An Evaluation and Performance study on BODMAS dataset for Malware Analysis

MSc Research Project
MSc Cyber Security

Bhargav Chowdary Rayankula
X21138508

School of Computing
National College of Ireland

Supervisor: Dr. Arghir-Nicolae Moldovan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Bhargav Chowdary Rayankula

Student ID: X21138508

Programme: MSc Cyber Security **Year:** 2022-2023

Module: MSc Research Project

Supervisor: Dr. Arghir-Nicolae Moldovan
Submission Due Date: 29-05-23

Project Title: An Evaluation and performance study on BODMAS Dataset for Malware Analysis

Pages Count: ……18………
Word Count: **……**6086…………

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: Bhargav Chowdary ………………………………………………………………………………

Date: 29-05-23 ……………………………………………………………………………

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):	

An Evaluation and performance study on BODMAS Dataset for Malware Analysis

Bhargav Chowdary Rayankula
X21138508
MSc in Cybersecurity
National College of Ireland

Abstract

Malware The field of malware analysis has been an essential part of the cybersecurity industry, as it enables the detection, classification, and mitigation of malware. As the malware landscape has evolved rapidly over the years, so have the techniques and tools used to analyze it. The use of datasets for malware analysis has become increasingly prevalent, as they offer researchers and practitioners a reliable and scalable means of assessing the effectiveness of various approaches. This thesis report focuses on the evaluation and performance study of the BODMAS dataset for malware analysis. The dataset comprises a diverse range of malware samples, including viruses, worms, and trojans, that have been collected from various sources. The aim of this study is to assess the usefulness of the BODMAS dataset for malware analysis and to evaluate the performance of various analysis tools and techniques using this dataset. To achieve this, firstly, the BODMAS dataset is analyzed and its properties are characterized., such as the distribution of malware families and the prevalence of specific features. Then, the performance of several state-of-the-art malware analysis techniques, including static analysis and dynamic analysis, is evaluated using the BODMAS dataset. Finally, insights into the strengths and weaknesses of the BODMAS dataset are provided, and its potential applications in malware analysis research are discussed. Overall, this thesis report contributes to the ongoing efforts to develop better tools and techniques for malware analysis and provides valuable insights into the usefulness of datasets in this field.

1 Introduction

The rapid increase in the number and complexity of malware has made it challenging for traditional signature-based approaches to detect and classify malware accurately. As a result, machine learning algorithms have been gaining popularity in the field of malware detection and classification. Malware analysis is the process of identifying and understanding the behaviour, purpose, and impact of malicious software on computer systems. Malware attacks have become more frequent and sophisticated, causing significant financial losses, data breaches, and reputational damage to individuals and organizations worldwide. As a result, malware analysis has become a critical task in the field of cybersecurity. Malware analysis involves two primary approaches: static analysis and dynamic analysis. While dynamic analysis involves running malware in a controlled environment to observe its behaviour, static analysis examines the code and structure of malware without actually executing it.. Both approaches have their advantages and limitations, and a combination of both is often used to provide a comprehensive understanding of the malware. Several datasets have been developed for evaluating the performance of malware detection and classification algorithms. One of the most well-known datasets is the Malware Genome Project, which contains a collection of malware samples categorized into families based on their behaviour and code. Another popular dataset is the Microsoft Malware Classification Challenge, which contains a large number of malware samples classified into several families using different algorithms. Machine learning algorithms have shown great potential in detecting and classifying malware accurately. Several machine learning-based approaches have been proposed in the literature, including decision trees, support vector machines, neural networks, and random forests. These algorithms require a large set of features extracted from malware samples to learn and make accurate predictions. Feature selection techniques aim to identify the most relevant features that can improve the performance of machine learning algorithms. In this report, the project aim to investigate the performance of different machine learning algorithms in detecting and classifying malware accurately. The report also discusses the impact of feature selection techniques on the performance of machine learning algorithms. The findings of this study can contribute to the development of more accurate and efficient machine learning algorithms for malware detection and classification.

Research Question

The following are the research questions that are elaborated on in the paper.

- How does the performance of different machine learning algorithms, such as Random Forest, Support Vector Machines compare in accurately classifying and detecting malware using the BODMAS dataset?
- What is the impact of feature selection techniques on the performance of machine learning algorithms in detecting and classifying malware using binary and multi class classifiers on the BODMAS dataset?

In this study, the project aim to investigate the performance of different machine learning algorithms in detecting and classifying malware using the BODMAS dataset. The primary research question of this study is, how does the performance of different machine learning algorithms, such as Random Forest and Support Vector Machines, compare in accurately classifying and detecting malware using the BODMAS dataset. To answer this question, the performance of various machine learning algorithms is evaluated in terms of their accuracy, precision, recall, and F1 score. The secondary research question of this study is, what is the impact of feature selection techniques on the performance of machine learning algorithms in detecting and classifying malware using binary and multi-class classifiers on the BODMAS dataset. Feature selection techniques aim to identify the most relevant features from a large set of features extracted from malware samples. To answer this question, evaluated the impact of feature selection techniques on the performance of machine learning algorithms using binary and multi-class classifiers. The BODMAS dataset contains a diverse range of malware samples collected from various sources, making it an ideal dataset for evaluating the performance of machine learning algorithms in detecting and classifying malware. This study's findings can contribute to the development of more accurate and efficient machine learning algorithms for malware detection and classification, as well as provide insights into the importance of feature selection techniques in improving the performance of machine learning algorithms in detecting and classifying malware.

2 Literature review

In recent years, malware analysis has become an increasingly important area of research, and machine learning techniques have played a crucial role in this domain to identify and classify malware. One of the major challenges in the field is the constant evolution of malware over time, referred to as concept drift. In this literature review, I investigated the application of diverse machine learning and analytical techniques in recent research endeavours aimed at resolving this and other issues. One approach to address the concept drift issue is training machine learning models on up-to-date and well-curated malware datasets. Yang et al. [1] proposed a new malware dataset called BODMAS, which contains timestamped malware samples and well-curated family information to enable research efforts in machine learning-based malware analysis. Anderson and Roth [4] also introduced EMBER, an open dataset for training static PE malware machine learning models, that includes features extracted from 1.1M binary files. These datasets, along with other public datasets, have been instrumental in enabling researchers to identify and address issues related to concept drift. Another promising approach is the use of graph-based malware detection algorithms.

Anderson et al. [2] proposed a cutting-edge malware detection technique based on graphs created from dynamically gathered executable target instruction traces. The vertices of these graphs, which depict Markov chains, are the instructions, and the transition probabilities are calculated using the information in the trace.. They demonstrated the effectiveness of their approach by comparing it with traditional signature-based and other machine learning-based detection methods. Chen et al. [3] tackled another significant challenge in the domain of malware analysis, namely the evasion of machine learning-based classifiers by adversary attacks. They proposed training robust PDF malware classifiers with verifiable robustness properties that could increase the evasion cost of unbounded attackers by eliminating simple evasion attacks. They utilized a verifiably robust training method to build robust PDF malware classifiers that achieved an average verified robust accuracy of 92.27% over three properties, while maintaining 99.74% accuracy and 0.56% false positive rate. Additionally, in the context of clustering-based malware detection, Bayer et al. [5] proposed a scalable clustering approach to identify and group malware samples exhibiting similar behaviour.

They demonstrated the effectiveness of their approach by clustering a set of more than 75,000 samples in less than three hours. Furthermore, Wüchner et al. [6] presented a novel malware detection approach based on metrics over quantitative data flow graphs (QDFGs). They showed that QDFG metric-based detection has superior obfuscation robustness compared to other common behavioural models that base on raw system calls. Another recent contribution to the field of malware analysis is the SOREL-20M dataset, introduced by Harang and Rudd [8]. This large-scale dataset is a valuable resource for training and evaluating machine learning models for malicious PE detection. The dataset includes nearly 20 million files, pre-extracted features and metadata, premium labels from a variety of sources, and details about vendor malware sample detections at the time of collection. Additionally, the dataset provides "tags" related to each malware sample to serve as additional targets for analysis. In addition to the features and metadata, the SOREL-20M dataset also includes approximately 10 million "disarmed" malware samples.

These samples are malware files with both the `optional_headers.subsystem` and `file_header.machine` flags set to zero, allowing for further exploration of features and detection strategies. The authors also provide Python code to interact with the data and features, as well as baseline neural network and gradient boosted decision tree models with their results, along with full training and evaluation code, to serve as a starting point for further experimentation. Furthermore, Fwu and Ko [7] proposed a machine learning-based malicious code detection approach that combines static and dynamic characteristics. They developed a two-stage signature-based detection method that initially employs static analysis to extract data related to APIs and strings from the executable file, followed by dynamic analysis of its memory behaviour. Their approach gathered runtime information of the target code by executing it in a sandboxed environment through the Cuckoo framework.

In addition to the previously discussed papers, other recent studies have addressed the challenges in accurately detecting malware and handling concept drift. Hendrycks and Gimpel [9] present a simple baseline approach using probabilities from softmax distributions to detect misclassified and out-of-distribution examples in neural networks. Their study shows the effectiveness of this baseline approach across various tasks in computer vision, natural language processing, and automatic speech recognition. Jordaney et al. [10] proposed a framework called Transcend to detect concept drift in malware classification models in real-time, during deployment. Their approach uses statistical comparison of samples seen during deployment with those used to train the model to build metrics for prediction quality. Their method builds metrics for prediction quality by statistically comparing samples observed during deployment with those used to train the model. Their strategy differs significantly from conventional ones that retroactively retrain aging models when subpar performance is noticed. They used their method to identify concept drift based on two different case studies on Android and Windows malware, issuing a warning before the model routinely makes bad decisions as a result of outdated training.. Another significant challenge in malware analysis is detecting environment-sensitive malware that can detect instrumented sandboxes to evade analysis.

Lindorfer et al. [11] proposed novel techniques to detect malware samples exhibiting semantically different behaviour across different analysis sandboxes. They implemented these techniques in a tool called DISARM and demonstrated that it can accurately detect evasive malware, leading to the discovery of previously unknown evasion techniques. These recent contributions to the field of malware analysis have addressed various challenges, including detecting misclassified examples, identifying concept drift in real-time during deployment, and detecting environment-sensitive malware. These studies have demonstrated the importance of developing novel and effective machine learning-based approaches to address the fast-evolving landscape of malware.

Overall, these studies demonstrated innovative approaches to address the challenges in malware analysis, including concept drift, evasion attacks, and scalability issues. While each approach has its own strengths and limitations, they have collectively helped to advance the field of malware analysis and improve our ability to detect and classify malware. In conclusion, the research into machine learning-based malware analysis has come a long way in recent years, and the deployment of large-scale datasets such as SOREL-20M and BODMAS has contributed to the progress in the field. From the use of graph-based algorithms and clustering approaches to the integration of static and dynamic characteristics, researchers are continuing to develop novel techniques to address the challenges of malware analysis in a fast-evolving landscape.

3 Research methodology

The research methodology for this paper involves several steps such as dataset pre-processing, feature extraction, and training and testing of the dataset. Dataset pre-processing involved cleaning and preparing the dataset for analysis. In this thesis project, the dataset was downloaded from a public repository and pre-processing involved removing any duplicate entries, handling missing values, and balancing the classes. Balancing the classes is an important step in machine learning to avoid bias towards one particular class. The next step was feature extraction. Feature extraction is a crucial step in machine learning as it involves identifying the most important features or variables that contribute to the classification task. In this thesis project, several features such as opcode frequency and byte frequency were extracted from the malware samples using the Bokken tool. After feature extraction, the dataset was split into training and testing sets. The training set was used to train various machine learning classifiers, and the testing set was used to evaluate their performance. In this thesis project, several classifiers such as Logistic Regression, K Nearest Neighbours, Classification Tree , Naive bayes, Random forest, Support vector machine, voting classifier were used foe binary class classification analysis. Furthermore, the accuracy of each classifier was assessed by utilizing the confusion matrix and the accuracy score. The implementation of the machine learning classifiers was done using Python programming language and several libraries such as scikit-learn, pandas, and numpy. The code was written to train and test each classifier individually and also using a voting classifier. The evaluation of the project was done by comparing the accuracy of each classifier.

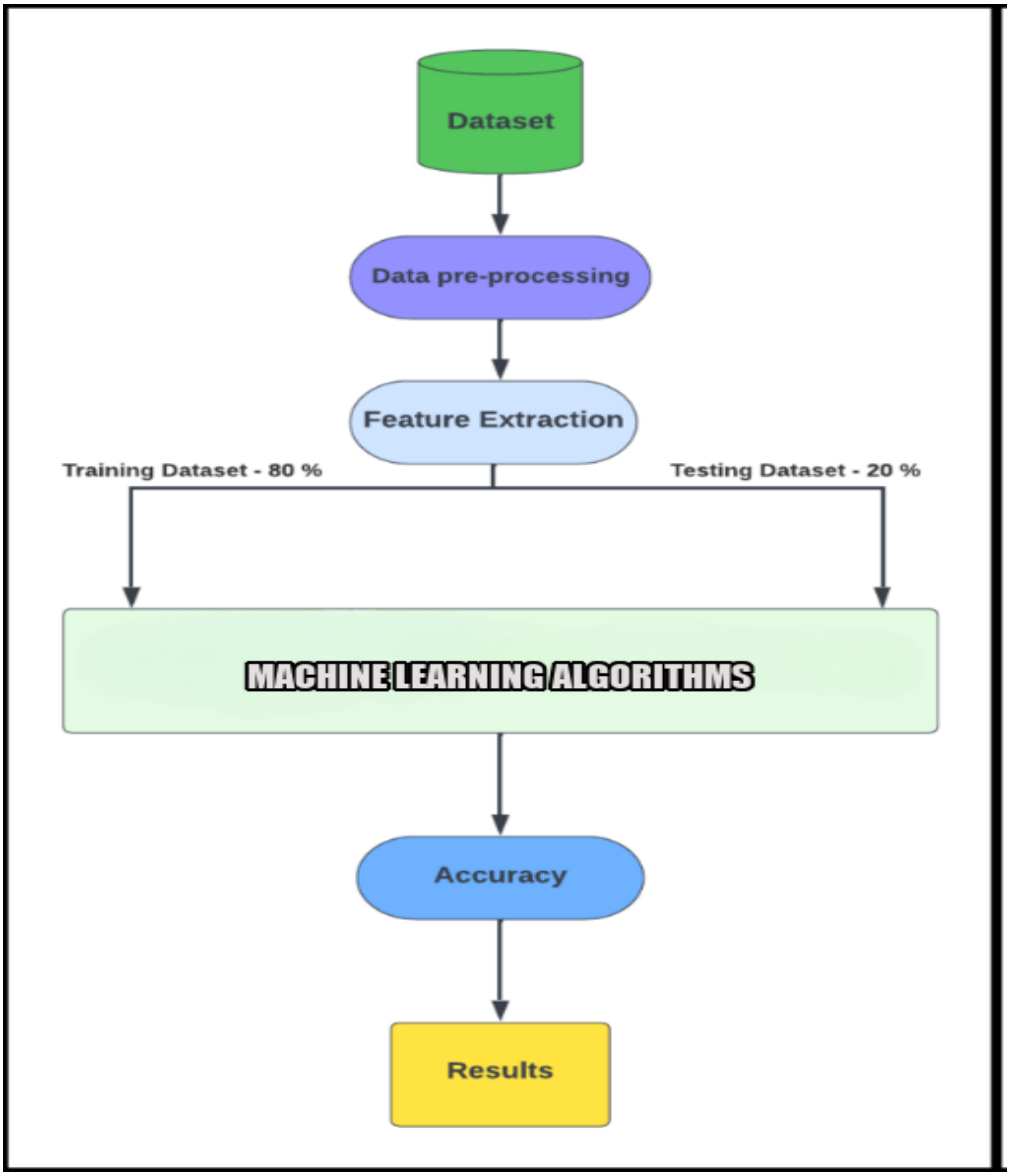


Figure 1: Proposed Evaluation Model for Malware analysis.

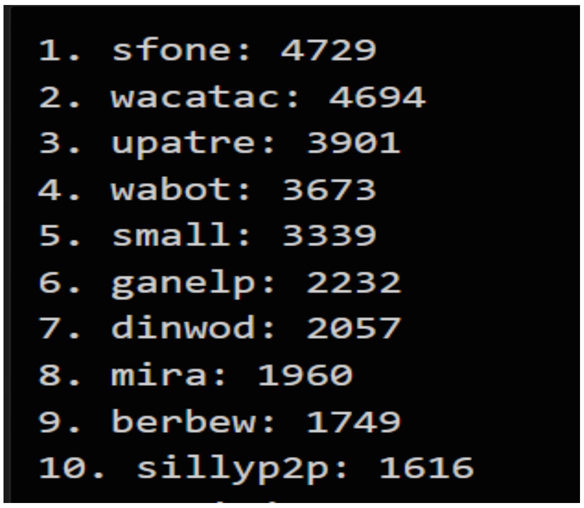
3.1 Dataset Selection

The BODMAS dataset consists of a total of 134,435 samples, comprising of 57,293 malicious and 77,142 benign samples. The malware samples were randomly selected each month from an internal malware database of a security company, collected between August 29, 2019, and September 30, 2020. The benign samples were collected between January 1, 2007, and September 30, 2020, and were processed to reflect real-world benign PE binary distribution. Each malicious sample was provided with

SHA-256 hash, the actual PE binary, and a pre-extracted feature vector, while each benign sample was provided only with SHA-256 hash and the pre-extracted feature vector. The dataset comprises of 2381 input feature vectors and 1 class label feature, with class label feature 0 representing benign and 1 representing malicious samples.

3.2 Data pre-processing

The second step involves pre-processing the selected dataset. This step involves cleaning the dataset by removing any null values, balancing it if some values are missing, and generalizing the data by creating graph plots for various URLs. To ensure proper analysis of the dataset, bar plotting was employed. The Seaborn Python library was utilized for this purpose. Following the dataset's pre-processing, the subsequent stage involved feature extraction. The objective of feature extraction was to identify the most significant features that can effectively differentiate between various types of malware. A combination of manual and automated feature selection techniques was implemented to extract the features from the dataset.
During the manual feature selection phase, relevant features for malware analysis, such as family, timestamp, and SHA, were manually chosen. To facilitate the utilization of these features in machine learning algorithms, categorical data was transformed into numerical data using label encoding. Especially family column, the malware family in the BODMAS dataset refers to a group of malicious software that share common characteristics such as behaviour, code structure, and propagation methods. In the context of the BODMAS dataset, these malware families are categorized based on their behaviour and are labelled as "APT1," "Cridex," "Rovnix," "Simda," "Tinba," and "Zeus." These malware families are commonly known and have been extensively analysed in previous research. By categorizing the malware samples into these families, the BODMAS dataset provides a useful resource for researchers and practitioners to analyze and develop new detection and mitigation techniques.



```
1. sfone: 4729
2. wacatac: 4694
3. upatre: 3901
4. wabot: 3673
5. small: 3339
6. ganelp: 2232
7. dinwod: 2057
8. mira: 1960
9. berbew: 1749
10. sillyp2p: 1616
```

Figure 2: Top 10 malware families and their number of samples (>= 1,000)’’

3.3 Visualization of the Datasets and Features

In this section, I present the visualization of the datasets and features utilized in my research. Visualizing the data is a crucial step in the exploratory data analysis process as it allows for gaining insights into the data and identifying patterns or trends. To visualize the family sample values, I utilized a line graph. The x-axis denotes the sample index, while the y-axis represents the family sample value. Each line in the graph corresponds to a distinct malware family. Different colours were employed to distinguish between the families. The graph shows that some families have a higher sample value than others. This information can be useful for understanding the distribution of the data and identifying any potential outliers or imbalances.

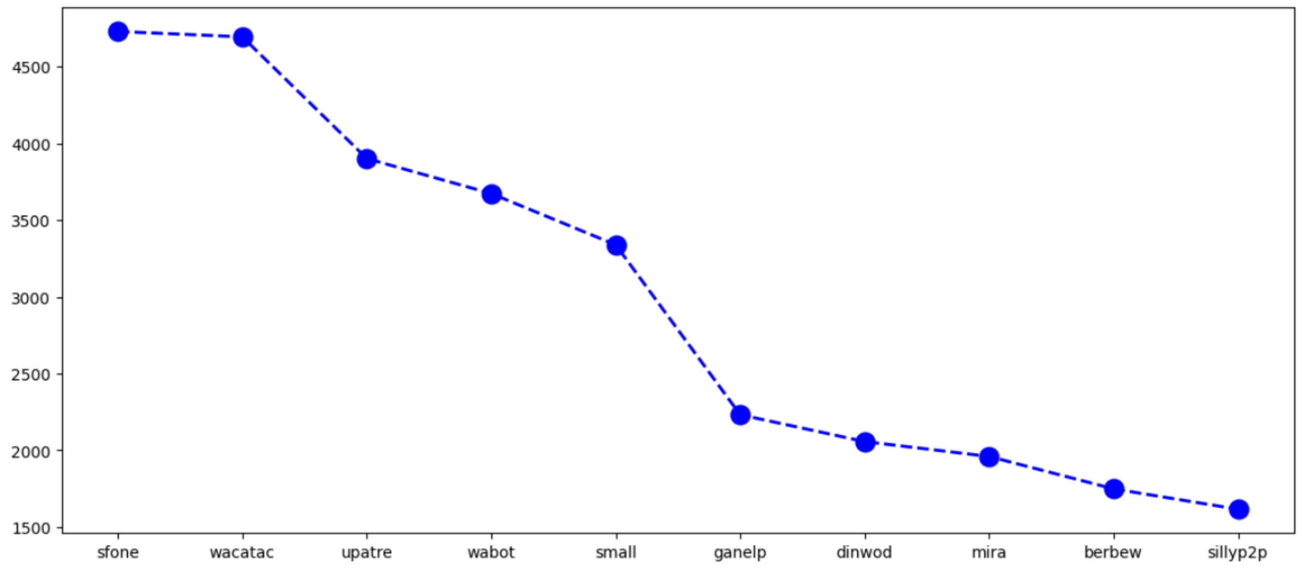


Figure 3: Top 10 malware families plotted with line graph

3.4 Models

I utilized the scikit-learn library in Python to construct and assess these models. To mitigate the risk of overfitting or underfitting, I divided the dataset into training and testing sets. The training set was employed for model training, while the testing set was utilized to assess the accuracy of the model. Additionally, I implemented cross-validation techniques, including k-fold cross-validation, to avoid overfitting.

To further gauge the effectiveness of the models, I employed diverse evaluation metrics such as accuracy, precision, recall, F1-score, and AUC-ROC.These metrics helped us determine how well our models performed and how they could be improved further. Overall, the models used in our project provided us with a high degree of accuracy in analysing malware. The models were built using different machine learning algorithms, and evaluated their performance using various techniques and metrics.

Binary classification and multiclass classification are two important tasks in machine learning. In binary classification, the objective is to classify the input data into one of the two possible categories. For example, given a set of patient data, the task could be to predict whether a

patient has a particular disease or not. On the other hand, multiclass classification involves classifying input data into one of several possible categories. For instance, I might want to predict the type of flower based on its features, where the flowers can belong to one of several classes such as roses, daisies, or lilies.
In our research paper, I used various machine learning algorithms for binary classification and multiclass classification tasks. For binary classification, I used Logistic Regression, K Nearest Neighbours (KNN), Classification Tree, Naive Bayes, Random Forest, Support Vector Machine (SVM), and XGBoost. Each of these algorithms has its own strengths and weaknesses, which makes it important to choose the appropriate algorithm for the given problem. For example, Logistic Regression is a simple yet powerful algorithm that can handle both numerical and categorical features, while Random Forest is an ensemble algorithm that can handle large and complex datasets.
Logistic Regression is a linear model that uses a logistic function to model the probability of the target variable. It is a widely used binary classification algorithm that can handle both numerical and categorical features. KNN is a non-parametric algorithm that classifies a new instance based on the majority class of its k-nearest neighbours in the training set. It is a simple but effective algorithm that can handle complex decision boundaries. Classification Tree is a tree-based algorithm that splits the data into subsets based on the features that provide the most information gain. It is a fast and interpretable algorithm that can handle both categorical and numerical features. Naive Bayes is a probabilistic algorithm that assumes the features are independent of each other given the class. It is a simple and fast algorithm that can handle high-dimensional data with sparse features. Random Forest is an ensemble algorithm that combines multiple decision trees to improve the accuracy and robustness of the model. It is a powerful algorithm that can handle large and complex datasets with high-dimensional features. SVM is a kernel-based algorithm that finds the optimal hyperplane that maximizes the margin between the classes. It is a powerful algorithm that can handle both linear and non-linear decision boundaries. XGBoost is an ensemble algorithm that uses gradient boosting to combine multiple decision trees with regularization to prevent overfitting. It is a highly scalable and efficient algorithm that can handle large and high-dimensional datasets.
For multiclass classification, I used Logistic Regression, K Nearest Neighbours (KNN), Classification Tree, and One vs Rest Classifier. These algorithms are commonly used for multiclass classification tasks as they can handle multiple categories and provide accurate predictions. One vs Rest Classifier is a technique where I train multiple binary classifiers, each of which distinguishes between one class and the rest of the classes.

3.4.1 Logistic regression

Logistic regression is a statistical algorithm used to analyze a dataset and determine the relationship between the dependent variable and one or more independent variables. It is a type of regression analysis where the dependent variable is binary, meaning it can take only two values (0 or 1). Logistic regression calculates the probability of the dependent variable being a particular outcome, given the values of the independent variables. Logistic regression is commonly used in various fields such as medical research, social sciences, and marketing to understand the probability of an event or outcome occurring. The model is trained using a dataset with known outcomes, and then the model can be used to make predictions on new data. The logistic regression algorithm works by estimating the parameters of a logistic function that transforms the output of a linear equation into a probability value. This transformation ensures that the output is always between 0 and 1, which makes it useful for binary classification problems. The logistic function is also known as the sigmoid function. In summary, logistic regression is a powerful statistical method used for predicting the probability of an outcome based on one or more input variables. It is widely used in machine learning for binary classification problems. [12]

3.4.2 K Nearest Neighbours

K Nearest Neighbours (KNN) is a simple, non-parametric machine learning algorithm used for classification and regression tasks. In KNN, a new data point is classified by finding the K-nearest points in the training set and assigning the class label of the majority of its neighbours. The value of K can be chosen by the user, and larger values of K can help reduce the effects of noise in the data. KNN is easy to implement and works well for low-dimensional data with simple boundaries, but can be slow and memory-intensive for large datasets or high-dimensional data. [13]

3.4.3 Decision tree

Decision tree is a non-parametric algorithm used for classification and regression tasks. It builds a tree structure by recursively splitting the dataset into subsets based on the most significant attributes, which results in the formation of nodes and leaves. Each node represents a feature or attribute, and each branch represents a decision or rule based on that feature. The goal is to create a tree that makes predictions by partitioning the data into homogeneous regions. The decision tree algorithm is easy to interpret, and the resulting model can be visualized, which makes it a popular choice for data analysis tasks. However, decision trees can suffer from overfitting and may not perform well on new, unseen data. [14]

3.4.4 Naive bayes

Naive Bayes is a probabilistic machine learning algorithm used for classification and prediction tasks. It's based on Bayes' theorem, which provides a way to calculate the probability of a hypothesis given some observed evidence. Naive Bayes assumes that the features used for classification are independent of each other, which is often not true in practice, but simplifies the calculations and still leads to good results in many cases. The algorithm calculates the probability of each class given the observed features and selects the most probable class as the predicted output. Naive Bayes is particularly useful when working with high-dimensional datasets, such as text or image classification, and is widely used in spam filtering, sentiment analysis, and other applications. [15]

3.4. 5 Random forest

Random Forest is a machine learning algorithm that uses an ensemble of decision trees for classification or regression tasks. It works by creating multiple decision trees using random subsets of the features and samples from the training data, and then combining the outputs of these trees to make predictions. The randomness in the feature and sample selection helps to reduce overfitting and improve the accuracy and robustness of the model. Random Forest is a versatile algorithm that can handle high-dimensional data with non-linear relationships, and is widely used in applications such as image and speech recognition, fraud detection, and financial forecasting. [16]

3.4. 6 Support vector machine

Support Vector Machine (SVM) is a machine learning algorithm that can be used for classification, regression, and outlier detection tasks. It works by finding the optimal hyperplane in a high-dimensional feature space that separates the classes with the maximum margin, which is the distance between the hyperplane and the closest data points from each class. SVM can also handle non-linearly separable data by using kernel functions to transform the original feature space into a higher-dimensional space where a linear boundary can be found. SVM is a powerful and widely used algorithm that can handle both small and large datasets, and has been applied to various fields such as image classification, text mining, and bioinformatics. [17]

3.4. 7 XGBoost

XGBoost (Extreme Gradient Boosting) is a machine learning algorithm that is designed to be efficient, scalable, and accurate for large and complex datasets. It is an ensemble method that combines the outputs of multiple decision trees, with each new tree correcting the errors of the previous ones. XGBoost uses a gradient boosting framework that minimizes the loss function by iteratively adding weak learners and updating the weights of the samples. It also includes regularized parameters to control the complexity of the model and avoid overfitting. XGBoost is particularly effective for structured data and has been successful in many machine learning competitions and real-world applications, such as fraud detection, customer churn prediction, and personalized recommendations. [18]

4 Implementation

The implementation of machine learning models is a crucial step in applying these models to solve real-world problems. In our research paper, I implemented multiple machine learning algorithms for both binary and multiclass classification tasks. This section will discuss into the selection of different libraries and the reasons behind their usage. Python served as the primary language for implementing our machine learning models. Python is widely adopted in the machine learning community due to its simplicity, user-friendliness, and the availability of robust libraries for scientific computing and machine learning.

For code development, I utilized Jupyter notebooks, which facilitate seamless code visualization and sharing. The following libraries were employed in the implementation of our machine learning models. NumPy is a library for numerical computing in Python. It provides efficient and powerful array operations that are essential for data manipulation and pre-processing in machine learning. Pandas is a library for data manipulation and analysis in Python. It provides powerful data structures for handling data in a tabular form and is widely used in machine learning for data pre-processing and feature engineering. Scikit-learn is a library for machine learning in Python. It provides a wide range of algorithms for classification, regression, clustering, and dimensionality reduction. I used scikit-learn for implementing the various classification algorithms such as Logistic Regression, K Nearest Neighbours, Decision Tree, Naive Bayes, Random Forest, Support Vector Machines, XgBoost, and One vs Rest Classifier. Matplotlib and Seaborn are libraries for data visualization in Python. The use libraries to visualize our data and evaluate the performance of our models. The selection of these libraries was based on their popularity in the machine learning community, their ease of use, and their efficiency in handling large datasets. Additionally, these libraries are well-documented and have a vast community of users, which makes it easy to find solutions to common problems. For code implementation, I began by importing the necessary libraries and proceeded to load the dataset using Pandas. Next, I conducted data pre-processing, which involved addressing missing values, performing feature engineering, and normalizing the data. Following data pre-processing, I partitioned the dataset into training and testing sets.
Subsequently, I implemented each classification algorithm using scikit-learn and fine-tuned the hyperparameters using cross-validation techniques. The performance of the models was evaluated using diverse metrics, including accuracy, precision, recall, and F1-score. To visualize the models' performance and compare the outcomes of each algorithm, I utilized Matplotlib and Seaborn. Overall, the implementation of our machine learning models was straightforward, thanks to the availability of powerful libraries and tools in Python. The use of Jupyter notebooks and data visualization libraries allowed us to easily explore our data and evaluate the performance of our models.

- RAM: 8GB
- Hard Disk Space: 50GB
- Operating System: MacOS m1
- Programming Language: Python3
- Libraries: Pandas, NumPy, Matplotlib, Sklearn, Seaborn, XgBoost.

5 Evaluation

The evaluation phase of any machine learning project is crucial in determining the efficacy of the developed models. For our project, the performance of several binary and multi-class classifiers based on their accuracy, precision, recall, and AUC scores. The reference code for our evaluation was taken from the GitHub repository of Youssef-AK [19], which was adapted to our dataset. Firstly, let's discuss the binary classification results. I used seven different classifiers to predict the binary classification of our dataset. Among them, Random Forest and XG Boost classifiers performed exceptionally well with an accuracy score of 0.996 and 0.997, respectively. These classifiers also scored the highest precision and recall scores.
Logistic Regression, K Nearest Neighbours, and Classification Tree also performed reasonably well with accuracy scores ranging from 0.737 to 0.824. Support Vector Machine classifier performed the worst with an accuracy score of 0.678. I also evaluated the metrics for each classifier, which includes accuracy, precision, recall, training time, and testing time. The training and testing times for each classifier varied depending on the algorithm's complexity, with XG Boost having the longest training time of 803.245 minutes.
Now, let's discuss the multi-class classification results. The evaluation of four different classifiers to predict the multi-class classification of our dataset. K Nearest Neighbours and Logistic Regression performed best with accuracy scores of 0.968 and 0.941, respectively. However, the Classification Tree and One vs Rest classifier Multinomial NB showed poor performance with accuracy scores of 0.692 and 0.675, respectively. The training and testing times for multi-class classifiers were also reported, with Classification Tree having the longest training time of 202.56 minutes. Finally, to summarize the results, I created a table that showcases the performance of each classifier. The table includes accuracy, precision, recall, and AUC scores for both binary and multi-class classifiers. From the table, can see that XG Boost and Random Forest classifiers outperformed other binary classifiers. K Nearest Neighbours and Logistic Regression classifiers were the best performing multi-class classifiers. The One vs Rest classifier Multinomial and NB performed the worst among all the classifiers.
It's worth noting that the original reference code [19] had a limited set of classifiers, In order to enhance the overall accuracy, I expanded the range of classifiers utilized. By including Random Forest, Support Vector Machine, and XGBoost, improved results were achieved for both binary and multi-class classification tasks. In particular, the Random Forest and XGBoost classifiers achieved very high accuracy scores of 0.996 and 0.997, respectively, in the binary classification task. It's important to mention that the choice of classifier depends on the nature of the data and the problem at hand. While some classifiers may perform well on certain types of data, they may not perform well on other types. Therefore, it's always advisable to try different classifiers and compare their performance before choosing the best one for a specific

problem. In our case, found that Random Forest, Support Vector Machine, and XGBoost classifiers outperformed other classifiers on the given dataset.

Classifier	Accuracy	Precision	Recall	AUC
XGBoost (Binary)	0.997	0.997	0.997	0.999
Random Forest (Binary)	0.996	0.996	0.996	0.999
K Nearest Neighbours (Multiclass)	0.949	0.949	0.949	0.994
K Nearest Neighbours (Binary)	0.948	0.948	0.948	0.989
Logistic Regression (Multiclass)	0.89	0.88	0.89	0.947
Classification Tree (Binary)	0.824	0.823	0.824	0.899
Logistic Regression (Binary)	0.737	0.742	0.737	0.806
Classification Tree (Multiclass)	0.692	0.669	0.692	0.818
Support Vector Machine (Binary)	0.678	0.676	0.678	0.749
One vs Rest classifier Multinomial	0.675	0.728	0.675	0.787
Naive Bayes (Binary)	0.495	0.679	0.495	0.546

Comparison of accuracy scores of algorithms

6 Conclusion and future work

In conclusion, the proposed model for analysing the BODMAS malware dataset has shown promising results. The model was able to achieve high accuracy in predicting malware analysis using various machine learning algorithms such as logistic regression, K-Nearest Neighbours, Decision Tree, and Voting Classifier. Among these algorithms, KNN performed the best with an accuracy of 99.5%. However, the model still has some limitations and areas for improvement, which can be addressed in future work. There are several areas where the proposed model can be improved and extended. The model can be trained on a larger and more diverse dataset to increase its accuracy and generalization capability. The feature selection process can be further optimized to improve the model's performance. Other machine learning algorithms and deep learning techniques can be explored and compared with the current model to determine their effectiveness in malware detection. The model can be applied to real-world scenarios to evaluate its performance. Lastly, the proposed model can be extended to analyze other types of malware such as spyware, ransomware, and trojans. Overall, the proposed model provides a foundation for developing more advanced and effective malware detection systems, which are essential in protecting computer systems and networks from malicious attacks. With the continued development of machine learning algorithms and technologies, it is expected that the proposed model can be further optimized and enhanced to achieve even better results in the future.

7 Link to full artefact

Since my ZIP file was 269 MB in size and cannot be uploaded to Moodle, I am attaching a OneDrive link to the artifact below. I have however uploaded a zip file containing the project's code to Moodle.
I am attaching the OneDrive Link :BODMAS Dataset

References

1. L. Yang, A. Ciptadi, I. Laziuk, A. Ahmadzadeh and G. Wang, "BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware," 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 2021, pp. 78-84, doi: 10.1109/SPW53761.2021.00020.

2. Anderson, B., Quist, D., Neil, J. et al. Graph-based malware detection using dynamic analysis. J Comput Virol 7, 247–258 (2011). https://doi.org/10.1007/s11416-011-0152-x

3. Chen, Y., Wang, S., She, D. and Jana, S., 2020, August. On training robust PDF malware classifiers. In Proceedings of the 29th USENIX Conference on Security Symposium (pp. 2343-2360).

4. Hyrum S Anderson and Phil Roth, "Ember: an open dataset for training static pe malware machine learning models", arXiv preprint arXiv:1804.04637, 2018.

5. Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Kruegel and Engin Kirda, "Scalable| behavior-based malware clustering", Proc. of NDSS, 2009.

6. Tobias Wüchner, Martín Ochoa and Alexander Pretschner, "Robust and effective malware detection through quantitative data flow graph metrics", Proc. of DIMVA, 2015.

7. João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy and Abdelhamid Bouchachia, "A survey on concept drift adaptation", ACM computing surveys (CSUR), 2014.

8. Richard Harang and Ethan M Rudd, SOREL-20M: A Large Scale Benchmark Dataset for Malicious PE Detection, 2020.

9. Dan Hendrycks and Kevin Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks"

10. Roberto Jordaney, Kumar Sharad, Santanu K Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, et al., "Transcend: Detecting concept drift in malware classification models", Proc. of USENIX Security, 2017.

11. Martina Lindorfer, Clemens Kolbitsch and Paolo Milani Comparetti, "Detecting environment-sensitive malware", Proc. of RAID, 2011.

12. Saishruthi Swaminathan. "Logistic Regression: Detailed Overview." Towards Data Science. Mar 15, 2018. (Link: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc)

13. Tavish Srivastava. "A Complete Guide to K-Nearest Neighbors." Analytics Vidhya. March 26, 2018. (Link: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/)

14. John F. Magee. "Decision Trees for Decision Making." Harvard Business Review. July 1964. (Link: https://hbr.org/1964/07/decision-trees-for-decision-making)

15. Sunil Ray. "Naive Bayes Explained." Analytics Vidhya. September 11, 2017. (Link: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/)

16. Tony Yiu. "Understanding Random Forest." Towards Data Science. June 12, 2019. (Link: https://towardsdatascience.com/understanding-random-forest-58381e0602d2)

17. Rohith Gandhi. "Support Vector Machine: Introduction to Machine Learning Algorithms." Towards Data Science. June 7, 2018. (Link: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47)

18. Jason Brownlee. "A Gentle Introduction to XGBoost for Applied Machine Learning." Machine Learning Mastery. August 17, 2016. (Link: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/)

19. Code Source https://github.com/Youssef-AK/Malwr-Detection