware detection tasks. Additional refinement of data balancing techniques, such as the implementation of alternative oversampling or undersampling methods beyond SMOTE, may also enhance model performance, particularly in the context of highly imbalanced datasets.

Additionally, the accuracy and efficacy of the model could be improved by integrating feature engineering techniques, such as the selection of more informative features and the utilisation of domain-specific knowledge. In addition, research could evaluate the efficacy of real-time malware detection models in live environments to evaluate their scalability and robustness by integrating model deployment.

References

Talukder, S. and Talukder, Z. (2020). A survey on malware detection and analysis tools, *International Journal of Network Security & Its Applications (IJNSA)*, 12.

Gharghasheh, S.E. and Hadayeghparast, S. (2022). Mac OS X malware detection with supervised machine learning algorithms, *Handbook of Big Data Analytics and Forensics*, pp. 193-208.

Mijwil, M.M. (2020). Malware Detection in Android OS Using Machine Learning Techniques, *International Journal of Data Science and Applications*, 3(2), pp. 5-9.

Prachi and Kumar, S. (2022). An effective ransomware detection approach in a cloud environment using volatile memory features, *Journal of Computer Virology and Hacking Techniques*, 18(4), pp. 407-424.

Yücel, Ç. and Koltuksuz, A. (2020). Imaging and evaluating the memory access for malware, *Forensic Science International: Digital Investigation*, 32, p. 200903.

Botacin, M., Grégio, A. and Alves, M.A.Z. (2020, September). Near-memory & in-memory detection of fileless malware, In *Proceedings of the International Symposium on Memory Systems*, pp. 23-38.

Sihwail, R., Omar, K. and Arifin, K.A.Z. (2021). An Effective Memory Analysis for Malware Detection and Classification, *Computers, Materials & Continua*, 67(2).

Dener, M., Ok, G. and Orman, A. (2022). Malware detection using memory analysis data in big data environment, *Applied Sciences*, 12(17), p. 8604.

Ramesh, S.P., Anand, S.R. and Karthikeyan, V.G. (2024, November). Machine Learning Approach for Malware Detection Using Malware Memory Analysis Data, In *International Conference on Applications and Techniques in Information Security*, pp. 135-145. Singapore: Springer Nature Singapore.

Chen, A.C. and Wulff, K. (2022). Machine learning for OSX malware detection, *Handbook of Big Data Analytics and Forensics*, pp. 209-222.

Thaeler, A., Yigit, Y., Maglaras, L., Buchanan, W.J., Moradpoor, N. and Russell, G. (2023, November). Enhancing Mac OS Malware Detection through Machine Learning and Mach-O File Analysis, In *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 170-175. IEEE.

Shafin, S.S., Karmakar, G. and Mareels, I. (2023). Obfuscated memory malware detection in resource-constrained IoT devices for smart city applications, *Sensors*, 23(11), p. 5348.

Venkatraman, S., Alazab, M. and Vinayakumar, R. (2019). A hybrid deep learning image-based analysis for effective malware detection, *Journal of Information Security and Applications*, 47, pp. 377-389.

Damaševičius, R., Venčkauskas, A., Toldinas, J. and Grigaliūnas, Š. (2021). Ensemble-based classification using neural networks and machine learning models for windows pe malware detection, *Electronics*, 10(4), p. 485.

Azeez, N.A., Odufuwa, O.E., Misra, S., Oluranti, J. and Damaševičius, R. (2021, February). Windows PE malware detection using ensemble learning, In *Informatics*, 8(1), p. 10. MDPI.
Singh, A. and Bist, A.S. (2020). OSX malware detection: Challenges and solutions, *Journal of Information and Optimization Sciences*, 41(2), pp. 379-385.

Fang, Z., Wang, J., Geng, J. and Kan, X. (2019). Feature selection for malware detection based on reinforcement learning, *IEEE Access*, 7, pp. 176177-176187.

Euh, S., Lee, H., Kim, D. and Hwang, D. (2020). Comparative analysis of low-dimensional features and tree-based ensembles for malware detection systems, *IEEE Access*, 8, pp. 76796-76808.

Botacin, M., Grégio, A. and Alves, M.A.Z. (2020, September). Near-memory & in-memory detection of fileless malware, In *Proceedings of the International Symposium on Memory Systems*, pp. 23-38.

Bumanglag, K. (2022). An Application of Machine Learning to Analysis of Packed Mac Malware.

Hilabi, R. and Abu-Khadrah, A. (2024). Windows operating system malware detection using machine learning, *Bulletin of Electrical Engineering and Informatics*, 13(5), pp. 3401-3410.

Carrier, T., Victor, P., Tekeoglu, A. and Lashkari, A.H. (2022, February). Detecting Obfuscated Malware using Memory Feature Engineering, In *Icissp*, pp. 177-188.

Abualhaj, M., Abu-Shareha, A., Shambour, Q., Alsaaidah, A., Al-Khatib, S. and Anbar, M. (2024). Customized K-nearest neighbors' algorithm for malware detection, *International Journal of Data and Network Science*, 8(1), pp. 431-438.

Khalil, M. and Abu Al-Haija, Q. (2023). Memory Malware Identification via Machine Learning, In *Mobile Computing and Sustainable Informatics: Proceedings of ICMCSI 2023*, pp. 301-315. Singapore: Springer Nature Singapore.

Chaganti, R., Ravi, V. and Pham, T.D. (2022). Deep learning based cross architecture internet of things malware detection and classification, *Computers & Security*, 120, p. 102779.

Dolesi, K., Steinbach, E., Velasquez, A., Whitaker, L., Baranov, M. and Atherton, L. (2024). A machine learning approach to ransomware detection using opcode features and k-nearest neighbors on windows, *Authorea Preprints*.

Kimmell, J.C. (2022). Analyzing and Explaining Machine Learning Based Online Malware Detection in Cloud (Master's thesis, Tennessee Technological University).