Advances in Networks
2025, Vol. 12, No. 1, pp. 19-28
https://doi.org/10.11648/j.net.20251201.12

Research Article

Efficient Malware Classification Using Multiprocessing and Bag-of-Words Vectorization

Avijit Chowdhury[1, *], Touhidul Alam Seyam[2], Moin Uddin Ahmed Babar[1], Chonchal Khan[3], Saima Akter[4]

[1]Department of Mechanical Engineering, Chittagong University of Engineering and Technology, Chattogram, Bangladesh
[2]Department of Computer Science and Engineering, BGC Trust University, Chattogram, Bangladesh
[3]Department of Computer Science and Engineering, Daffodil international University, Dhaka, Bangladesh
[4]Department of Computer Science and Engineering, University of Science and Technology, Chittagong, Bangladesh

Abstract

The increasing incidence of malware presents significant obstacles to cybersecurity, necessitating sophisticated and effective techniques for detection and categorization. This paper presents a novel method that improves the precision and efficacy of malware classification by utilizing multi-processing and Bag-of-Words (BoW) vectorization. The suggested technique utilizes feature vectors to depict malware samples, leveraging the parallel processing capabilities of contemporary computer systems to expedite classification operations. This solution centers on a bespoke HexVectorizer that transforms hexadecimal strings obtained from malware binaries into feature vectors. The approach employs a balanced subset of the Microsoft Malware Classification dataset, meticulously preprocessed for dependable evaluation to assure thorough analysis. Python's multiprocessing module is utilized to meet the computing requirements of extensive datasets, facilitating the parallelization of vectorization processes and markedly enhancing processing performance. The classification system is centered on XGBoost's XGBClassifier, recognized for its superior performance and accuracy in addressing malware detection and classification issues. The experimental findings validate the efficacy of the suggested technique in real-time malware detection, confirming its relevance to diverse cybersecurity contexts. This paper offers an exhaustive elucidation of the implementation procedure, accompanied by thorough performance assessments. The results highlight the method's promise as a scalable and efficient approach for tackling the increasing issues of malware classification in cybersecurity. This research greatly advances the creation of resilient and efficient malware detection systems by integrating sophisticated vectorization techniques with cutting-edge machine learning algorithms, therefore fulfilling essential requirements in a dynamic threat environment.

Keywords

Bag-of-Words (BoW), Cybersecurity, Multiprocessing, Malware Classification, Machine Learning

[*]Corresponding author: u1903045@student.cuet.ac.bd (Avijit Chowdhury)

Received: 3 January 2025; **Accepted:** 20 January 2025; **Published:** 21 March 2025

1. Introduction

The rapid spread of malware is a persistent and major problem in cybersecurity. There is a pressing need for novel methods to effectively detect and categorize malware, as malevolent actors are always developing more complex ways to avoid detection and corrupt systems. This research proposes a unique malware classification methodology integrating Bag-of-Words (BoW) vectorization techniques and multiprocessing. This research is motivated by the need to defend digital infrastructure against an ever-evolving array of cyber-attacks.

Given the vast number and diversity of malware variants, conventional signature-based detection algorithms have found it challenging to remain current. These strategies are less successful against new, unknown, or polymorphic threats because they depend on well-known malware patterns and traits that have already been recognized. Thus, it is necessary to investigate substitute tactics that may offer stronger and more flexible defenses. The proposed methodology seeks to improve the malware classification accuracy and efficiency by utilizing the parallel processing power of contemporary computer architectures and BoW vectorization to represent malware samples as feature vectors.

This research aims to address the main issue of the inability of standard malware detection algorithms to handle a large and continuously increasing number of malware variants. Because signature-based techniques rely on preexisting knowledge, they are becoming less successful in recognizing novel and complex malware. More dynamic and scalable methods are required to adjust to the ever-changing threat landscape without incurring exorbitant computing expenses.

How might integrating multiprocessing and Bag-of-Words (BoW) vectorization techniques enhance the effectiveness and precision of malware classification compared to conventional signature-based methodologies.

By combining multi-processing and Bag-of-Words (BoW) vectorization approaches, this research seeks to enhance the field of malware classification significantly. The main goal is to provide a solid framework that presents an extensive malware categorization approach that may change to accommodate evolving cyber threats. This work aims to reduce computational costs by improving malware classification accuracy and efficiency by utilizing the parallel processing power of contemporary computer systems.

This study presents a methodical procedure for gathering data, preprocessing, extracting features, and training a malware-specific model. This approach ensures that every stage is thoughtfully planned to maximize the classification system's overall performance. Moreover, performance metric analysis and experimentation were used for empirical validation to demonstrate the effectiveness of the proposed methodology in practical settings.

2. Literature Review

Malware classification is a critical component of cybersecurity, and the aim is to identify and categorize malicious software to protect computer systems and networks. This section provides a comprehensive review of the existing literature, focusing on the role of multi-processing and Bag-of-Words (BoW) vectorization in malware analysis. Latent Support Measure Machines (latent SMM) enhance Bag-of-Words (BoW) data classification by estimating latent vectors for words, capturing the semantic relationships and co-occurrence patterns that traditional Support Vector Machines (SVMs) miss. This approach leads to state-of-the-art accuracy in BoW text classification and demonstrates robustness in terms of hyperparameters. Additionally, latent SMM is valuable for visualizing the relationships between words. In contrast, SVMs struggle to accurately reflect the co-occurrence of similar words because the kernel values between data points may not be properly defined [1]. AttentionXML proposes a label tree-based deep learning model for extreme multi-label text classification, featuring a multilabel attention mechanism that uses raw text input to capture the most relevant parts of the text for each label and a shallow, wide probabilistic label tree (PLT) for efficiently handling millions of labels, especially rare "tail labels." Whereas traditional Bag-of-Words (BoW) methods overlook word context and deep semantic information, and deep learning methods often struggle with scalability and capturing subtext, AttentionXML outperforms all existing methods on six benchmark datasets, particularly for handling tail labels among label tree-based approaches [2]. A novel approach for malware classification leverages multi-processing and Bag-of-Words (BoW) vectorization to enhance efficiency. The method uses multithreading to process data in parallel across all CPU cores, thereby significantly optimizing the computation time. Employing bigram BoW and pixel intensity features improves accuracy and ensures efficient data processing through parallel execution [3]. Wang et al. proposed a new malware classification system that achieved 99.87% accuracy using the XGBoost algorithm. This system utilizes a fusion feature set that combines binary and assembly malware features through a forward feature stepwise selection technique based on the BIG2015 dataset. While small n-grams may fail to capture complex code patterns owing to code reuse in malware, the fusion feature set effectively addresses this issue. The LightGBM and CatBoost algorithms were also tested, achieving accuracies of 99.84% and 99.76%, respectively [4]. The authors present a hybrid XceptionCNN-LightGBM model for Windows Malware classification utilizing the Malimg malware dataset. This model integrates the XceptionCNN architecture with the LightGBM algorithm to enhance classification accuracy and performance. While the generic LightGBM algorithm achieved a classification accuracy of 99% True Positive Rate (TPR), the proposed

XceptionCNN-LightGBM technique achieved a remarkable 100% TPR, indicating superior performance. However, the authors acknowledge the need for further improvements in accuracy and performance, particularly when scaling up to larger sample sizes [5]. The article fails to address effective malware classification by multiprocessing and Bag-of-Words vectorization. It emphasizes URL-based malware detection via NLP approaches, notably implementing the ROBERTa model for enhanced accuracy and minimal false-positive rates [6]. Employed multiprocessing and Bag-of-Words vectorization for effective malware categorization. Utilized multithreading to process data for optimization concurrently [7]. This study presents a hybrid model that combines XceptionCNN and LightGBM for malware classification, attaining a 100% True Positive Rate and decreased prediction durations. It omits the discussion of multiprocessing and Bag-of-Words vectorization for effective malware classification [8]. The authors concentrate on a machine learning methodology for malware classification, including data analysis, feature engineering, and modeling techniques. Nevertheless, it does not explicitly encompass multiprocessing or Bag-of-Words vectorization methodologies [9]. The study omits the topic of effective malware classification by multiprocessing and Bag-of-Words vectorization. The study emphasizes using byte sequences from executable files to identify and classify IoT malware with machine learning techniques such as SVM, KNN, and MLP [10]. The report fails to discuss effective malware classification via multiprocessing and Bag-of-Words vectorization. The framework employs machine learning methods for malware identification and classification, utilizing Cuckoo Sandbox for analysis and the Weka Framework for model construction [11]. The study does not address effective malware categorization by multiprocessing or bag-of-words vectorization. The study emphasizes the categorization of malware using extracted printable strings with several classification methods, attaining a classification accuracy of 97% without the aforementioned methodologies [12]. The study emphasizes memory-efficient malware detection with a bag-of-words methodology, achieving a 95% reduction in memory consumption during training. Nonetheless, it does not explicitly discuss multiprocessing methodologies for malware categorization [13]. The study omits the topic of effective malware classification by multiprocessing and Bag-of-Words vectorization. It emphasizes a hybrid methodology integrating static and dynamic analytic capabilities with machine learning to forecast malware execution duration and family categorization [14].

3. Methodology

Data preparation, feature extraction, parallel processing, and model training are important phases in the multiprocessing and Bag-of-Words (BoW) vectorization approach for malware classification. This section outlines each procedure to explain the methodology employed in this study thoroughly.

3.1. Data Preprocessing

3.1.1. Data Loading and Exploration

The dataset was loaded from. Byte files, each containing a hexadecimal representation of the malware binaries. Initial exploratory data analysis (EDA) was conducted to understand the distribution and characteristics of the data, including inspecting the malware families and the size of the malware. Bytes files.

3.1.2. Data Balancing

The dataset was imbalanced, with some malware families being more represented than others. To address this, the authors sampled the dataset to create a balanced subset, ensuring each malware family has an equal number of samples. This step is crucial to prevent the model from being biased towards more prevalent families.

3.2. Feature Extraction

3.2.1. Hexadecimal String Extraction

The hexadecimal strings were extracted from. Byte files by removing non-hexadecimal characters and memory addresses. This results in clean sequences of hex values that represent malware binary content.

3.2.2. Custom HexVectorizer

A custom HexVectorizer class was developed to transform hex strings into feature vectors. This class leverages Scikit learnaˆs CountVectorizer to tokenize the hex values and convert them into a sparse matrix of token counts. The HexVectorizer processes the hex values in blocks (e.g., two bytes at a time) to build a Bag-of-Words (BoW) model.

3.2.3. Including Additional Features

In addition to the BoW vectors, here is the file size for each. Byte file is included as an additional feature. This helps to capture the differences in file sizes across various malware families, potentially improving the classification accuracy.

3.3. Parallel Processing (Multiprocessing) for Feature Extraction

Owing to the large size of the dataset, the feature extraction is computationally intensive. To address this, the authrs implemented multi-processing using Python's multi-processing library. The data are split into chunks, and the parallelize function distributes these chunks across multiple CPU cores for concurrent processing, thereby significantly reducing the time required for vectorization.

Figure 1. Parallel Processing Using a Pool of Processes.

3.4. Model Training

3.4.1. XGBoost Classifier Selection

The XGBClassifier from the XGBoost library was selected for its strong performance and scalability. XGBoost is recognized for its efficiency and precision in dealing with structured data, making it ideal for classification tasks.

3.4.2. Hyperparameter Tuning

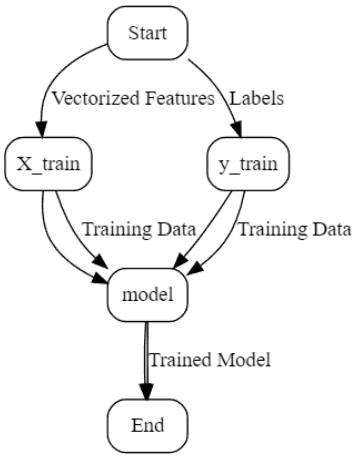

Figure 2. Process of Training an XGBoost Classifier.

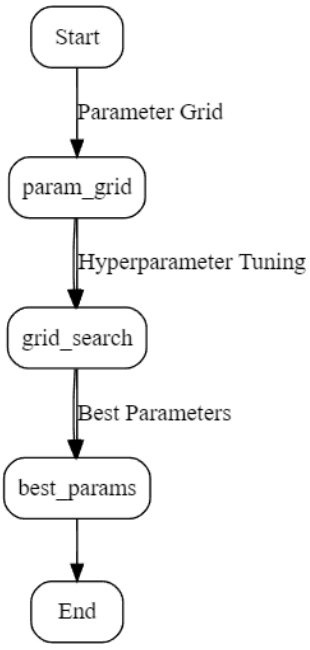

Figure 3. Grid Search for Hyperparameter Tuning.

To optimize the performance of the model, hyperparameter tuning was performed using GridSearch CV. Through cross-validation, parameters such as the learning rate, maximum depth, and number of estimators were fine-tuned.

3.4.3. Model Training

The XGBClassifier was trained on the prepared training dataset using the optimized hyperparameters.

model = XGBClassifier(**best params) model.fit(X train, y train)

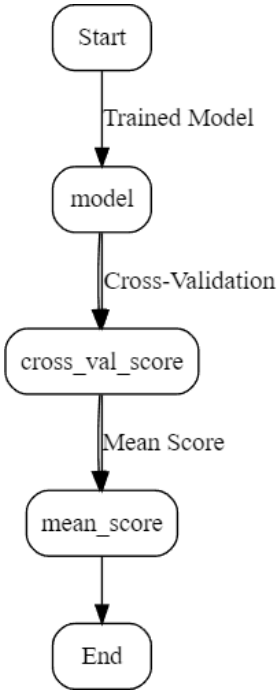

Figure 4. Cross-Validation for Model Evaluation.

3.5. Model Evaluation

3.5.1. Cross-Validation

Cross-validation was performed to evaluate the performance of the model. The dataset was divided into training and validation sets multiple times to ensure robustness and generalizability of the model.

3.5.2. Performance Metrics

The model's accuracy, precision, recall, and F1-score were calculated to provide a comprehensive evaluation of its performance [15]. These metrics assess the model's effectiveness

in correctly classifying the malware samples.

3.5.3. Comparison with Baseline Models

The performance of the XGBClassifier was compared with baseline models, such as logistic regression and random forest. This comparison demonstrated the superiority of XGBoost in handling malware classification tasks.

This methodology outlines a systematic approach to malware classification, from pre-processing and feature extraction to parallel processing and model training. The authors provide a robust and scalable solution for large-scale malware detection and classification by combining traditional machine-learning techniques with advanced computational strategies.

4. Results and Discussions

The problem at hand involves a dataset of known malware files contained within train.7z and test.7z zip files, encompassing nine different malware families. The class labels for these malware files are provided in a separate trainLabels.csv CSV file. Each malware file in the training and test sets is uniquely identified by a 20-character hash value serving as the file name, and each file belongs to a specific malware family denoted by an integer class label. The malware families included in the dataset are Ramnit, Lollipop, Kelihos ver3, Vundo, Simda, Tracur, Kelihos ver1, Obfuscator. ACY, and Gatak.

Each file's raw data contains the hexadecimal representation of the binary content, with the PE header excluded to ensure sterility. Additionally, a metadata manifest (files with a.asm extension) is provided, containing various metadata information extracted from the binary, such as function calls and strings, generated using the IDA disassembler tool.

To classify files into malware families, the following steps are taken

1) Equal samples are randomly selected from each class, except Simda.
2) Hexadecimal strings (length 2) are extracted from bytes files.
3) A Bag-of-Words model is constructed using CountVectorizer from scikit-learn.
4) A classification model is built using XGBClassifier from XGBoost.

The objective of this solution is to select a balanced sample from the original dataset and improve the efficiency of the solution by parallelizing computationally intensive tasks and file operations given the substantial size of the dataset.

The project structure was meticulously organized to streamline the handling, preprocessing, and analysis of the malware classification dataset. The main directories and files are as follows

The code/ directory contains

1) Microsoft Malware Classification (with Multiprocessing). ipynb: This Jupyter notebook contains the main code for the project.
2) HexVectorizer.py: A Python script for the custom HexVectorizer used in feature extraction.
3) Submission.csv: The final submission file containing the classification results.

The input/ directory contains the following:

malware-classification/:

a. trainLabels.csv: The CSV file with class labels for the training dataset.
b. train/: This folder contains multiple. byte files, each representing the hexadecimal content of a malware sample, such as 0ACDbR5M3ZhBJajygTuf. bytes.
c. test/: This folder includes multiple. byte files for the test set, such as ITSUPtCmh7WdJcsYDwQ5.bytes.

The Microsoft-malware-sample/ directory includes:

1) trainLabels bal.csv: The CSV file with balanced class labels for the training dataset.
2) The file containing the vectorized training dataset.
3) test vec.csv: The file containing the vectorized test dataset.

4.1. Balanced Subset from Original Dataset

When working with a dataset, it's often important to create a balanced subset to ensure that analyses and insights drawn from the data are representative of the larger population. A balanced subset typically maintains proportions of various classes or categories present in the original dataset, which can be crucial for tasks like machine learning and statistical analysis.

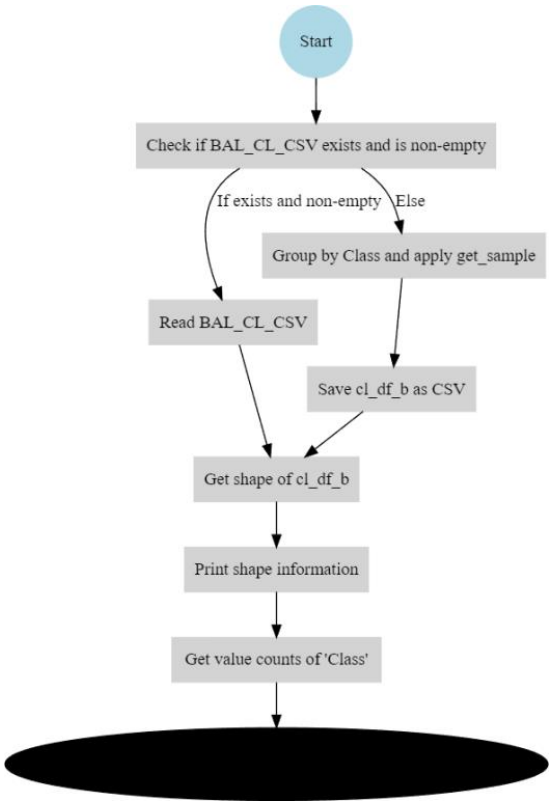

Figure 5. *Workflow for Checking, Loading, and Processing Balanced Malware Subset Data.*

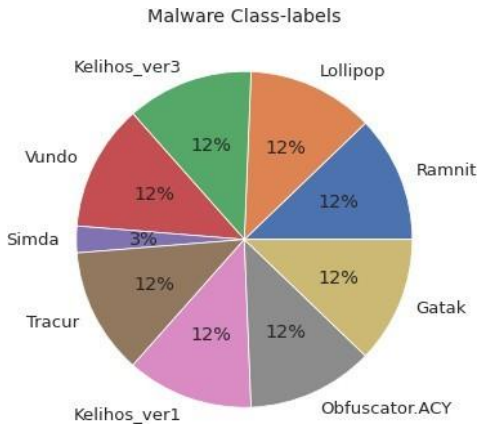

Figure 6. *Malware Class-levels.*

4.2. High-Dimensional Data Visualization

High-dimensional data visualization is a critical process in data analysis that helps to represent complex datasets with numerous variables in a more understandable and visually interpretable format. Data visualization techniques are derived from certain aspects presented in the referenced paper [15].

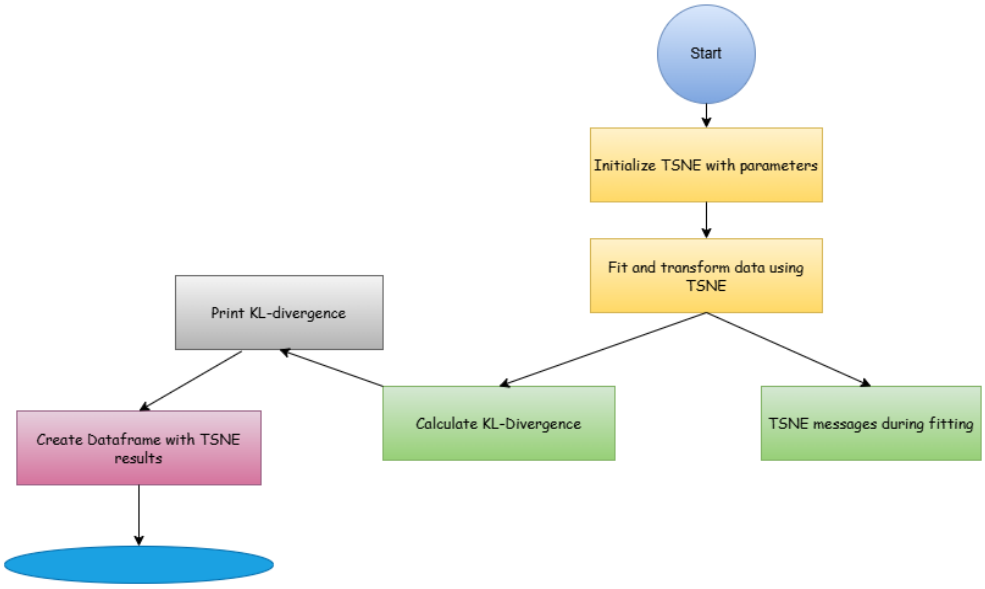

Figure 7. *Workflow for TSNE Transformation and Visualization of Malware Data.*

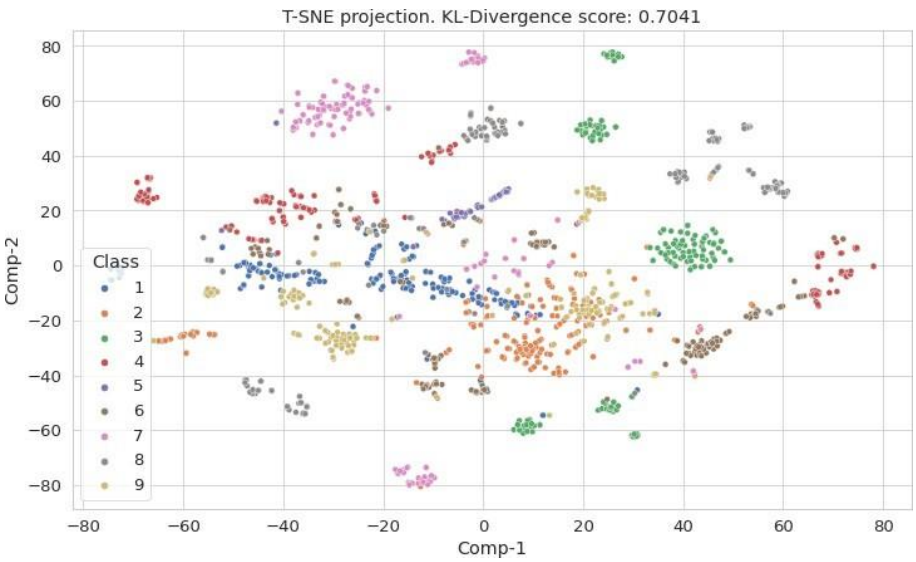

Figure 8. *High-Dimensional Data Visualization.*

4.3. Tuning

To determine the optimal range for the hyperparameters mentioned above, plot the training and validation (or test) errors. Then, identify the optimal point where:

1) The validation error is minimized.
2) The gap between the training error and the validation error is smallest.

4.3.1. Param: n Estimators

Hyperparameter tuning (n estimators=range (1, 70, 10))

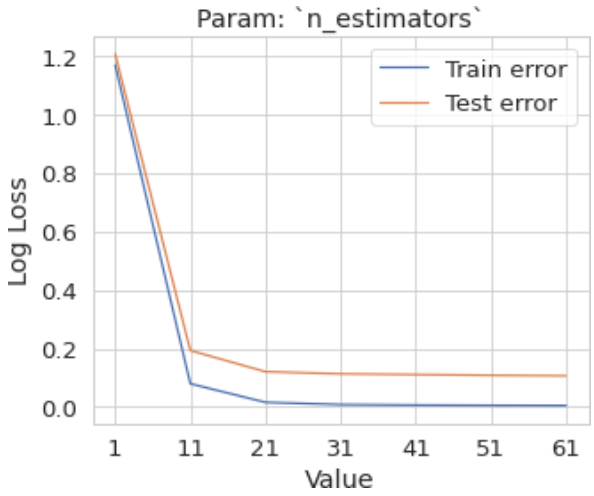

Figure 9. *Param: n estimators.*

4.3.2. *Param: max depth* Hyperparameter Tuning (n Estimators=40, Max Depth=Range (1, 10))

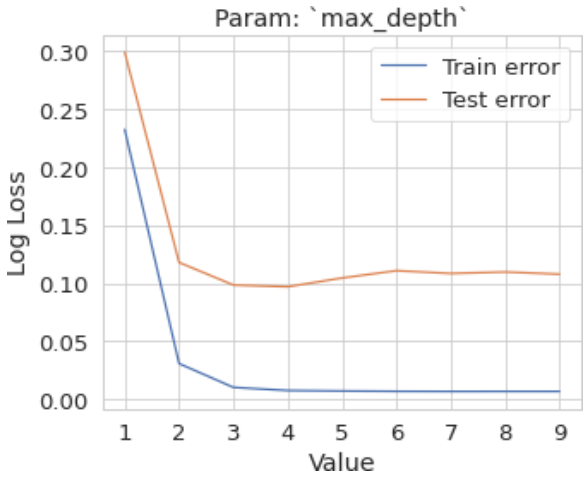

Figure 10. *Param: max depth.*

4.3.3. Param: Learning Rate

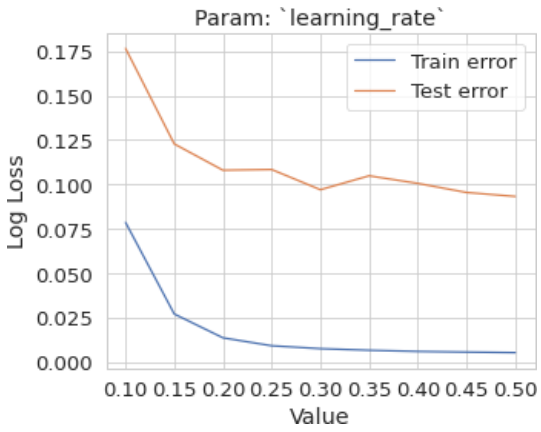

Figure 11. *Param: learning rate.*

4.3.4. Param: Reg Alpha

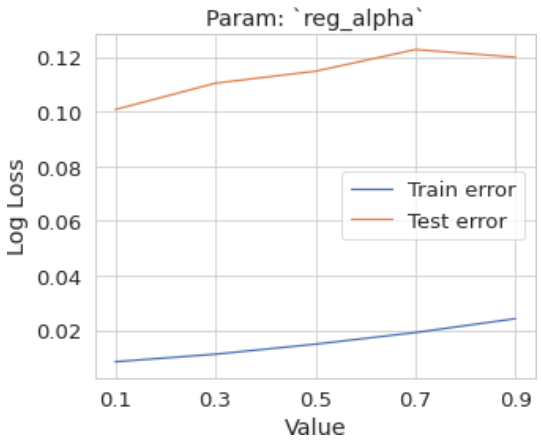

Figure 12. *Param: Param: reg alpha.*

4.3.5. Param: Reg Lambda

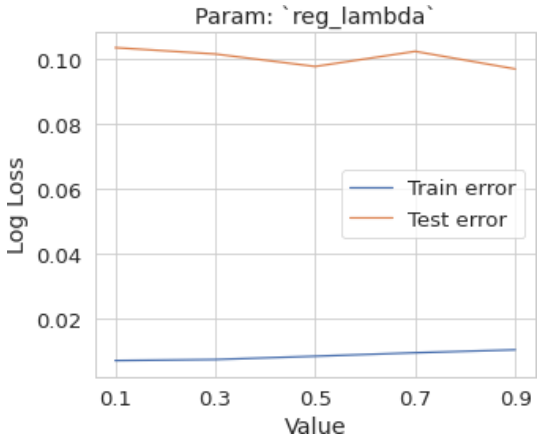

Figure 13. *Param: Param: reg lambda.*

The project involved classifying malware files into their respective families using a combination of feature extraction, hyperparameter tuning, and model evaluation [15, 16].

1) Grid Search CV The model employs a Grid Search CV with the XGBClassifier to fine-tune the parameters. The best parameters identified were a learning rate of 0.4, a maximum depth of 4, and 45 estimators, achieving an accuracy of 0.9604.
2) Model Testing Feature engineering is performed by loading a vectorized test dataset or generating feature vectors from byte files. The test dataset was vectorized and saved for future use.

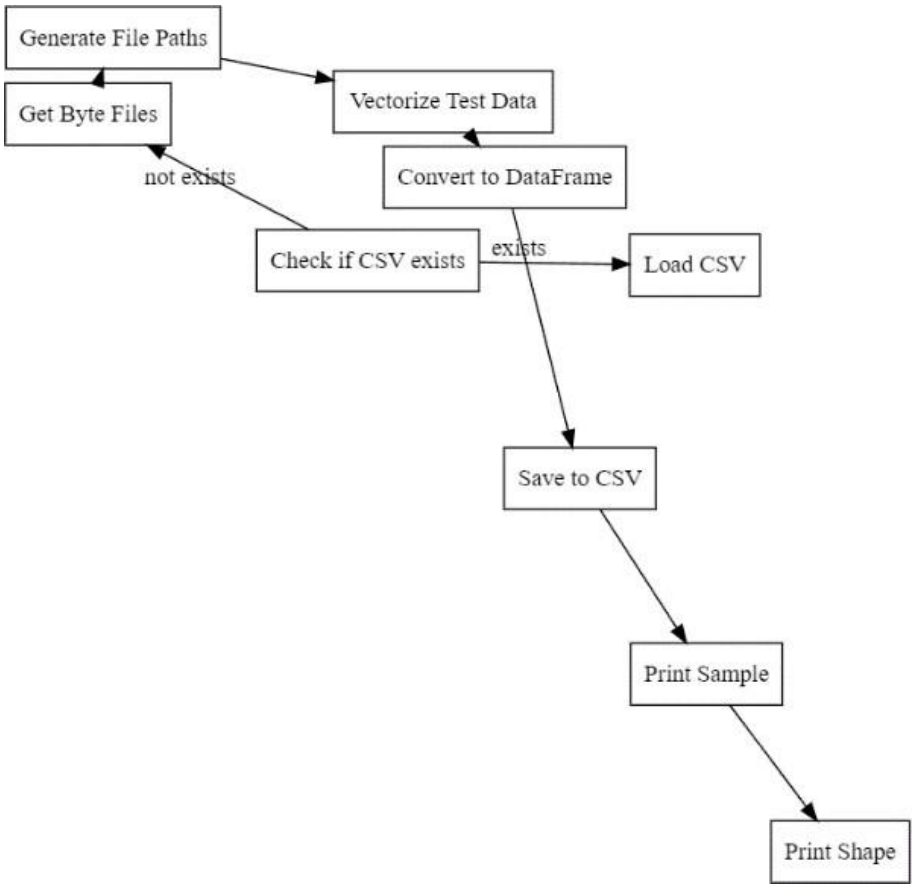

Figure 14. *Steps of the Feature Engineering Process for the Test Dataset.*

Predictions Finally, the authors generated predictions
Using the trained model and saved the results. Here's the process:

1) Generate Predictions
2) Prepare the Output Data
3) Save the Predictions to CSV

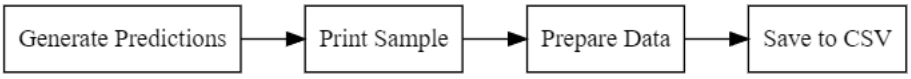

Figure 15. *The Steps Involved in Generating and Saving the Predictions.*

5. Conclusions

The increasing incidence of malware presents significant obstacles to cybersecurity, necessitating sophisticated and effective techniques for detection and categorization. This paper presents a novel method that improves the precision and efficacy of malware classification by utilizing multi-processing and Bag-of-Words (BoW) vectorisation. The proposed technique utilizes feature vectors to characterize malware samples, using the parallel processing capabilities of

contemporary computer systems to enhance classification speed. This solution centers on a proprietary HexVectorizer that transforms hexadecimal strings obtained from malware binaries into feature vectors. The approach employs a balanced subset of the Microsoft Malware Classification dataset, meticulously preprocessed for dependable evaluation to assure thorough analysis. Python's multiprocessing module is utilized to meet the computing requirements of extensive datasets, facilitating the parallelization of vectorization processes and markedly enhancing processing performance. The classification system is centered on XGBoost's XGBClassifier, which is recognized for its superior performance and accuracy in addressing malware detection and classification issues. Experimental findings validate the proposed method's efficacy in real-time malware detection, confirming its relevance to diverse cybersecurity contexts.

This article thoroughly elucidates the implementation process, accompanied by meticulous performance assessments. The results highlight the method's capacity as a scalable and efficient approach for tackling the increasing issues of malware classification in cybersecurity. This research greatly advances the creation of resilient and efficient malware detection systems by integrating sophisticated vectorization techniques with cutting-edge machine learning algorithms, fulfilling essential requirements in a dynamic threat environment.

Abbreviations

BOW Bag of Words

Funding

This research was entirely self-funded by the authors with no external funding received.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Yoshikawa, Yuya, Tomoharu Iwata, and Hiroshi Sawada. “Latent Support Measure Machines for Bag-of-Words Data Classification.” Proceedings Article, 2014.

[2] Ronghui, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. “AttentionXML: Label Tree-based Attention-Aware Deep Model for High- Performance Extreme Multi-Label Text Classification.” Proceedings Article, 2019.

[3] Banerjee, Shobhan, Bibhuti Bhusan Dash, M. Rath, Tanmaya Swain, and Tapaswini Samant. “Malware Classification using Bigram BOW, Pixel Intensity Features, and Multiprocessing.” Proceedings Article, 2022. https://doi.org/10.1109/CONECCT55679.2022.9865764

[4] Chen, Zhiguo, and Xuanyu Ren. “An Efficient Boosting-Based Windows Malware Family Classification System Using Multi-Features Fusion.” Journal Article, Applied Sciences, 2023. https://doi.org/10.3390/app13064060

[5] Onoja, M. Nelson, Abayomi Jegede, N. V Blamah, Abinbola Victor Olawale, and Temidayo Oluwatosin Omotehinwa. “EEMDS: Efficient and Effective Malware Detection System with Hybrid Model based on XceptionCNN and LightGBM Algorithm.” Journal Article, 2022.

[6] Y. P. Kumar S, S. Mishra and R. K. Singh, "Unleashing the Potential of Machine Learning and NLP Contextual Word Embedding for URL-Based Malicious Traffic Classification," 2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring), Singapore, Singapore, 2024, pp. 1-5, https://doi.org/10.1109/VTC2024-Spring62846.2024.10683482

[7] S. Banerjee, B. B. Dash, M. K. Rath, T. Swain and T. Samant, "Malware Classification using Bigram BOW, Pixel Intensity Features, and Multiprocessing," 2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2022, pp. 1-5, https://doi.org/10.1109/CONECCT55679.2022.9865764

[8] M. Onoja, A. Jegede, N. Blamah, O. V. Abimbola, and T. O. Omotehinwa, “EEMDS: Efficient and Effective Malware Detection System with Hybrid Model based on XceptionCNN and LightGBM Algorithm”, JCSI, vol. 1, no. 2, pp. 42–57, Oct. 2022, https://doi.org/10.33736/jcsi.4739.2022

[9] Houssem Hosni. Machine learning approach for malware multiclass classification. BRAINS 2019 - 1st Blockchain, Robotics, AI for Networking Security Conference, Mar 2019, Rio de janeiro, Brazil. ffhal-02075139f.

[10] T. -L. Wan et al., "Efficient Detection and Classification of Internet-of-Things Malware Based on Byte Sequences from Executable Files," in IEEE Open Journal of the Computer Society, vol. 1, pp. 262-275, 2020, https://doi.org/10.1109/OJCS.2020.3033974

[11] Kamalakanta Sethi. A novel malware analysis for malware detection and classification using machine learning algorithms. In Proceedings of the 10th International Conference on Security of Information and Networks (SIN '17). Association for Computing Machinery, New York, NY, USA, 107–113. https://doi.org/10.1145/3136825.3136883

[12] R. Tian, L. Batten, R. Islam and S. Versteeg, "An automated classification system based on the strings of trojan and virus families," 2009 4th International Conference on Malicious and Unwanted Software (MALWARE), Montreal, QC, Canada, 2009, pp. 23-30, https://doi.org/10.1109/MALWARE.2009.5403021

[13] Yueming, Wang, Meng, Zhang. 13. Memory-efficient detection of large-scale obfuscated malware. International Journal of Wireless and Mobile Computing, https://doi.org/10.1504/ijwmc.2024.136586

[14] Sean, Kilgallon, Leonardo, De, La, Rosa., John, Cavazos. (2017). 14. Improving the effectiveness and efficiency of dynamic malware analysis with machine learning. https://doi.org/10.1109/RWEEK.2017.8088644

[15] A. Chowdhury, "Advancing Multi-Class Arc Welding Defect Classification: DEEPTLWELD Intelligent System Utilizing Computer Vision, Deep Learning, and Transfer Learning on Radiographic X-ray Images for Bangladesh's Manufacturing Sector," 2024 IEEE International Conference on Computing, Applications and Systems (COMPAS), Cox's Bazar, Bangladesh, 2024, pp. 1-6, https://doi.org/10.1109/COMPAS60761.2024.10796006

[16] Hossain, Sazzad, et al. "Fine-tuning LLaMA 2 interference: a comparative study of language implementations for optimal efficiency." (2025). https://doi.org/10.48550/arXiv.2502.01651