Enhancing Malware Detection and Family Identification with Machine Learning Models and Explainable AI

A Project Work Submitted in Partial Fulfillment
of the requirements for the Degree of

Bachelor of Technology
in
Computer Science and Engineering

Submitted by

Souptik Dutta	(202002021001)
Jyoti Bikash Choudhury	(202002022101)
Ritik Raj	(202002022097)

Under the supervision of

Dr. Amitava Nag

Department of Computer Science and Engineering

केन्द्रीय प्रौद्योगिकी संस्थान कोकराझार

CENTRAL INSTITUTE OF TECHNOLOGY KOKRAJHAR

(Deemed to be University under MoE, Govt. of India)

BODOLAND TERRITORIAL AREAS DISTRICTS :: KOKRAJHAR :: ASSAM :: 783370

Website: www.cit.ac.in

June 2024

Department of Computer Science and Engineering

केन्द्रीय प्रौद्योगिकी संस्थान कोकराझार

CENTRAL INSTITUTE OF TECHNOLOGY KOKRAJHAR

(An Autonomous Institute under MHRD)

Kokrajhar – 783370, BTAD, Assam, India

CERTIFICATE OF APPROVAL

This is to certify that the work embodied in this project entitled **Enhancing Malware Detection and Family Identification with Machine Learning Models and Explainable AI** submitted by Souptik Dutta, Jyoti Bikash Choudhury and Ritik Raj to the Department of **Computer Science & Engineering**, is carried out under our direct supervisions and guidance.

The project work has been prepared as per the regulations of Central Institute of Technology Kokrajhar and I strongly recommend that this project work be accepted in partial fulfillment of the requirement for the degree of B.Tech.

Supervised by:

Dr. Amitava Nag

(Professor, Dept. of CSE)

Countersigned By:

Dr. Amitava Nag

(Head of the Department)

Department of Computer Science and Engineering

केन्द्रीय प्रौद्योगिकी संस्थान कोकराझार

CENTRAL INSTITUTE OF TECHNOLOGY KOKRAJHAR

(An Autonomous Institute under MHRD)

Kokrajhar – 783370, BTAD, Assam, India

CERTIFICATE BY THE BOARD OF EXAMINERS

This is to certify that the work entitled **Enhancing Malware Detection and Family Identification with Machine Learning Models and Explainable AI** submitted by Souptik Dutta, Jyoti Bikash Choudhury and Ritik Raj to the Department of **Computer Science & Engineering**, of Central Institute of Technology, Kokrajhar has been examined and evaluated..

The project work has been prepared as per the regulations of Central Institute of Technology and qualifies to be accepted in partial fulfillment of the requirement for the degree of B. Tech.

Mr. Mithun Karmakar
Assistant Professor
Dept. of CSE

(**Signature of the Examiner**)

Abstract

This project dives deep into the topic of Machine Learning and Explainable Artificial Intelligence (XAI) with the aim to detect and identify various malicious software and the family it belongs to. This study looks into how XAI methods in association with machine learning models can improve malware detection and family identification. A large diverse dataset has been taken into consideration consisting of various kinds of hash values and entropy measures among the features. For dealing with the hashes, a hash-encoding algorithm has been proposed. An effective feature selection using the Boruta algorithm is performed to extract the best features. Standard classification machine learning models have been implemented to distinguish and classify between malware and benignware, with Random Forest (RF) producing the best results. XAI methods have been utilised for model explainability and accountability. Clustering methods have also been utilised to identify and cluster different malicious samples into their respective families and also identify potential zero-day malware in the process.

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to our guide, **Dr. Amitava Nag** for his valuable guidance, consistent encouragement, personal care and timely help which provided us with an excellent atmosphere for doing our project. In spite of the busy schedule, he has extended his cheerful and cordial support to us, without which we could not have completed our project work.

We express our heartfelt thanks to our Head of the Department, **Dr. Amitava Nag** who has been actively involved and very influential from the start till the completion of our project.

We would also like to thank all the teaching and non-teaching staff of the Computer Science Engineering Department for their constant support and Encouragement.
Last but not least it is our pleasure to acknowledge the support and wishes of our friends and well-wishers, both in academic and non-academic spheres.

Date:

(Signature of the candidate)
Souptik Dutta
(202002021001)

(Signature of the candidate)
Jyoti Bikash Choudhury
(202002022101)

(Signature of the candidate)
Ritik Raj
(202002022097)

Contents

List of Tables

List of Figures

List of Abbreviations

ML	Machine Learning
AI	Artificial Intelligence
PE	Portable Executable
FS	Feature Selection
RFE	Recursive Feature Elimination
XGB	Extreme Gradient Boosting
RF	Random Forest
DT	Decision Tree
TPR	True Positive Rate
TPR	False Positive Rate
FCM	Fuzzy C-Means
WCSS	Within-Cluster Sum of Squares
XAI	Explainable Artificial Intelligence
SHAP	Shapley Additive Explanations
LIME	Local Interpretable Model-agnostic Explanations
ROC	Receiver Operating Characteristics
SHA	Secure Hash Algorithm
MD5	Message Digest Algorithm 5

Chapter 1

Introduction

Today's digital world has malware, or malicious software, as a big problem.[23] Ransomware, spyware, trojans, bots, worms, backdoors, and viruses are only a few of the numerous varieties. Malware attacks are increasing rapidly. In September 2019, there were 17.70 million malware attacks.[12] This shows that the condition is worsening. Additionally, 560,000 new malware samples are discovered every day, making it impossible to manually evaluate each one. This emphasizes the importance of automated methods for categorizing malware into families.

Antivirus vendors play an important role in developing knowledge bases that help identify shared code and similar behaviors across malware samples from the same family. This important information improves classification accuracy, indicating the crucial significance of malware detection in cybersecurity protection.

Malware detection approaches are divided into two categories: signature-based and anomaly detection methods. Signature-based detection uses determined signatures in malware code to identify malicious software [18]. However, it has difficulties in identifying new (zero-day) malware and developing threats such as structural and polymorphic malware. Anomaly detection techniques, on the other hand, are critical in finding new and unknown malware by studying anomalies from typical software behaviors, hence contributing to improved classification accuracy and combating emerging threats.

The landscape of malware analysis and detection is evolving, with innovative technologies emerging to handle the growing threat of zero-day malware. Notably, a novel malware family categorization and clustering method designed for real-time processing of zero-day malware instances makes a substantial contribution to this evolving field.

Zero-day malware is a significant danger to organizational cybersecurity, exploiting undiscovered and unprotected vulnerabilities, making it difficult to identify and defend against. These vulnerabilities, often known as zero-day vulnerabilities, are exploited in the field before software manufacturers issue updates. The period between vulnerability exploitation and patch deployment is known as "day zero".[19] Zero-day malware exploits these flaws, frequently spreading widely before enterprises can build protections.[33]

In 2021, a set of vulnerabilities in Microsoft Exchange was exploited, resulting in the emergence of several zero-day malware variants, including Hafnium.[15] Hafnium is a data-stealing malware that uses Microsoft Exchange exploits to gain access to susceptible servers and steal emails and user credentials.

Traditional cybersecurity tactics are frequently useless in protecting against zero-day malware because they rely on prior knowledge of the vulnerability or exploit, which zero-day threats lack.[39] Clustering offers a promising method for detecting zero-day malware by grouping together comparable samples, even if they have never been seen before and lack a recognized signature. Clustering helps machine learning-based detection systems reduce false positives by focusing attention on clusters of samples that are most likely to be harmful. This strategy not only speeds malware analysis by directing analysts' focus, but it also helps detect and remove outliers that may have been incorrectly categorized as malicious. Thus, clustering is an important instrument in the ongoing fight against zero-day malware and the protection of digital infrastructures.

Our study has been structured in such a way that the literature review of all the related works has been explained in Chapter 2 under the chapter name "Related Works". The working of the Boruta Feature Selection algorithm, the different classification algorithms, the various unsupervised techniques or clustering algorithms along with a detailed discussion about Explainable Artificial Intelligence have been thoroughly covered under Chapter 3 named "Preliminaries". Our proposed architecture and the methods used by us towards achieving our goal are explained in Chapter 4 under the heading "Malware Detection and Classification using ML" respectively. An in-depth analysis of the various results obtained by us has been discussed in Chapter 5 named "Results and Discussion". Finally, our study has been wrapped up and the ideas of the improvement of the project have been mentioned in Chapter 7 named "Conclusion and Future Scope".

1.1 Motivation

The motive for defending against cyber-attacks that exploit previously unknown vulnerabilities. These attacks can hurt both persons and organizations, and they are difficult to identify and protect against. By effectively categorizing and clustering zero-day malware, security professionals may take proactive steps to avert damage and remain ahead of attackers. This initiative is essential because it can improve the security and integrity of digital systems and networks.

The goal is to detect and prevent evil people from deploying hidden defects in software to harm others. We can help preserve computers and gadgets from cyberattacks by finding and categorizing these hidden defects. This is significant because these attacks can inflict harm to individuals and organizations and are difficult to identify and prevent. By completing this project, we can help to make the internet safer and more secure.

1.2 Objective

Our research aims to create a system that can process incoming harmful samples and assign them to existing or new malware families. The system should be able to divide malware samples into recognized malware families or decide whether they should be clustered using a self-organizing map. By effectively clustering malicious software and identifying potential zero-day malware, the system can contribute to the security and integrity of digital systems and networks. Our objectives in this study can be laid out as:

- To implement suitable feature selection and classification methods for effective malware analysis.
- To analyze model explainability and accountability through XAI methods.
- To effectively cluster malicious samples and identify their respective families through clustering methods.

Chapter 2

Related Work

Detection and prevention of malware attacks, especially zero-day malware attacks has been a continuous area of rigorous study and research over the last few years.

Ibrahim S. et al. [14] have developed XRan, a sophisticated ransomware detection technology that uses dynamic analysis. In contrast to conventional techniques, XRan uses a CNN for detection and integrates several behavioral sequences. It outperforms state-of-the-art techniques and achieves up to 99.4% True Positive Rate (TPR) by utilizing LIME and SHAP for explainability. It blends many sequences based on dynamic analysis, each of which represents a distinct executable view, to enhance the feature space.

Fang Z. et al. [11] have proposed a model named DQFSA, a Deep Q-learning-based architecture for malware detection. In contrast to conventional supervised learning techniques that depend on fixed, human-selected features, DQFSA chooses a small but very useful feature set on its own. As a result, the model performs better and requires less human involvement. DQFSA performs better than baseline techniques in experiments.

The work by Ahmed H., Hassan S. et al. [1] highlights the application of the Boruta algorithm for feature selection. The authors compare the Boruta wrapper and information gain filter approaches in order to handle the problem of high feature numbers.

For reliable network intrusion detection, the authors Barnard P., Marchetti N. and Da Silva L. [8] presented a two-stage pipeline using the SHapley Additive exPlanation (SHAP) framework to explain the model's actions. An Extreme Gradient Boosting (XGBoost) model for supervised intrusion detection is implemented in the first step. These SHAP-based explanations teach an auto-encoder to discriminate between known and unknown attacks in the second stage. The suggested solution correctly identifies novel threats that are encountered during testing, according to experiments conducted on the NSL-KDD dataset.

The main focus of the work by Jureckova O. et al. [20] is the online analysis of incoming harmful samples to group them into known malware families or categorize them into new malware families. From the EMBER dataset, features for seven popular malware families were recovered through the use of static analysis of portable executable files. The classification accuracy of multilayer perceptrons was 97.21% with a balanced accuracy of 95.33%.

Askari S. et al. [6] have presented an algorithm named Revised Fuzzy C-Means (RFCM), which tackles issues with noise, outliers, and varying cluster sizes that arise while clustering unlabeled data. RFCM successfully addresses these issues by adjusting limitations and using adaptive exponential functions. A comparative examination shows that RFCM outperforms previous methods in handling both uneven cluster sizes and noisy data. This method improves the accuracy of clustering and can be used in both engineering and physics.

The study carried out by Mohammadrezapour O. et al. [28] uses a genetic algorithm in association with fuzzy c-means (FCM) and K-means to cluster hydrochemical parameter data. K-means finds five ideal clusters, but FCM finds six. While comparing clustering results, FCM performs better than K-means since it takes uncertainty factors into account while generating class borders. Hydrogeological analysis and resource management are aided by the identification of regions with uniform groundwater quality with the help of this technical technique.

The effect of cluster size distribution on fuzzy c-means (FCM) and k-means clustering is investigated by Zhou K. et al. [41] in their study. It emphasizes that, especially for datasets with different cluster sizes, the widely accepted fuzzifier value of m = 2 in FCM might not be the best choice. Therefore, a smaller fuzzifier value is recommended while applying FCM clustering on datasets with unequal cluster sizes, and k-means clustering is preferred over FCM when considerable variability in cluster sizes is present.

With the objective to simplify implementation, Borlea et al. [9] have presented a Unified Form (UF) clustering algorithm, which combines K-Means (KM) and Fuzzy C-Means (FCM) into a single adjustable solution. Furthermore, it also suggests the Partitional Implementation of Unified Form (PIUF) algorithm, addressing the difficulties faced while sequentially processing big datasets, alongside maintaining scalability. By effectively managing large datasets and combining single-machine and distributed processing techniques, PIUF gets beyond hardware constraints.

Moustafa N. et al. [29] provides an extensive overview of Explainable Artificial Intelligence (XAI) approaches for intrusion detection in Internet of Things networks using anomaly-based methods. It looks at how XAI improves cybersecurity defenses by giving deep learning models interpretability and confidence, which are essential for locating attack surfaces and vectors. The study covers contemporary research at the convergence of XAI, anomaly-based IDS, and IoT, as well as machine learning and deep learning models and anomaly-based detection strategies

The incorporation of mobile devices in the Industrial Internet of Things (IIoT) enables the remote supervision and management of industrial operations. But because of this interconnection, there are serious cybersecurity threats, especially from malware assaults that can seriously impair operations. Amin M. et al. [4] suggest using statistical drift detection techniques to spot changes in data patterns and then building machine learning classifiers to quickly counter new malware threats to address this. With a 94% F1-score and 95.2% accuracy, the suggested method showed a high success rate, indicating its effectiveness and simplicity of use.

In order to efficiently evaluate unbalanced encrypted traffic streams, the research work by Niu et al. [32] presents a method based on Improved Adaptive Random Forests (IARF). Improved parameter updates and sensitivity to malware families with few samples in unbalanced datasets are demonstrated by IARF. Three cutting-edge techniques were compared and contrasted with the suggested approach, which was assessed using a mixed dataset from several sources. The approach is especially effective for identifying criminal activities.

Luo P. et al. [24] investigate the use of deep learning techniques in order to categorize raw encrypted traffic, which provides great accuracy but lacks interpretability. In order to tackle this issue, the research presents a new feature engineering approach known as "BITization" together with a sliding window method for determining crucial bytes concerning classification. This strategy improves accuracy over traditional machine learning techniques by at least 14.1%.

The study carried out by Gulatas et al. [13] examines 64 families of IoT malware discovered between 2008 and 2022 in order to meet the pressing need for practical solutions. Based on target architecture, devices, distribution strategies, attack vectors, persistence tactics, and their evolutionary history, it methodically describes these families. With the use of the "Cyber Kill Chain" and "Mitre ATT&CK for ICS" frameworks, the study offers a thorough understanding of the behaviors of IoT malware, which can help guide future research and security tactics.

Cloud systems are exposed to significant security concerns due to its inadequate kernel-sharing and system resource isolation methods. In this study by Wang Y. et al. [38], a real-time malware detection solution, called DockerWatch, was designed for cloud systems that use containers. DockerWatch uses behavior-based graphs and a collection of static features to detect harmful patterns when it pulls executable files from containers in a non-intrusive manner. A two-stage hybrid deep learning technique improves the performance and speed of detection. Comprehensive tests on real-world datasets show that DockerWatch offers better detection performance with less overhead.

In order to identify in-container malware, Xinfeng H. et al. [16] provide a framework called Malware Detection for Container Runtime based on Virtual Machine Introspection (MDCRV) in their study. This approach can automatically export the memory snapshots and rebuild container semantics from memory snapshots. The in-container malware and benign application activities are converted into grayscale images, and then the malware features are extracted from the self-constructed dataset using a convolutional neural network. Based on the testing results, MDCRV improves security and achieves excellent accuracy.

Dynamic analysis approaches are more effective than traditional signature-based techniques in the detection of zero-day malware in Windows PE files. In this work by Nguyen M.T., Nguyen V.H. and Nathan S. [31], a novel method based on graph attention networks on multi-edge directed heterogeneous graphs built from API calls is presented. This approach overcomes the drawbacks of existing graph representations, which frequently omit important parameter data that influences behavioral analysis. Experimental results reveal that this method outperforms other relevant techniques in terms of TPR and FPR scores.

The objective of Anastasia P. et al. [34] is to provide thorough control over the execution of questionable samples on test systems by presenting developments in dynamic malware analysis techniques. The technique suggested by them uses memory page access control to isolate application code from system code, making it possible to identify all system API calls as well as unusual control flow transfers. It uses a multiprocessor hierarchical fair scheduling method based on Windows Round Robin with Priorities, and it uses cloaking techniques to hide the existence of the Scheduling module and minimize performance implications.

In order to avoid the requirement for manual feature extraction, this research carried out by Malhotra V. et al. [25] suggests approaching malware classification as a graph classification problem utilizing function call graphs, which provide information on program functions and their inter-procedural calls. Different Graph Neural Network (GNN) architectures are trained to produce embeddings for classification by utilizing Local Degree Profile features. Even with longer training times, the study discovers that the best GNN models outperform earlier studies on the MalNet-Tiny Android malware dataset and prevent overfitting problems that are typical of non-GNN approaches.

The usefulness of several machine learning and deep learning methods, such as Support Vector Machines (SVM), Hidden Markov Models (HMM), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) networks, for malware detection and classification, is investigated in this study. The work of Mehta R. et al. [27] specifically looks into a hybrid architecture in which feature vectors from several classifiers are used to train HMMs on opcode sequences and produce hidden states.

Ref	Auther	Year	Contribution
[14]	Gulmez, Sibel and Kakisim, Arzu Gorgulu and Sogukpinar, Ibrahim	2024	Developed a CNN model for ransomware detection
[25]	Malhotra, Vrinda and Potika, Katerina and Stamp, Mark	2024	Malware classification as a graph classification problem approach utilizing function call graphs
[1]	Ahmed, Hala and Soliman, Hassan and Elmogy, Mohammed	2023	Utilization of Boruta FS algorithm to extract best feature set
[29]	Moustafa, Nour and Koroniotis, Nickolaos and Keshk, Marwa and Zomaya, Albert Y and Tari, Zahir	2023	XAI approaches for intrusion detection in IoT network
[8]	Barnard, Pieter and Marchetti, Nicola and DaSilva, Luiz A	2022	XGB based two- stage pipeline approach for intrusion detection
[9]	Borlea, Ioan-Daniel and Precup, Radu-Emil and Borlea, Alexandra-Bianca and Iercan, Daniel	2021	A unified form of clustering combining K-Means and FCM
[41]	Zhou, Kaile and Yang, Shanlin	2020	comparative study of effective cluster size distribution of K-Means and FCM

Table 2.1: Related Work Reference Summary

Chapter 3

Preliminaries

This section provides a comprehensive background study for the project titled **"Enhancing Malware Detection and Family Identification with Machine Learning Models and Explainable AI"**. This study aims to establish the context and rationale behind the selection of the Boruta Feature Selection method and various classification and clustering methods for effective detection and family identification of malware samples.

3.1 Feature Selection

Feature selection is an important stage in the machine learning pipeline since it helps discover the most effective features while eliminating irrelevant ones. This decreases the dimension of the data and boosts the model's performance. There are three way to feature selection: filter methods, wrapper methods, and embedded methods. Filter methods, such as the correlation coefficient, chi-squared, and ANOVA, rank features according to their importance to the target variable.[22] Wrapper approaches, such as recursive feature elimination (RFE) and sequential feature selection (SFS), evaluate features based on their contribution to the model's performance. Embedded approaches, such as autoencoders (AEs) and local sensitive hashing (LSH), learn to represent data points in a lower-dimensional space while retaining their key characteristics.

3.1.1 Boruta Feature Selection Algorithm

The Boruta algorithm is a feature selection technique that uses a random forest classifier to select the importance of features in a dataset. It is based on the concept of comparing the value of original features with the importance of randomly permuted features (shadow features).[5] The algorithm executes numerous rounds, where in each iteration it trains a random forest classifier and evaluates the value of each feature. Then it compares the value of each original characteristic against the maximum importance of the shadow features. If the value of an original feature exceeds the maximum importance of the shadow features, it is considered an important feature. The Boruta algorithm repeats this process for a number of iterations to increase the confidence in the selected features[3].

The steps of working principle and functionality of Boruta Feature Selection are as follows:

- The algorithm begins by generating a new feature matrix that includes both the original features and their shadow features. The new matrix has twice the number of features as the original matrix.
- The added attributes are randomly disrupted to remove their correlation with the response.
- Generally a Random Forest (RF) classifier with the updated feature matrix is trained as input. In some cases, the classifier can be any other ML algorithm like Extreme Gradient Boosting (XGB) or Decision Tree (DT) as well. The classifier gives an importance score to each feature depending on how accurately it predicts the response variable.
- The Z_Score for original features and shadow features are calculated as given:

$$Z = \frac{X - \mu}{\sigma}$$

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{n}}$$

- The maximum Z_score value, i.e., S_max is determined in the shadow features matrix, and used it as the screening index.
- Original features with a Z_score higher than S_max are considered "important" and reserved. Features having a Z_score less than S_max are regarded as "unimportant" and deleted from the feature collection.

The above process is repeated until all characteristics have been assigned some importance.

Selecting and keeping only pertinent features is how the Boruta feature selection algorithm improves model performance. In addition to lowering computational complexity and enhancing interpretability, it helps avoid overfitting. Boruta ensures robust selection, which improves the dependability and efficiency of models by comparing the relevance of characteristics with shadow features. A detailed flow diagram showing the working of Boruta FS as discussed above has been given below.

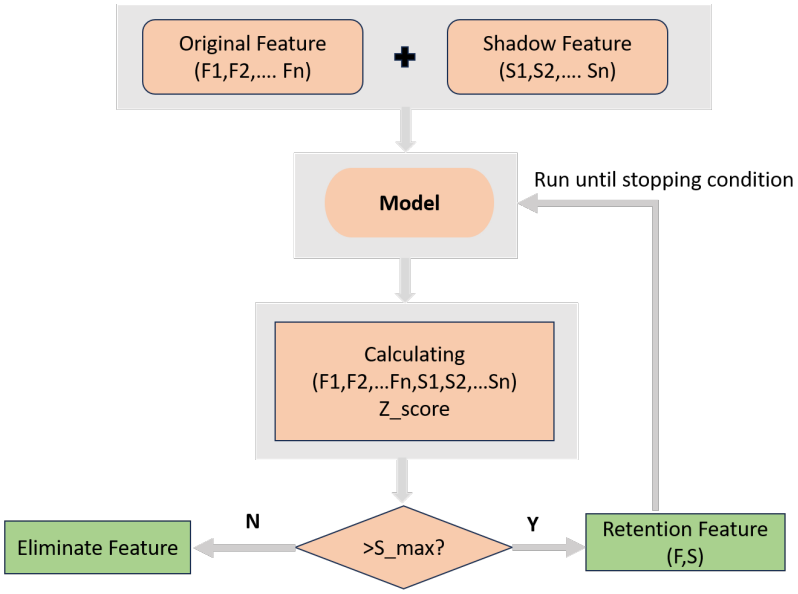


Figure 3.1: Flow Diagram of Boruta Algorithm

3.2 Supervised Learning & Classification Algorithms

Supervised learning is a form of machine learning where a model or algorithm is trained on labeled data, meaning the input comes with the correct output. Based on this training data, the model learns to map inputs to outputs, enabling it to forecast results for unknown data. In supervised learning, standard models for classification include Random Forest (RF), Decision Tree (DT), Support Vector Machine (SVM) and some other boosting-based algorithms such as Extreme Gradient Boosting algorithm (XGB), Light Gradient Boosting Algorithm (LGBM), etc. Our work focuses on leveraging the use of these ML algorithms both through feature selection and classification towards our goal of malware detection and family identification. Some of the models used in this study have been discussed in detail in the following subsections.

3.2.1 Extreme Gradient Boosting

XGBoost, short for extreme Gradient Boosting, is well-known in the field of machine learning for its robust skills in regression, classification, and ranking problems. Its precisely constructed distributed gradient boosting library exemplifies efficiency, versatility, and portability, making it an excellent alternative for a wide range of applications. XGB, an efficient implementation of the gradient boosting framework, focuses on speed and performance, distinguishing itself from previous algorithms[10].

At the foundation of XGB is its tree-based learning approach, a sequential process in which each tree is carefully built to correct the mistakes of the previous one. It adopts a regularized model throughout the training process to manage the model's complexity and prevent overfitting. The integration of L1 and L2 regularization terms is critical for lowering

model complexity while improving generalization performance, resulting in a well-balanced and accurate conclusion.

One of XGB's unique features is its creative use of "parallel processing" approaches. XGB uses parallelization to efficiently split the tree-building process across multiple CPUs, dramatically increasing training speeds and overall performance. This strategic strategy, together with the revolutionary "tree pruning" technique, work together to counteract over-fitting and improve the model's precision and reliability[35].

XGB's outstanding ability to manage missing numbers is one of its true strengths. XGBoost distinguishes itself from typical algorithms by autonomously navigating the challenge of missing values using a sophisticated method known as "missing value imputation" XGB effectively estimates and absorbs missing values by leveraging the knowledge encoded in other dataset features, hence increasing its resilience and adaptability in real-world applications.

Furthermore, XGB has a diverse set of hyperparameters that can be rigorously tuned to optimize the model's performance. Parameters such as the learning rate, maximum tree depth, number of trees, regularization parameters, and sub-sampling rate allow for extensive optimization. By precisely changing these hyperparameters, users may boost the model's accuracy, maximize performance, and avoid the pitfalls of overfitting, demonstrating the algorithm's versatility and precision in many machine-learning settings.

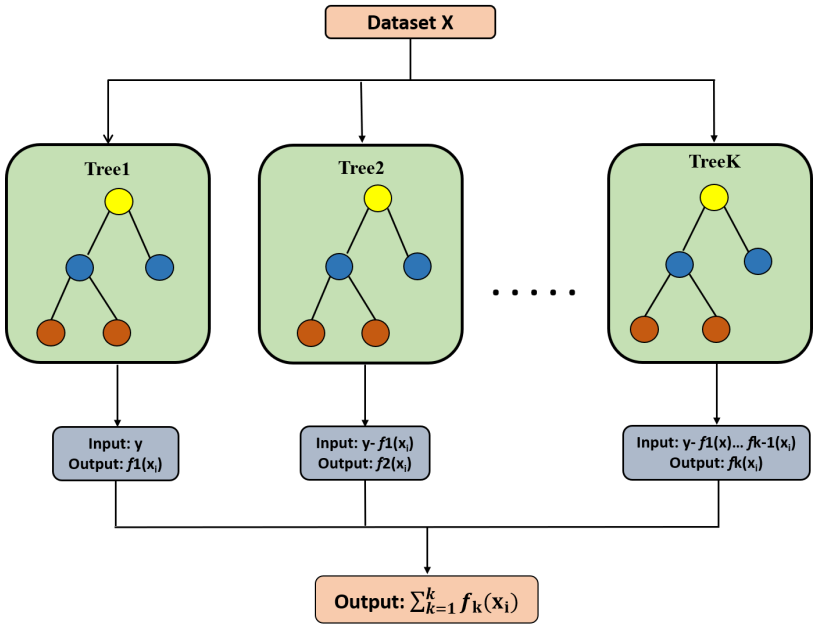


Figure 3.2: Flow Diagram of working of XGB Classifier

3.2.2 Random Forest Classifier

The RF classifier is a common machine-learning technique that can solve classification and regression problems. It is built on the concept of ensemble learning, which involves combining multiple methods to solve a complex problem and increase model performance [36].

The RF classifier builds a group of decision trees from a randomly selected subset of the training data. The final forecast is calculated by averaging the predictions from all decision trees. This strategy reduces overfitting and enhances model accuracy[7].

This algorithm's primary advantage is its capacity to manage a high number of characteristics and missing values. It can be used to select features by calculating and their contribution to the model's accuracy. It has various hyper-parameters that can be adjusted to improve model performance. These hyper-parameters include the number of decision trees, the tree's maximum depth, and the number of features to evaluate during splitting at each node. [26]

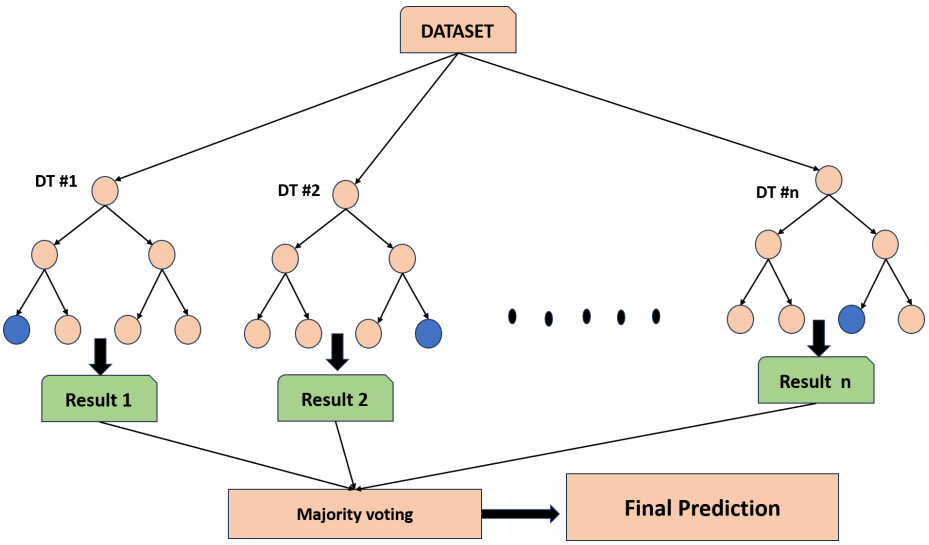


Figure 3.3: Functioning of Random Forest Algorithm

The key steps to implementing a Random Forest Algorithm are as follows:

- **Data preparation**- It is the first stage in training the model in machine learning. This includes loading the dataset, cleaning it, and putting it into a training-ready format. This could include encoding categorical variables, scaling numerical variables, and partitioning the dataset into training and testing sets.

- **Creating Decision Trees**- The next stage is to generate decision trees from a randomly selected portion of the training data. At each node of the tree, a subset of features is picked, and the optimal split is determined using a criterion such as Gini impurity or entropy.

- **Ensemble Learning**- The RF algorithm combines predictions from numerous decision trees to produce a single prediction. For regression issues, this is accomplished by averaging the predictions, whereas for classification problems, majority voting is used.

- **Hyperparameter Tuning**- The RF algorithm's performance can be improved by adjusting its hyperparameters. This contains the number of decision trees, the maximum depth of the tree, and the undefined number of features to evaluate when splitting at each node.

3.3 Unsupervised Learning & Clustering algorithms

Grouping a set of items so that they are more similar to each other than to those in other groups is known as clustering, and it is a fundamental strategy in unsupervised learning. Since this approach works with unlabeled data, it's perfect for identifying patterns in datasets without labels.[30] Popular clustering algorithms that offer distinct methods for creating and finding clusters depending on the underlying structure of the data are k-means, Fuzzy c-means, hierarchical clustering, and DBSCAN. To find underlying patterns and links, clustering is frequently employed in a variety of domains, such as market segmentation, bioinformatics, and image analysis.

Traditional clustering methods are widely used, but they have inherent problems that can reduce their effectiveness. Choosing the starting cluster centers, figuring out the ideal number of clusters, and being sensitive to outliers are some of these difficulties. For example, in k-means, the initial cluster centers can have a major impact on the final clustering result, frequently resulting in less-than-ideal solutions.[40] Similar to this, a lot of clustering algorithms demand a predetermined number of clusters, which the data may not always show. The representation of real data structures can be distorted by outliers, which can have a disproportionately large impact on the clustering process.

Fuzzy clustering permits data points to belong to numerous clusters with different degrees of membership, whereas, in the case of crisp clustering, each data point is allocated unambiguously to a single cluster. A well-known fuzzy clustering technique is fuzzy c-means (FCM), which expands on the k-means strategy by adding membership levels for every data item in every cluster.

More and more recent studies have concentrated on how data distribution affects the performance of clustering algorithms. The features of the data distribution, including skewness, kurtosis, noise, and outliers, are important factors in deciding how effective clustering is. For instance, k-means, which uses Euclidean distance, could not work well with data that has a lot of noise or non-globular cluster forms. Although they can be more computationally demanding for big data sets, hierarchical clustering techniques might be more resilient to

these fluctuations. Fuzziness-based FCM can potentially handle overlapping clusters more well, but it can also be more susceptible to initialization and noise.

There are still few quantitative and comparative analyses of various clustering techniques with diverse data distributions. The main causes of this discrepancy are the unsupervised clustering method's lack of ground truth and the academic community's focus on algorithms' capacity for generalization. Furthermore, no single clustering algorithm is always the best choice for every kind of data distribution.[37]

As a result, even while clustering is still a strong and adaptable tool for data analysis, its effectiveness depends largely on the features of the data set and the particular method used. More research is essential to create more resilient clustering techniques that can adjust to different data distributions and get around the built-in drawbacks of conventional algorithms. The accuracy and usefulness of these techniques in real-world situations will increase with an understanding of the subtle effects of data distribution on clustering performance.

An in-depth discussion regarding the working and functionality of the most widely used clustering algorithms, namely the K-means algorithm and the Fuzzy c-means algorithm is given below.

3.3.1 K-Means Algorithm

In the fields of data analysis and machine learning, the K-means algorithm is a simple and in-demand clustering method. The K-means was first proposed by MacQueen in 1967. His aim is to split a dataset into k clusters. Each of which is portrayed by its centroid and each has the nearest centroid which data point is linked with the cluster.[17].

Initially, At the beginning of the process, k initial centroids are selected randomly from the dataset. These centroids are necessary as they act as the beginning of the clustering process. The next step is to allocate each data point the nearest centroid which is known as Euclidean distance. This allocating process develops the initial clusters.

Once the first allocation is complete, the algorithm reevaluates each cluster's centroids. The new centroid is the mean of all the data points assigned to that cluster. Each data point is reallocated to the nearest centroid and new centroids are formed using the modified clusters. These processes are done repeatedly. This repeated process is continued until the centroids no longer noticeably change, indicating the confluence.

The prime objective of the K-means algorithm is to reduce the within-cluster sum of squares (WCSS) which is also known as the inertia. The WCSS measures the closeness of the clusters and it is defined as:

$$WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

where C_i indicates the set of points in cluster and *i* and μ_i is the centroid of cluster *i*. Reducing the WCSS confirms that the data points within each cluster are as near to each other as possible, Hence forming solid and distinct clusters.

K-means clustering has numerous limitations regardless of its widespread acceptance and simplicity. The major limitation is its sensitivity to the original selection of centroid. Poor selection of initial centroids may lead to not substandard clustering results and the algorithm is stuck in nearby minima. To address this issue, numerous initiation techniques have been developed, such as the K-means++ algorithm, which provides a smarter technique for selecting original centroids and offers better clustering results.

Another drawback of K-means is to predetermine the number of clusters *k*. If the preliminary knowledge of the data distribution is confined then It is difficult to select the correct value for *k*. To help in determining the appropriate number of clusters, techniques such as the Elbow Method, the Silhouette Method, and the Gap Statistic are frequently used.

With a temporal complexity of $\mathcal{O}(n \cdot k \cdot d \cdot t)$, where *n* is the number of data points, *k* is the number of clusters, *d* is the dimensionality of the data, and *t* is the number of iterations until convergence, K-means clustering is computationally efficient in reality. Because of its effectiveness and simplicity, it's a good option for large datasets when speed is important. K-means clustering is computationally efficient in reality, with time complexity of $\mathcal{O}(n \cdot k \cdot d \cdot t)$, where *n* is the number of data points, *k* is the number of clusters, *d* is the dimensionality of the data, and *t* is the number of iterations until convergence. Its effectiveness and simplicity make it a good option for large datasets when speed is needed.

In summary, the K-means algorithm is a popular and powerful clustering technique that successfully splits data into *k* clusters by repeatedly refining the centroid of the cluster. Regardless of its drawbacks, which include initialization sensitivity, the requirement to predetermine the number of clusters, and assumptions regarding cluster spaces, its simplicity, effectiveness, and popularity make it a fundamental tool in data analysis.

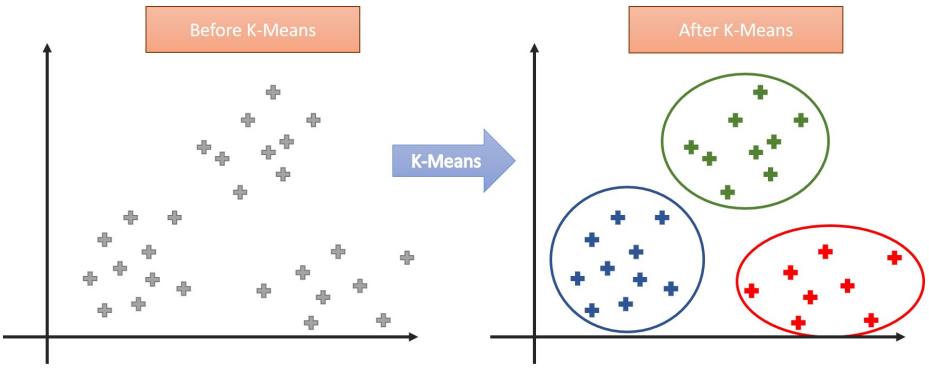


Figure 3.4: An overview of K-Means Clustering

3.3.2 Fuzzy C-Means Algorithm:

Proposed by Dunn in 1973 and then improved by Bezdek in 1981, the Fuzzy C-Means (FCM) algorithm is a well-known clustering method that is widely used in regional frequency analysis as well as other domains. Differentiating itself from the traditional K-means clustering method, fuzzy logic concepts are integrated into FCM, enabling every data point to be a part of numerous clusters with different levels of membership.[2] More flexibility and realism in data modeling are offered by this sophisticated approach, particularly when cluster boundaries are not clearly established.

Fundamentally,[7] FCM works on the tenet that every data point has a membership grade assigned to it by each cluster and that the total of these membership grades for a particular data point across all clusters is one. A membership function is used to formalize this, which is usually characterized by a level of fuzziness represented by the parameter m, where $m > 1$. The degree of cluster fuzziness is determined by the parameter m; larger values produce fuzzier clusters.

The FCM algorithm minimizes an objective function iteratively in order to optimize the clustering process. The objective function Jm for FCM is defined as:

$$J_m = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}^{m} \|x_i - c_j\|^2$$

Here, n is the number of data points, c is the number of clusters, u_{ij} represents the membership degree of data point x_i in cluster J, c_j denotes the center of cluster J $||x_i - c_j||$is the Euclidean distance between x_i and c_j. The algorithm seeks to minimize this objective function by iteratively updating the membership values u_{ij} and the cluster centers c_j.

The iterative process in FCM involves two main steps:

1. **Update Membership Values**: The membership value u_{ij} is updated based on the inverse distance of the data point x_i to the cluster center c_j. The formula for updating $u_{ij}$is:

$$u_{ij} = \left(\sum_{k=1}^{c} \left(\frac{x_i - c_k}{x_i - c_j} \right)^{\frac{2}{m-1}} \right)^{-1}$$

2. **Update Cluster Centers**: The cluster centers c_j are recalculated as the weighted average of all data points, where the weights are the membership degrees. The formula for updating c_j is:

$$c_j = \sum_{i=1}^{n} u_{ij} m \sum_{i=1}^{n} u_{ij} x_i$$

These steps are repeated until the change in the membership values and cluster centers between successive iterations falls below a predefined threshold, indicating convergence.

When compared to rigorous clustering techniques like K-means, the FCM algorithm has a number of advantages. By capturing the inherent ambiguity and overlaps across clusters, it offers a fuller representation of the data, which is especially useful in real-world circumstances where rigorous division is impractical. Furthermore, by using the fuzzy memberships to spot outliers and unclear data points, the clustering process can be made more robust.

FCM is not without its limits, though. Selecting the fuzziness parameter m can have a substantial effect on the clustering outcomes, and figuring out the ideal number of clusters C is still a difficult task. Furthermore, FCM can converge to local minima and is sensitive to the initial cluster center selection, just as K-means. For this reason, it is necessary to do many runs or employ sophisticated initialization approaches to guarantee reliable clustering results.

In summary, the Fuzzy C-Means algorithm is an advanced clustering methodology that builds upon the classic K-means approach by integrating fuzzy logic. Its versatility in handling ambiguous data and offering adaptable clustering solutions makes it an invaluable instrument for a wide range of applications, such as picture segmentation, pattern recognition, and regional frequency analysis. The algorithm's abilities to capture subtleties in data, in spite of its computational complexity and sensitivity to initialization, highlight its ongoing relevance and usefulness in data analysis.

3.4 Explainable Artificial Intelligence (XAI)

Explainable Artificial Intelligence (XAI) is an important subfield of artificial intelligence (AI) research and applications. Generating AI models whose judgments and forecasts are comprehensible and interpretable to people is XAI's main objective. In addition to producing precise forecasts, this entails offering an understanding of the methodology behind the forecasts. For AI systems to be transparent, trustworthy, and accountable, they must have this explanatory capacity.

3.4.1 Importance of Explainable AI

Clarifying the inner workings of AI models is one of the main goals of XAI. Stakeholders can learn more about a model's behavior by comprehending how it interprets inputs to produce outputs. This knowledge is essential for confirming the robustness and dependability of the model, especially in intricate and high-stakes industries like banking and healthcare.

3.4.2 Types of Explainable AI

- **Model-Based Approaches:** The creation of models with inherent interpretability is the goal of model-based XAI. Such models as decision trees and linear regression models, which naturally lend themselves to simple interpretation, are created with transparency and simplicity in mind.
- **Post Hoc Approaches:** After a model has been trained to understand its predictions, post hoc methods are used. They fall into two categories:
    1. **Black Box**: Methods that allow one to understand the predictions of intricate, opaque models—like deep learning networks—without changing the model directly. Two such are LIME (Local Interpretable Model-agnostic Explanations) and SHAP (Shapley Additive Explanations).
    2. **White Box**: Techniques that apply directly to simpler, more comprehensible models by looking at and comprehending the internal components of the model.

Additionally, XAI is essential for enhancing so-called "black box" models, which are frequently intricate and opaque. Developers can gain insights into these models through the application of interpretability techniques, which can result in improvements and tweaks that boost dependability and performance. Iteratively transforming black box models into more comprehensible and reliable systems is made possible by this method.

3.4.3 XAI by Scope

- **Global Scope** Comprehending the model's behavior throughout the complete dataset is necessary for global interpretability. Through the identification of broad patterns and relationships in the data, this method aims to give a thorough overview of the model's decision-making process.
- **Local Scope** Local interpretability explains why a model chooses a particular course of action in a given situation by concentrating on individual forecasts. When doing a case-by-case analysis, such as determining the reason behind a loan application's approval or denial, this is especially helpful.

3.4.4 XAI by Model

- **Model Specific** Interpretability techniques that are specific to a given model type are called model-specific procedures. To offer explanations, these techniques take advantage of the special qualities and configurations of the model. Convolutional neural networks, on the other hand, can be directly interpreted by employing methods such as activation mapping, whilst decision trees can be immediately read by looking at the tree structure.
- **Model Agnostic** Any kind of model can be used with model-agnostic techniques, regardless of the internal organization of the model. These methods offer a broad and adaptable approach to interpretability, which enables their application to a variety of models and use situations.

To sum up, Explainable AI is a broad field that improves the accountability, transparency, and reliability of AI systems. Through the use of a variety of interpretability techniques—both model-based and post hoc—and taking into account the breadth and depth of explanations, XAI offers crucial insights into AI models, guaranteeing their fairness, comprehensibility, and compliance with legal requirements.

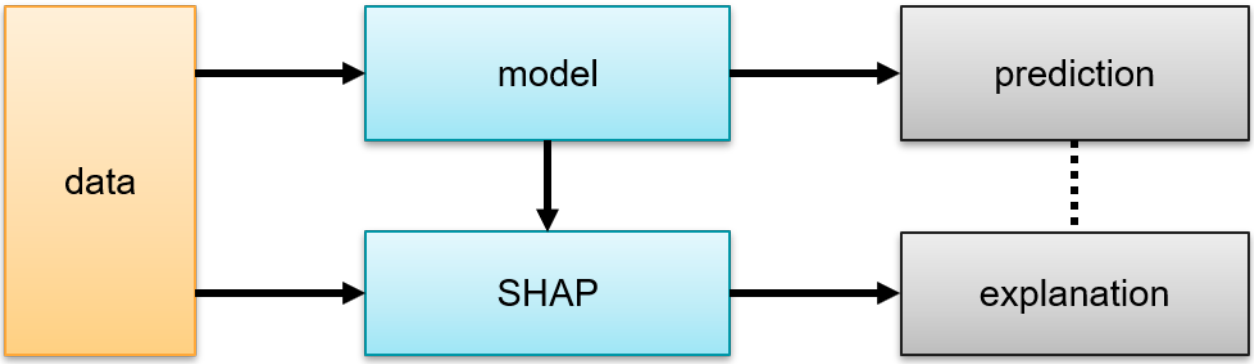


Figure 3.5: An overview of XAI

Chapter 4

Malware Detection and Classification using ML

This section discusses the study design, data-gathering strategies, and analytical approaches used. To guarantee the study's reproducibility, it describes the methods and resources utilized to collect and process data.

4.1 Proposed Architecure

The flow diagram of the proposed architecture and its detailed discussion of our study is given below:

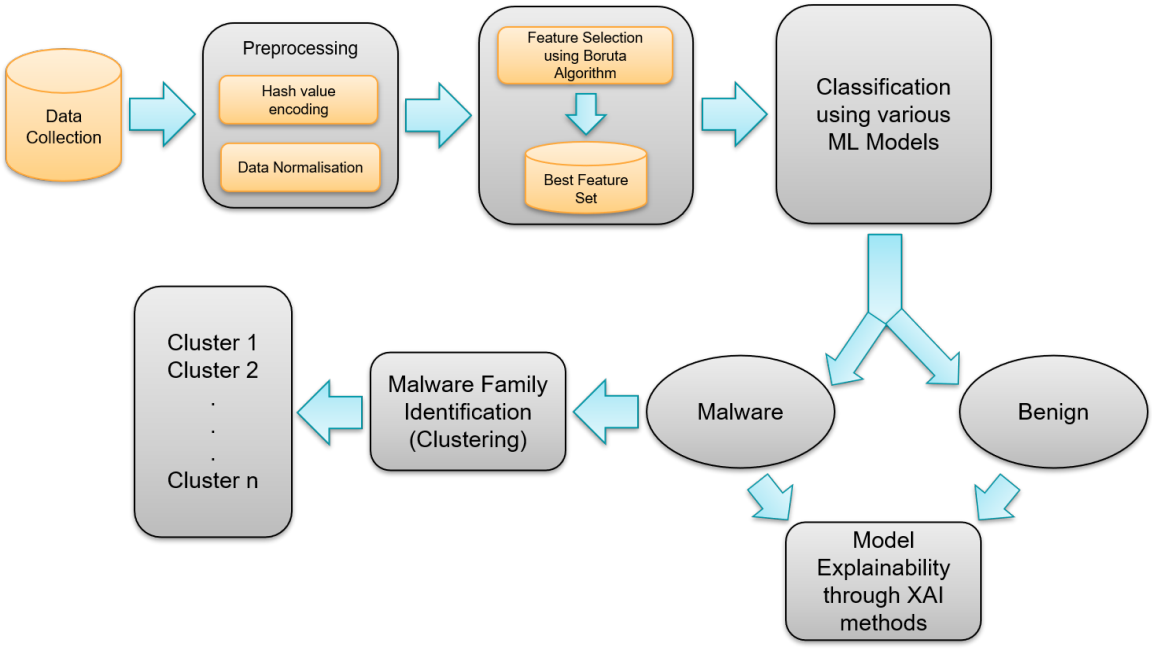


Figure 4.1: Proposed Architecture

- **Data Collection and Preprocessing:** Our dataset consists of 201549 samples including both benign and malicious PE files. The hash value columns are encoded and converted into equivalent integer values taken into a manageable working range. These were divided into a specific number of columns for a given type of hash function.

- **Feature selection using Boruta FS:** The Boruta algorithm is used for feature selection following preprocessing. The most important features are determined using the robust feature selection technique called Boruta. This process yields a "best feature set", which improves interpretability and improves model performance by removing unnecessary and duplicated features.

- **Classification using ML Models:** The selected features are then fed into various machine learning models like RF, XGB and DT for classification. Every model is trained to distinguish between categories of benign data and malware from the input data.

- **Model Explainability through XAI Methods:** Our architecture's incorporation of Explainable AI (XAI) techniques such as SHAP is a key element. XAI techniques assist in explaining the traits that led to the classification of a given sample as benign or malicious, as well as the reasons behind the choice. This improves the ML models' interpretability and reliability.

- **Malware Family Identification and Clustering:** For samples identified as malware, clustering algorithms were utilized to identify the specific malware family. These combine malware samples that are similar in terms of their attributes. Multiple clusters (Cluster 1, Cluster 2,..., Cluster n) are produced as a result of the clustering, each of which represents a different malware family. Any malicious sample that won't fall under any specific kind of family would be treated as a zero-day attack.

The following sections are designed to discuss in detail the methodology and the results obtained at every step of our study.

4.2 Data Collection

The dataset that was employed in this investigation is a set of samples that were gathered from multiple sources by Mr. Michael Lester and comprise a variety of malware and benignware specimens[21]. **In total, we have 201549 Portable Executable samples (eg: .exe, .dll, .scr). It comprises of 86812 legitimate files or benignware and 114737 malicious files or malware.** This dataset aims to help security and AI researchers enhance cybersecurity in the business by making raw tagged portable executables available to them.

Each sample is a Windows PE file and is defined by a number of attributes, such as the hash values of MD5, SHA-1, and SHA-256, which function as unique identifiers for the

corresponding malware instances. Other attributes include metadata, file type, filing date, user ID, length of the hash value and Shannon entropy of the file. An overview of the dataset is produced below.

Field	Description	Example
id	The identifier for the sample that corresponds to the name of the file in the samples directory	5
md5	The MD5 hash of the file.	ad27f1a72dda61d1659810c406f37ab8
sha1	The SHA1 hash of the file.	f8fd630c880257c7e74c1f87929993477453d989
sha256	The SHA256 of the file.	984d732c9f32197232918f2fce0aa9cedc1011d93 e32acb4ad01e13f2f76d599
total	The total number of antivirus engines that scan this file at the time of the query.	67
positive	The number of antivirus engines that flag this files malicious at the time of the query.	0
list	Either blacklist or whitelist indicating whether or not the file is malicious or legitimate respectively.	Whitelist
filetype	his field will always be exe for this data set.	exe
submitted	The date that the sample was entered into my database	6/24/2018 4:18:38 PM
user_id	Redacted.	1
length	The length of the file in bytes.	211,456
entropy	The Shannon entropy of the file. The values will range from 0 to 8.	2.231824

Table 4.1: Dataset Description

MD5 (Message Digest Algorithm 5): A popular cryptographic hash algorithm called MD5 generates a 128-bit (16-byte) hash value, which is commonly represented as a 32-character hexadecimal integer. Ronald Rivest created the design in 1991. Although MD5 is widely used for digital signatures and checksums, its susceptibility to collision attacks has caused it to be phased out in favor of more secure hash functions. Due to flaws found over time, MD5, despite being widely used in the past, is no longer regarded as secure for cryptographic applications. In particular, it has been shown that MD5 is incompatible with security-sensitive applications due to collision attacks, which occur when two distinct inputs provide the same hash output.

SHA-1 (Secure Hash Algorithm 1): Another cryptographic hash function, called SHA-1, was created by the National Security Agency (NSA) and released in 1993 by the National Institute of Standards and Technology (NIST). It generates a hash value of 160 bits (20 bytes), which is commonly expressed as a 40-character hexadecimal integer. Many security applications and protocols, such as digital signatures, Git repositories, and SSL/TLS cer-

tificates, made extensive use of SHA-1. It has been discovered that SHA-1 is susceptible to collision attacks, much as MD5. Researchers were able to effectively show a collision attack against SHA-1 in 2017, which led to a general shift away from the use of SHA-1 in favor of more secure alternatives.

SHA-256 (Secure Hash Algorithm 256): SHA-256 belongs to the family of hash algorithms known as SHA-2 (Secure Hash Algorithm 2), which also includes SHA-224, SHA-256, SHA-384, and SHA-512. These hash functions have varying output sizes. As its name implies, SHA-256 generates a 256-bit (32-byte) hash value, which is commonly expressed as a 64-character hexadecimal integer. When it comes to resistance to collision attacks, SHA-256 is considered a secure cryptographic hash function because it is far more resilient than MD5 and SHA-1. It is frequently utilized in many security applications, such as hashing algorithms for passwords, blockchain technology, digital signatures, and SSL/TLS certificates.

To sum up, although MD5 and SHA-1 were formerly extensively employed in cryptography, their susceptibility to collision attacks has made them inappropriate for contemporary security demands. However, SHA-256 is still a commonly used, safe cryptographic hash function that provides a strong defense against a range of cryptographic assaults.

4.3 Data Preprocessing

Data preprocessing refers to the process of cleaning, converting, and organizing raw data into usable format, and it is an essential stage in machine learning projects. Assuring the model can learn efficiently and generate precise predictions involves handling missing values, normalizing data, encoding categorical variables, and splitting datasets into training, validation and testing sets. Proper preprocessing boosts model performance. This step of our study focuses on data normalization and efficient handling of the hash values for usage in further model building and interpretability.

4.3.1 Hash Encoding

Hash encoding is an essential method for deriving useful features from unprocessed hash values and making ML tasks easier. As part of our method, we use a unique hash encoding methodology to convert the hash values into a structured format that can be effectively used for further computation.

A custom hash encoding function has been devised for our study, to map each hash value to a list of integers of fixed length within a manageable range. This process involves several key steps which are listed below in detail.

- **Hexadecimal Conversion**: Initially, the hash value is converted from its string representation to a hexadecimal format.

- **Binary Conversion**: The hexadecimal representation is then transformed into a binary string to facilitate subsequent partitioning.

- **Partitioning**: The binary string is divided into multiple blocks, each representing a segment of the original hash value. The division is performed based on the specified number of parts.

- **Integer Conversion**: Each binary segment is converted into an integer, thus generating a list of integers corresponding to the hash value.

- **Modulus Operation**: To ensure numerical stability and mitigate computational complexities, the resulting integer values are subjected to a modulus operation using a prime modulus.

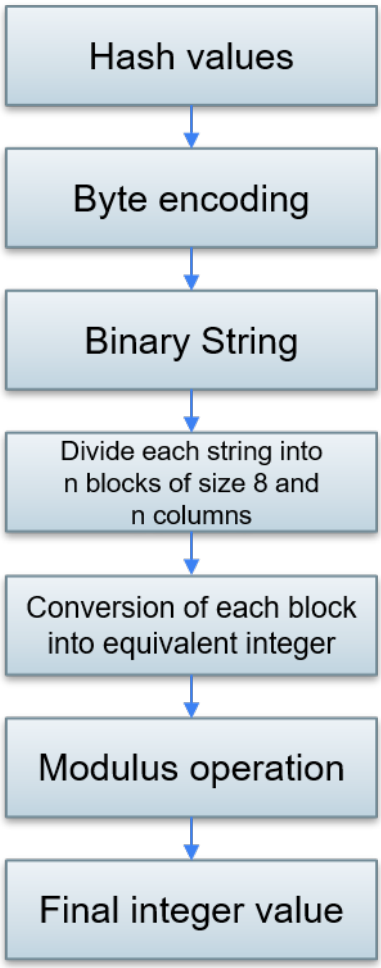


Figure 4.2: Flow diagram of hash encoding

Taking a hash value of 'N' bits for instance, say ad27. . . .7ab8.

- According to our devised method, the hash value is first converted into its equivalent binary string. The resultant 'N' bit binary string is divided by 32 to get 'n' blocks.

  Hence, no. of blocks, n = N/32.

- Now, strings in each of the blocks are converted to an equivalent integer.

- At last modulus operation is performed on the integer values to get the values in a manageable range to work with. In this regard, we have chosen 1009, which happens to be the smallest four-digit prime number. This ensures that every value comes within the range of 0 - 1008.

  Hence, the value m = n%1009 for each integer value.

- Finally, for each particular type of hash value, 'n' columns with the respective integer values in them are added to the dataset, and the original hash columns are omitted.

Applying the above-discussed hash-encoding technique, we have the following augmented columns in our dataset.

- **For MD5 hashes:** Each hash value is 128 bits long. So the number of blocks/columns = 128/32 = 4.
  The columns are: md5_int1, md5_int2, md5_int3, md5_int4.

- **For SHA-1 hashes:** Each hash value is 160 bits long. So the number of blocks/columns = 160/32 = 5.
  The columns are: sha1_int1, sha1_int2, sha1_int3, sha1_int4, sha1_int5.

- **For SHA-256 hashes:** Each hash value is 256 bits long. So the number of blocks/columns = 256/32 = 8.
  The columns are: sha256_int1, sha256_int2, sha256_int3, sha256_int4, sha256_int5, sha256_int6, sha256_int7, sha256_int8

4.3.2 Data Normalisation

A crucial preprocessing step in machine learning is data normalization or standardization, which guarantees that features are on the same scale and improves model performance. Standardization modifies data to have a mean of zero and a standard deviation of one, whereas normalization usually scales the data to a [0, 1] range. By reducing the bias

caused by different feature scales, this procedure speeds up the convergence of algorithms such as gradient descent and enhances the precision and effectiveness of the models. The MinMaxScaler method for the process of scaling was applied to our dataset to carry out the remaining work.

4.4 Feature Selection

Choosing the right features to include in machine learning models is essential for detecting malware and predicting family dynamics. In order to ensure model efficiency, interpretability, and generalization capability, it is crucial to choose the most pertinent and informative characteristics from the wide range of potential features that may be collected from malware samples. Through the reduction of dimensionality and the removal of redundant or irrelevant features, feature selection approaches seek to uncover a subset of characteristics that not only considerably improve the prediction performance of the model but also improve its interpretability. Typical techniques for feature selection are wrapper methods,[**b1**] which use the learning algorithm's predictive performance to choose features, embedded methods, which integrate feature selection straight into the learning process, and filter methods, which assess features apart from the learning algorithm. In this work, feature selection has been performed by using the Boruta feature selection algorithm, which is discussed in length below.

The Random Forest classifier (or a similar ensemble approach like XGBoost) is the foundation of the Boruta wrapper method. It functions by weighing the relative relevance of real features vs shadow features, which are produced by randomly varying the original feature values. The main point is that features that are genuinely significant are given more weight than arbitrary (shadow) features.

Our study uses the XGB classifier as the classification algorithm by Boruta. Normally RF can be used but our reasons for choosing XGB over RF are listed below.

- XGBoost is used with Boruta due to its powerful feature importance metrics derived from gradient boosting.
- It focuses on difficult-to-predict cases and constructs trees sequentially, which tends to yield more nuanced significance estimates.
- Boruta can capture intricate, nonlinear correlations between features and the target variable by utilizing XGBoost.

Among all the features of our dataset, the Boruta FS selected the three best columns for further investigation. These columns are **length, entropy and sha1_int3.**

4.5 Classification

The dataset at our perusal is labeled, i.e., a given file is either malicious or benign ware. But in case a file is malware, it is not known about the family of malware it belongs to. In this situation, it is essential to classify our data using machine learning models. Prior to that, we split the dataset into training and testing sets in the ratio of 75:25 and applied k-fold cross-validation on our dataset.

K-fold cross-validation is a reliable method for assessing how well machine learning models work. To do this, the dataset was divided into k equal subsets, or "folds." Each fold is used as the test set once, and the model is trained on 'k-1' folds before being tested on the remaining fold. This method is done k times. By utilizing every observation for both training and validation, this approach guarantees an all-encompassing evaluation of the model's efficacy. By reducing overfitting and providing a more accurate estimate of model generalization to unknown data, it raises the validity of experimental findings in scholarly studies. In our case, the dataset was 10-fold cross-validated and analyzed accordingly.

We have leveraged the utility of some standard ML models for the purpose of classification. Algorithms such as RF, XGB and DT produced quite impressive results. The classification was performed on both the hash-encoded dataset and also on the dataset containing the best features which were determined by Boruta FS. The results have been discussed in the subsequent chapter.

Furthermore, after a file is found to be malware, it is important to identify the family of malware it belongs to. Since our dataset is not labeled with respect to the type of malware, it became important to apply clustering to determine their respective families. In our study, the clustering methods such as K-Means and Fuzzy C-Means were utilized and evaluated with the relevant metrics.

4.6 Usage of XAI methods

In our study, improving the transparency and interpretability of the models requires the use of Explainable AI (XAI) techniques, specifically SHapley Additive exPlanations (SHAP). By quantifying each feature's contribution to the model's predictions, SHAP values provide a clear understanding of how machine learning models used for malware detection make decisions. Following sample classification with RF, XGB, and DT models, feature impacts are examined by computing SHAP values. Enhancing interpretability and trust, this method clarifies the reasons behind a model's classification of a file as benign or malignant. For instance, SHAP indicates when a hash value has a significant impact on a prediction, which helps cybersecurity professionals comprehend and validate model behavior. Furthermore, SHAP facilitates targeted model changes by identifying factors that contribute to inaccurate predictions, which helps with model debugging. During the clustering stage, SHAP clarifies the distinctive features of every malware family cluster, augmenting our comprehension of malware clusters. Overall, the efficacy of malware detection and classification is greatly

increased by XAI techniques through SHAP, which offers insightful, transparent information that improves the dependability and understandability of our machine-learning models for security analysts.

4.7 Clustering for Malware Family Identification

An important part of data mining and statistical machine learning is clustering, which is the unsupervised process of grouping data objects without any prior knowledge based on similarities or differences. It is an indispensable instrument for revealing the underlying structures in data, opening up a wide range of applications in domains like pattern recognition, image processing, anomaly detection, and information retrieval. Factors like dimensionality, size, noise, outliers, and attribute types all have a significant impact on how effective clustering methods are. Thus, it is crucial to comprehend how data distributions affect the results of clustering.

The primary objective of this phase was to identify specific malware families among the classified malware files. This was accomplished by using clustering techniques, namely K-means and Fuzzy C-means clustering. These unsupervised learning methods were chosen because of their ability to find inherent patterns in the data without the requirement for previous labeling. K-Means clustering was utilized to separate the malware files into multiple groups according to feature similarities. By reducing the variance within each cluster, this method clusters similar files together. On the other hand, fuzzy C-means clustering produced a clustering technique that was more flexible. Unlike K-Means, fuzzy C-Means provide files with membership levels for several clusters, indicating the degree to which each file is a part of a certain cluster.

Chapter 5

Results and Discussion

This section consists of two subsections: firstly the evaluation metrics used and the actual results. elaborates the various results of our study for our malware detection and prediction, and they are quite notable. The Random Forest model outperformed the other machine learning models that were assessed, achieving the highest accuracy, precision, recall, and F1 score. Notable trends surfaced, like the association between specific characteristics and malware categories. These outcomes highlight the usefulness of our strategy, providing improved detection powers over current techniques. This paves the way for an in-depth discussion of the data characteristics and a thorough comparative study of model performances. Along with that, the clustering methods utilized provide actual insights about the optimal number of clusters possible on our dataset.

5.1 Evaluation Metrics

This subsection provides an in-depth discussion about the various performance metrics used to classify and cluster and also explains the performance of our model.

5.1.1 Evaluation Metrics for Classification

- **Confusion Matrix:** An essential tool for assessing how well classification models work is the confusion matrix. A model's predictive power is can be evaluated and represented by the tabular representation of predicted class labels compared to the actual class labels that it displays. **True positives** (correctly predicted positive instances), **true negatives** (correctly predicted negative instances), **false positives** (incorrectly predicted as positive when they are negative), and **false negatives** (incorrectly predicted as negative when they are positive) are the four quadrants that make up a typical matrix. The model's strengths and shortcomings can be determined by looking at the data within these quadrants and calculating performance metrics like accuracy, precision, recall, and F1-score. These metrics are discussed next.

- **Accuracy:** The percentage of correctly identified occurrences with respect to all instances in the dataset is measured by accuracy. It offers a comprehensive evaluation of a model's performance by taking into account both accurate and inaccurate predictions. Although accuracy is helpful for datasets that are balanced, it can be deceptive when there is a class imbalance, hence additional metrics are required for a thorough assessment.

- **Standard Deviation:** The standard deviation calculates the variability or dispersion of data points from the mean. It shows how dispersed a dataset's values are, indicating how far they deviate from the mean.

- **ROC (Receiver Operating Characteristic) Curve:** A graphical representation called the ROC curve is used to assess how well binary classification models work. It displays the false positive rate (1-specificity) versus the real positive rate (sensitivity) at different threshold values. The model's ability to discriminate across various thresholds is revealed by the ROC curve, which aids in visualizing the trade-offs between sensitivity and specificity.

- **AUC (Area Under the ROC Curve):** The area under the ROC curve, or AUC, calculates a binary classification model's overall performance. Higher AUC values indicate better discrimination between positive and negative classes. It delivers a single scalar number. Because AUC provides a thorough assessment of model performance by taking into account all potential classification thresholds, it is commonly employed.

- **Precision:** The precision of a classification model is determined by how well it predicts positive outcomes. Out of all positive predictions, it represents the percentage of real positive predictions. High precision denotes a low false positive rate in the model, which makes it especially valuable in applications like fraud detection and medical diagnostics where false positives can have a significant financial impact.

- **Recall (Sensitivity):** Recall, sometimes referred to as sensitivity, indicates how well a model can locate all pertinent instances in a given dataset. This is the percentage of real positive cases that are true positive predictions.

- **F1 Score:** The F1 score offers a fair assessment of a model's performance since it is the harmonic mean of precision and recall. It provides a single score that strikes a compromise between recall and precision by accounting for both false positives and false negatives. When needed to establish the best possible balance between precision and recall in a model's predictions, the F1 score comes in handy.

- **Cohen-Kappa Score:** Cohen's Kappa score accounts for the agreement that would be expected by chance when evaluating the agreement between projected and actual classifications. It offers a reliable indicator of inter-rater reliability for data that is categorical. In disciplines like psychology, medicine, and the social sciences where

categorical judgments are frequently made, a high Cohen's Kappa value implies substantial agreement that goes beyond chance. For this reason, it is a useful indicator.

The descriptions of all the classification metrics stated above have been tabulated below.

Metrics	**Description**	**Formula**
Accuracy	Relation between the Sum of TP and TN divided by the total sum of the population.	Acc = TP+TN/TP+TN+FP+FN
Std.Deviation	Measures the dispersion of data point from the mean,calculated as the square root of the average of the squad differences between each data point and the mean.	$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Xi - \mu)^2}$
ROC	Represent the TPR against the FPR	precision=TP/TP+FN, FPR = FP/FP+TN
AUC	Measure the area under curve, indicating the model's ability to distinguish between classes.	AUC = $\int 0^1 TPR(t)d(FPR(t))$
Precision	Ratio of TP to the sum of TP and FP.	precision=TP/TP+FP
Recall	Ratio of the TP.	REC=TP/TP+FN
F1 Score	Harmonic mean of precision and recall.	F1= precision* Recall/precision+Recall
Cohen-Kappa Score	Measures inter-rate agreement for categorical items, adjusting for the possibility of chance agreement.	*K* = P_0 - P_e/*1* - P_e

Table 5.1: Standard Classification metrics

5.1.2 Evaluation metrics for Clustering

Silhouette Score for Clustering Evaluation: The silhouette score is a vital metric used for evaluating the quality of clustering in a dataset. It provides an indication of how well each data point fits within its assigned cluster and whether it might be better suited to another cluster. In this study, we utilized the silhouette score to determine the optimal number of clusters and to assess the effectiveness of the K-means clustering algorithm in identifying homogeneous regions within the data.

Definition and Calculation The silhouette width for the *i*th data point in cluster *k* is calculated using the formula:

$$SW_i = \frac{O_i - I_i}{\max\{I_i, O_i\}}$$

where:

- l_i is the mean distance between the *i*th data point and all other data points within the same cluster*k*.
- O_i is the minimum mean distance between the *i*th data point and all data points in the nearest neighboring cluster (the cluster that is not *k*)

The silhouette width ranges between -1 and +1:

- A value close to +1 indicates that the data point is well-matched to its own cluster and poorly matched to neighboring clusters.
- A value close to 0 suggests that the data point is on or near the boundary between two neighboring clusters.
- A value close to -1 indicates that the data point might be more appropriately assigned to a different cluster.

Elbow Analysis Method: A heuristic for figuring out the ideal number of clusters in a dataset is the elbow approach, which is applied in cluster analysis. To calculate the sum of squared errors (SSE) for each k, a clustering technique, such as k-means, must be run for a range of cluster numbers (k). When k is plotted against the SSE values, the "elbow point," or point at which the rate of SSE declines significantly, is shown to be the ideal number of clusters. This point offers a useful way to choose a suitable k by balancing the number of clusters and cluster compactness.

5.2 Results

The following subsection provides a thorough discussion of the results obtained by implementing the earlier stated classification algorithms evaluated through the performance metrics. Alongside, the clustering results and the explanation of the model performances are also discussed in detail.

5.2.1 Classification Results

Initially, the ML models were implemented on our encoded dataset consisting of all the original features. XGB classifier produced the best results with an accuracy of 82.55% and took a time of slightly more than 19 seconds to execute. The RF model performed well too, with an accuracy of 80.78% but taking significantly more time to execute.

After applying feature selection through the Boruta FS method, our dataset was reduced to three features on which the above-said models were again implemented. In this case, the RF model outperformed the others while classifying malware and benignware and achieved an accuracy of 84.46% with taking 191 seconds to execute. The performance of the XGB model was close enough at 84.01% accuracy and took much less time compared to RF. In this case, as well, DT produced satisfactory results, with 83.52% accuracy.

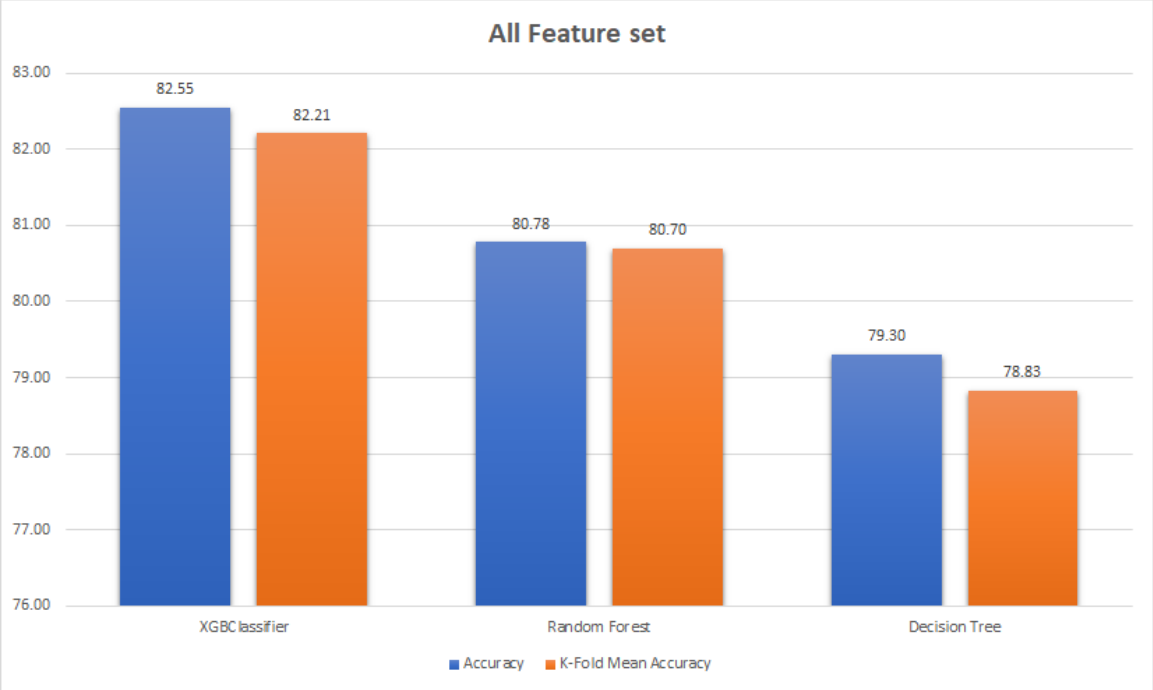
A detailed comparison of the results of the model performances, before and after applying Boruta FS, along with the ROC-AUC curve are tabulated as follows.

Model	**Accuracy**	**K-Fold mean accuracy**	**Std. Dev.**	**ROC-AUC**	**Precision**	**Recall**	**F1 score**	**CK score**	**Execution Time (in sec)**
RF	84.46	84.28	0.25	0.84	0.79	0.86	0.82	0.68	191.73
XGB	84.01	83.69	0.18	0.84	0.78	0.87	0.82	0.67	8.78
DT	83.53	82.87	0.16	0.83	0.80	0.80	0.80	0.66	7.55

Table 5.2: Classification results with best features

Model	**Accuracy**	**K-Fold mean accuracy**	**Std. Dev.**	**ROC-AUC**	**Precision**	**Recall**	**F1 score**	**CK score**	**Execution Time (in sec)**
RF	80.78	80.69	0.13	0.81	0.74	0.84	0.79	0.61	581.14
XGB	82.55	82.21	0.26	0.83	0.76	0.80	0.65	0.65	19.11
DT	79.30	78.83	0.22	0.78	0.76	0.75	0.57	0.57	49.26

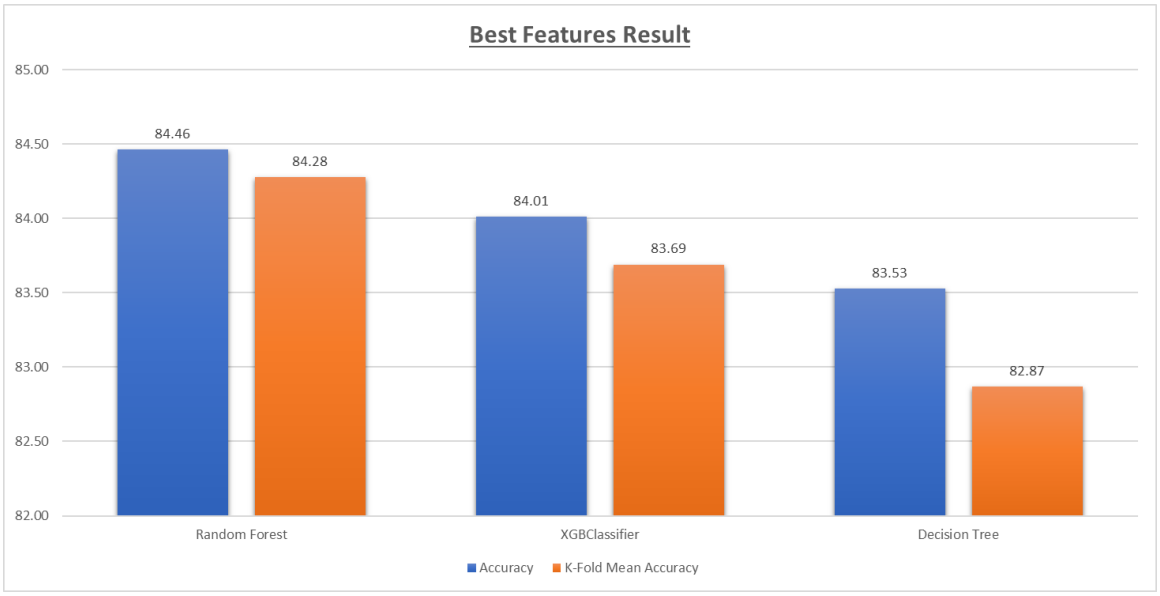
Table 5.3: Classification results with all features



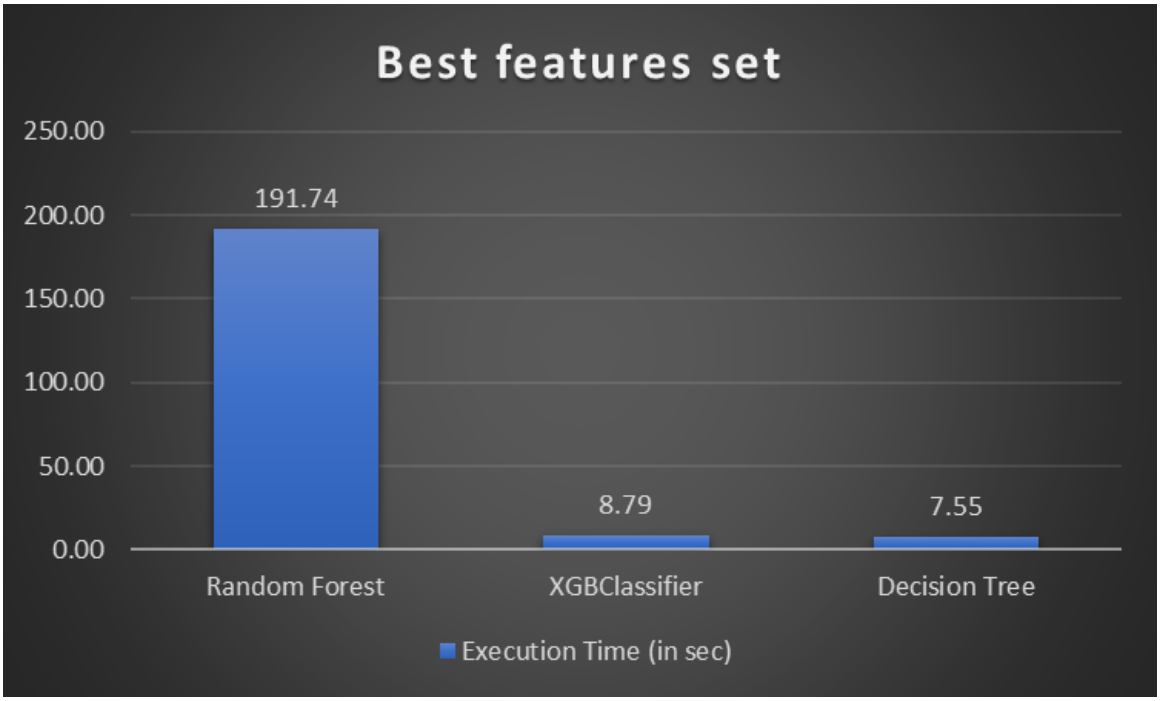
(a) Accuracy and k-fold mean accuracy

(b) Execution Time (in sec)

Figure 5.1:
Results with all features set



(a) Accuracy and k-fold mean accuracy



(b) Execution Time (in sec)

Figure 5.2:
Results with best features set

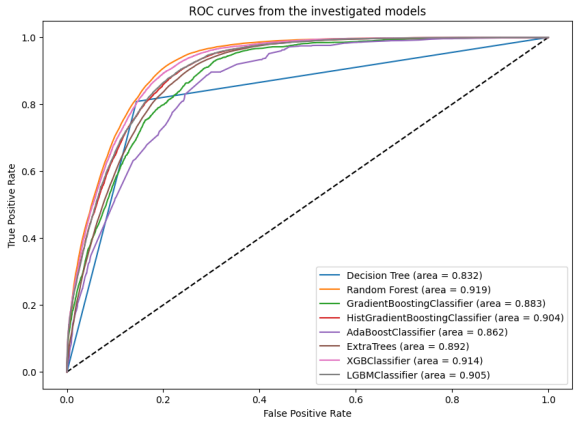
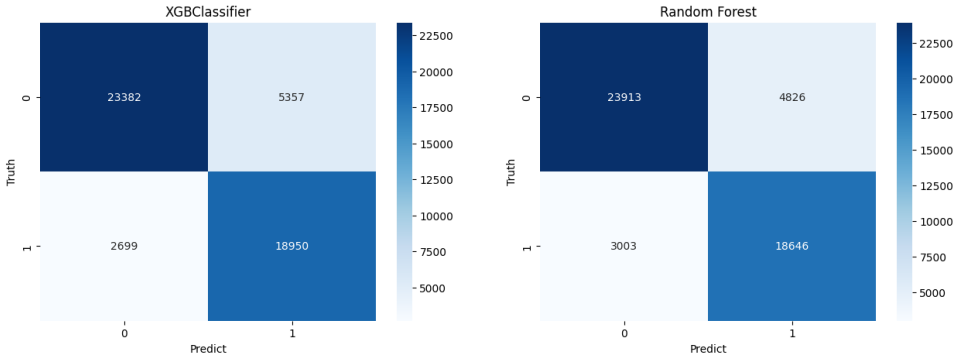


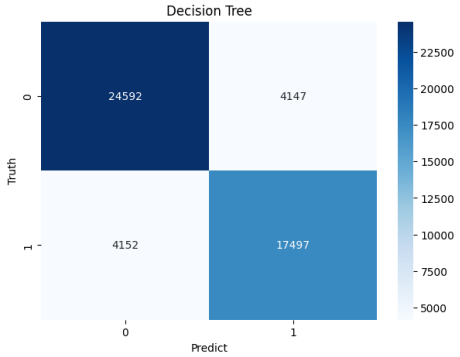
Figure 5.3: ROC-AUC curve of the implemented models

The confusion matrices for the results of the performance of models used on the dataset with the best features are given below.



(a) Confusion Matrix of XGB

(b) Confusion Matrix of RF



(c) Confusion Matrix of DT

Figure 5.4: Confusion Matrices of all the models implemented

5.2.2 Model Explainability

The SHAP (SHapley Additive exPlanations) summary plot gives an illustration of how each feature affects the Random Forest model's output for the two classes, Class 0 and Class 1, i.e., Malware and Benignware. The following provides a thorough analysis of the plot and key points:

Interpretation:

1. **Entropy:**
    - **Impact on Class 0**: When it comes to forecasting Class 0, entropy has the most impact. Class 0 is strongly influenced, as indicated by the red bar, which suggests that greater entropy levels are more representative of Class 0.
    - **Impact on Class 1**: The blue section indicates that entropy contributes less to Class 1 than it does to Class 0, with a reduced impact on Class 1 prediction.
2. **Length:**
    - **Impact on Class 0**: The lack of a red bar or a very small one for Class 0 suggests that the length attribute has little bearing on Class 0 prediction.
    - **Impact on Class 1**: The blue bar represents that length significantly improves the likelihood of Class 1. This indicates that longer values may be linked to Class 1.
3. **sha_int3:**
    - **Impact on Class 0**: Class 0 prediction is moderately positively impacted by the sha1_int3 feature.
    - **Impact on Class 1**: Class 1 is likewise impacted by this characteristic, although not as much as Class 0.

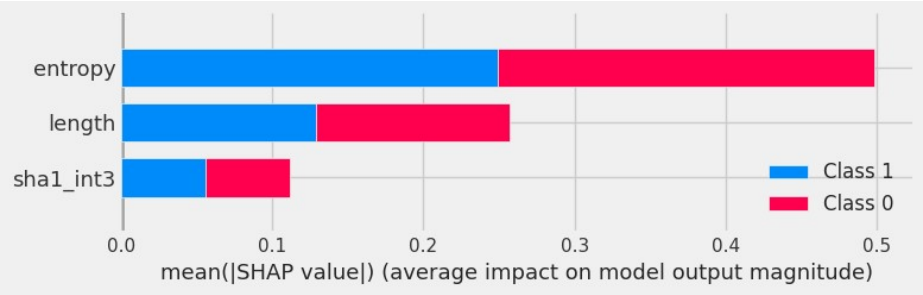
Insights for Results and Discussion::

1. **Dominant Features:**
    - **Entropy**: In terms of impacting the model's predictions, entropy is the most important factor. It shows a greater correlation with Class 0 and notably separates it from Class 1.
    - **Length**: Length is another crucial component that primarily influences forecasts for Class 0.
2. **Feature Contribution:**
    - Entropy and length mainly help in locating occurrences of Class 0, which may indicate that Class 0 is defined by particular length and entropy patterns.
    - sha1_int3 has a comparable yet balanced effect on both groups, indicating that it is more of a supporting element than a major driver.

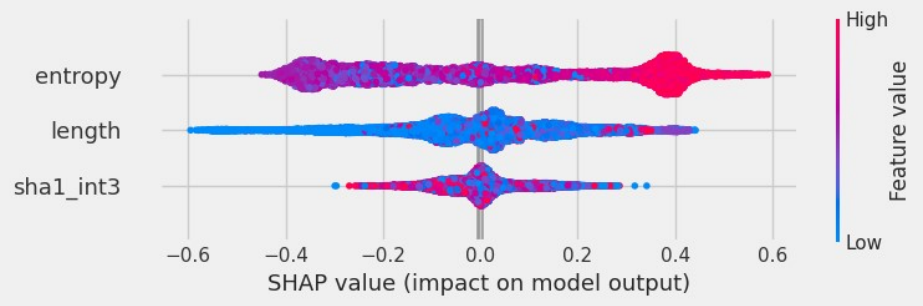
3. **Model Insights:**

   - Since entropy plays a major role in the model's ability to discriminate between the two classes, variations in entropy levels are essential for accurate classification.
   - The fact that the length feature has a large impact on Class 1 but not on Class 0 indicates that some length ranges are more likely to be connected to Class 1, designating a specific area for additional research.

The visual representation of the explainability of our classification model using the SHAP black box model is shown below.



(a) Feature impacts on overall model



(b) Feature impacts on model performance by target class

Figure 5.5: Model Explainability using SHAP

5.2.3 Clustering Results

Cluster coherence and separation are quantified via silhouette analysis, which yields coefficients between -1 and 1. Better clustering is indicated by higher numbers, which helps determine the ideal cluster count. In our study, silhouette analysis was carried out by taking clusters from range 3 to 8. **Results show that when divided into 4 clusters, the best silhouette score is produced, which is 0.42**. Elbow analysis was also performed showing there are four optimal clusters.

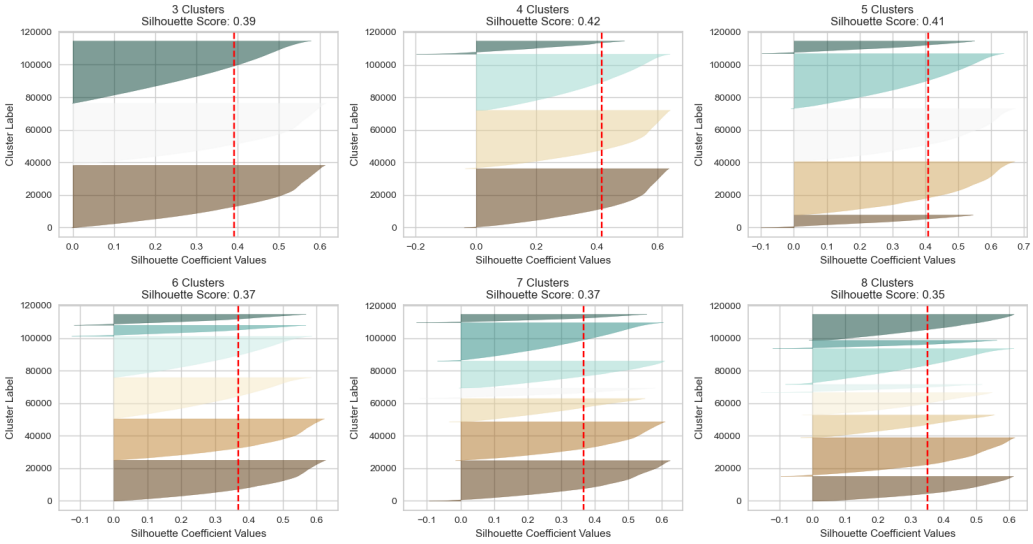


Figure 5.6: Silhoutte Analysis

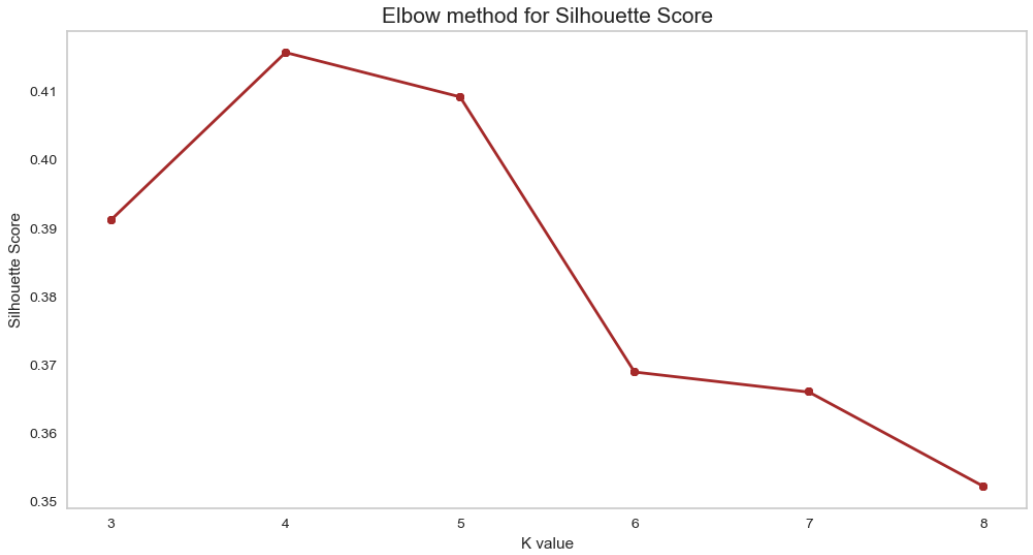


Figure 5.7: Elbow Analysis

The most widely known clustering technique, i.e., K-Means clustering was first implemented on the malware dataset consisting of 1.12 lakh malware files. As mentioned above, the malicious files were identified to be within four families with the number of samples in each cluster tabulated below. The plot for the original malware samples and after them being clustered into the respective families by K-Means clustering has been shown as follows.

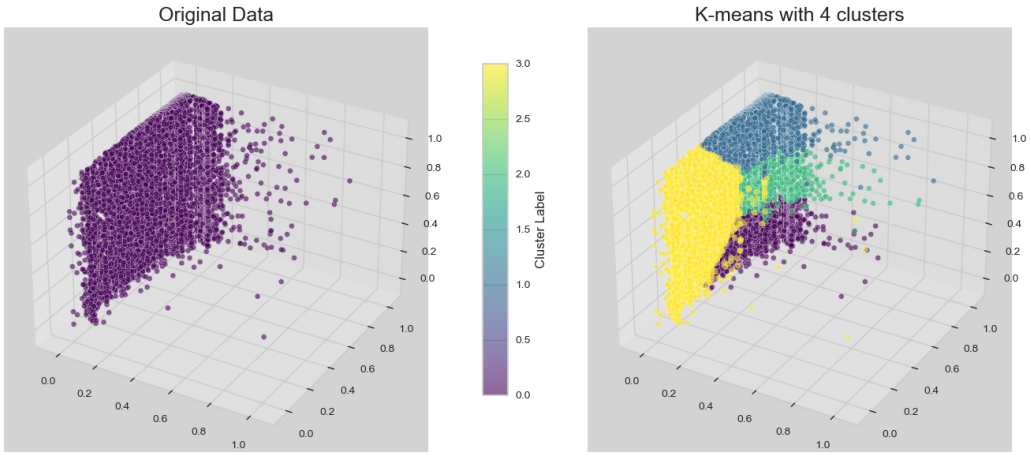


Figure 5.8: Comparison of original data v/s K-Means clustered malware data

In contrast to k-means, which allocates each data point to a single cluster, Fuzzy c-means permits each data point to belong to numerous clusters with differing degrees of membership. This is especially helpful for malware analysis because a piece of malware can have properties in common with several families. **Using this method, the malicious files were clustered into four families producing a better silhouette score of 0.48**, which indicates better division among the samples. The plot for the original malware samples and after them being clustered into the respective families by Fuzzy C-Means clustering has been shown as follows.

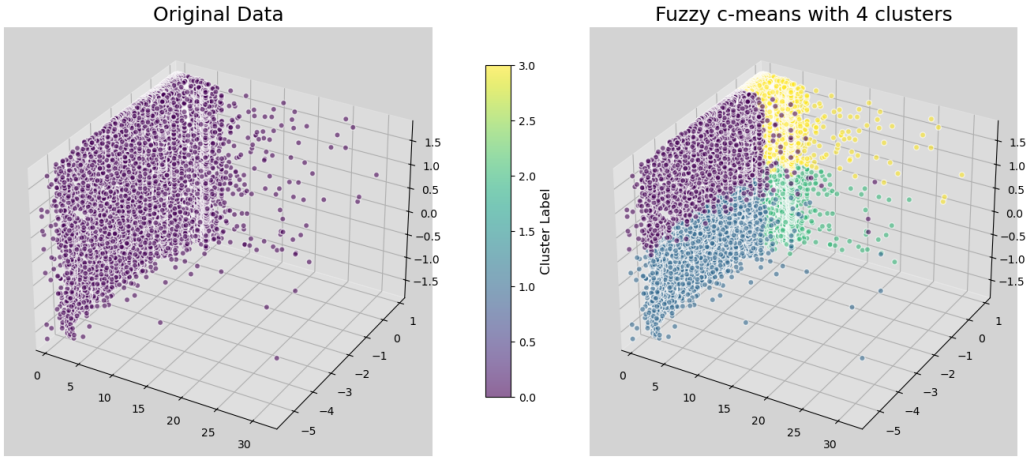


Figure 5.9: Comparison of original data v/s Fuzzy C-Means clustered malware data

Chapter 6

Conclusion and Future Work

To summarize, our experiment shows how machine learning models may be effectively combined with Explainable Artificial Intelligence (XAI) methods to improve malware detection and family recognition. We attain good accuracy in spotting malware and enhance transparency and confidence in the decision-making process by utilizing advanced algorithms and interpretability techniques. Stronger cybersecurity protections are made possible by our method, which enables deeper insights into malware activities and features. Progress in malware analysis and mitigation will be made in the future by honing these models and broadening their scope of use.

The project opens several avenues for future research and development:

- **Real-time Detection Systems:** Developing and deploying real-time malware detection systems leveraging the proposed methodology can provide immediate protection against emerging threats.

- **Deep Learning Models:** Investigating the use of deep learning architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), could further enhance detection capabilities, especially for complex and polymorphic malware.

References

[1] Hala Ahmed, Hassan Soliman, and Mohammed Elmogy. "Early detection of Alzheimer's disease using single nucleotide polymorphisms analysis based on gradient boosting tree". In: *Computers in Biology and Medicine* 146 (2022), p. 105622.

[2] Majed Alateeq. "Conditional Fuzzy C-Means (CFCM): Explanation and Implementation". In: (2022). URL: https://medium.com/@alateeq/conditional-fuzzy-c-means-cfcm-explanation-and-implementation-9614441ccc3d.

[3] Moosa Ali. "Boruta Feature Selection Explained in Python". In: (2022). URL: https://medium.com/geekculture/boruta-feature-selection-explained-in-python-7ae8bf4aa1e7.

[4] Muhammad Amin et al. "Cyber security and beyond: Detecting malware and concept drift in AI-based sensor data streams using statistical techniques". In: *Computers and Electrical Engineering* 108 (2023), p. 108702.

[5] Aalap Arun, Anjaly S Nair, and AG Sreedevi. "Zero Day Attack Detection and Simulation through Deep Learning Techniques". In: *2024 14th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE. 2024, pp. 852–857.

[6] Salar Askari. "Fuzzy C-Means clustering algorithm for data with unequal cluster sizes and contaminated with noise and outliers: Review and development". In: *Expert Systems with Applications* 165 (2021), p. 113856.

[7] Cengiz Avcı et al. "Comparison between random forest and support vector machine algorithms for LULC classification". In: *International Journal of Engineering and Geosciences* 8.1 (2023), pp. 1–10.

[8] Pieter Barnard, Nicola Marchetti, and Luiz A DaSilva. "Robust network intrusion detection through explainable artificial intelligence (XAI)". In: *IEEE Networking Letters* 4.3 (2022), pp. 167–171.

[9] Ioan-Daniel Borlea et al. "A unified form of fuzzy C-means and K-means algorithms and its partitional implementation". In: *Knowledge-Based Systems* 214 (2021), p. 106731.

[10] Tianqi Chen et al. "Xgboost: extreme gradient boosting". In: *R package version 0.4-2* 1.4 (2015), pp. 1–4.

[11] Zhiyang Fang et al. "Feature selection for malware detection based on reinforcement learning". In: *IEEE Access* 7 (2019), pp. 176177–176187.

[12] AV-TEST GmbH. "AV-TEST Product Review and Certification Report". In: (2022). URL: https://www.av-test.org/en/antivirus/home-windows/windows-10/test/jan-feb-2022-withsecures-elements-endpoint-protection-21-22/.

[13] Ibrahim Gulatas et al. "Malware threat on edge/fog computing environments from Internet of things devices perspective". In: *IEEE Access* 11 (2023), pp. 33584–33606.

[14] Sibel Gulmez, Arzu Gorgulu Kakisim, and Ibrahim Sogukpinar. "XRan: Explainable deep learning-based ransomware detection using dynamic analysis". In: *Computers & Security* 139 (2024), p. 103703.

[15] "HAFNIUM targeting Exchange Servers with 0-day exploits". In: (2021). URL: https://www.microsoft.com/en-us/security/blog/2021/03/02/hafnium-targeting-exchange-servers/.

[16] Xinfeng He and Riyang Li. "Malware detection for container runtime based on virtual machine introspection". In: *The Journal of Supercomputing* 80.6 (2024), pp. 7245–7268.

[17] Abiodun M Ikotun et al. "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data". In: *Information Sciences* 622 (2023), pp. 178–210.

[18] Md Jobair Hossain Faruk et al. "Malware Detection and Prevention using Artificial Intelligence Techniques". In: *arXiv e-prints* (2022), arXiv–2206.

[19] Olha Jurečková et al. "Classification and online clustering of zero-day malware". In: *Journal of Computer Virology and Hacking Techniques* (2024), pp. 1–14.

[20] Olha Jurečková et al. "Classification and online clustering of zero-day malware". In: *Journal of Computer Virology and Hacking Techniques* (2024), pp. 1–14.

[21] Michael Lester. "PE Malware Machine Learning Dataset". In: (2021). URL: https://practicalsecurityanalytics.com/pe-malware-machine-learning-dataset.

[22] Xuan Liu et al. "Adapting feature selection algorithms for the classification of Chinese texts". In: *Systems* 11.9 (2023), p. 483.

[23] Zephin Livingston. "What is Malware? Definition, Purpose Common Protections". In: (2022). URL: https://www.esecurityplanet.com/threats/malware/#spyware.

[24] Pengcheng Luo, Jian Chu, and Genke Yang. "IP packet-level encrypted traffic classification using machine learning with a light weight feature engineering method". In: *Journal of Information Security and Applications* 75 (2023), p. 103519.

[25] Vrinda Malhotra, Katerina Potika, and Mark Stamp. "A comparison of graph neural networks for malware classification". In: *Journal of Computer Virology and Hacking Techniques* 20.1 (2024), pp. 53–69.

[26] Rosie Yuyan Zou Matthias Schonlau. "The random forest algorithm for statistical learning". In: (2020). URL: https://doi.org/10.1177/1536867X20909688.

[27] Ritik Mehta, Olha Jurečková, and Mark Stamp. "A natural language processing approach to Malware classification". In: *Journal of Computer Virology and Hacking Techniques* 20.1 (2024), pp. 173–184.

[28] Omolbani Mohammadrezapour, Ozgur Kisi, and Fariba Pourahmad. "Fuzzy c-means and K-means clustering with genetic algorithm for identification of homogeneous regions of groundwater quality". In: *Neural Computing and Applications* 32 (2020), pp. 3763–3775.

[29] Nour Moustafa et al. "Explainable intrusion detection for cyber defences in the internet of things: Opportunities and solutions". In: *IEEE Communications Surveys & Tutorials* (2023).

[30] Samreen Naeem et al. "An unsupervised machine learning algorithms: Comprehensive review". In: *International Journal of Computing and Digital Systems* (2023).

[31] Minh Tu Nguyen, Viet Hung Nguyen, and Nathan Shone. "Using deep graph learning to improve dynamic analysis-based malware detection in PE files". In: *Journal of Computer Virology and Hacking Techniques* 20.1 (2024), pp. 153–172.

[32] Zequn Niu et al. "A novel approach based on adaptive online analysis of encrypted traffic for identifying Malware in IIoT". In: *Information Sciences* 601 (2022), pp. 162–174.

[33] Erdal Ozkaya. *Cybersecurity: the beginner's guide: a comprehensive guide to getting started in cybersecurity*. Packt Publishing Ltd, 2019.

[34] Anastasia Pereberina, Alexey Kostyushko, and Alexander Tormasov. "An algorithm for scheduling of threads for system and application code split approach in dynamic malware analysis". In: *Journal of Computer Virology and Hacking Techniques* 19.3 (2023), pp. 459–468.

[35] Robert P Sheridan et al. "Extreme gradient boosting as a method for quantitative structure–activity relationships". In: *Journal of chemical information and modeling* 56.12 (2016), pp. 2353–2360.

[36] Jaime Lynn Speiser et al. “A comparison of random forest variable selection methods for classification prediction modeling”. In: *Expert systems with applications* 134 (2019), pp. 93–101.

[37] “Unsupervised learning”. In: (2024). URL: https://en.wikipedia.org/wiki/Unsupervised_learning.

[38] Yulong Wang et al. “DockerWatch: a two-phase hybrid detection of malware using various static features in container cloud”. In: *Soft Computing* 27.2 (2023), pp. 1015–1031.

[39] Yanfang Ye et al. “A survey on malware detection using data mining techniques”. In: *ACM Computing Surveys (CSUR)* 50.3 (2017), pp. 1–40.

[40] Caizhi Zhang et al. “Review of clustering technology and its application in coordinating vehicle subsystems”. In: *Automotive Innovation* 6.1 (2023), pp. 89–115.

[41] Kaile Zhou and Shanlin Yang. “Effect of cluster size distribution on clustering: a comparative study of k-means and fuzzy c-means clustering”. In: *Pattern Analysis and Applications* 23.1 (2020), pp. 455–466.