JIRBDAI

Journal of Institutional Research, Big Data Analytics and Innovation

Integrating Principal Component Analysis and Chi-Square Feature Selection for Explainable Memory-Based Malware Detection Using Machine Learning

Gilbert Aimufua, *Akinshola-Awe Funmilayo
Department of Cybersecurity, Cyberspace Center, Nasarawa State University, Nigeria
***Corresponding Author: aimufua@nsuk.edu.ng**

ABSTRACT

Antimalware fails to detect modern malware because advanced threats use obfuscation and polymorphism to evade detection. Machine learning offers alternatives through pattern recognition, but redundant and correlated features in cybersecurity datasets limit the performance and generalizability of Machine Learning based malware detectors. Principal Component Analysis and Chi-Square have proven effective individually but their combined use for malware detection remains underexplored. Hybrid PCA–Chi-Square Feature Engineering (HPCF) framework that fuses PCA components with Chi-Square features into a single compact representation was employed in this study. Feature engineering, hyperparameter optimization and model training were nested inside a stratified ten-fold cross-validation pipeline to prevent information leakage and ensure reproducibility. The fused feature set was used to train optimized Decision Tree, Support Vector Machine and K-Nearest Neighbor classifiers, evaluated using Accuracy, Precision, Recall, F1-score, ROC-AUC, Matthews Correlation Coefficient, Friedman's test and Wilcoxon Signed-Rank Test. HPCF produced a smaller feature vector while largely preserving predictive performance, with Support Vector Machine and K-Nearest Neighbor significantly outperforming Decision Tree with CICMalmem dataset. The framework was further trained on PE dataset to assess cross dataset generalizability. SHAP and LIME explanations showed that memory artefacts related to process handles, service activity, and dynamic link library characteristics were the strongest predictors of malicious behavior. The results demonstrate an efficient trade-off between dimensionality reduction, feature relevance, computational cost and model interpretability. The proposed HPCF framework offers a transparent, reproducible and explainable approach to memory-based malware detection and a basis for future work on more resilient, reliable artificial intelligence systems for cyber-threat detection.

KEYWORDS: Malware Detection, Hybrid Feature Generation, Principal Component Analysis (PCA), Memory Forensics, Cybersecurity.

I. INTRODUCTION

Malware remains one of the most prevalent and continually morphing cybersecurity threats posing a serious risk to governments, financial services organizations, healthcare systems, critical national infrastructure and individuals alike (National Institute of Standards and Technology, 2024). Malware has evolved toward increasingly sophisticated evasion techniques such as polymorphism, metamorphism, packing and encryption, code obfuscation, reflective DLL injection, process hollowing and fileless execution. These techniques allow malware to continually modify its characteristics, rendering it undetectable by classic signature-based solutions that depend entirely

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

on a previously defined fingerprint (Buriro et al., 2023; Kumar, 2022). As cybercriminals increasingly rely on behavior-based, memory-resident execution to mask malicious activity, static malware analysis has become less effective than previously in catching modern and unknown attacks. Memory-based malware is a kind of threat that is not easy, as this type of malicious code runs in volatile memory and does not ever create persistent artefacts on any disk. These types of attacks primarily abuse genuine and valid system processes using process injection, memory abuse and reflective loading techniques to enter the end point making traditional AV pointless in detection. Consequently, Memory forensic analysis has emerged as an indispensable tool to identify nefarious actions that are only evident during runtime by probing memory artefacts such as process structures, thread activities, handle operations and service registries (Dener et al., 2022) and dynamic-link library. Unlike static executable analysis, memory analysis provides contextualized evidence of execution behavior during the lifetime of programs resulting from dynamic behavior which replaces the need to separate benign applications and malicious processes.

As the malware is becoming more and more sophisticated, machine learning (ML) approaches for smart malware detection are gathering momentum in utilization. Machine learning algorithms automatically learn complicated relationships in labelled datasets and generalize sufficiently to detect previously unseen malware variants, so they are very effective for executing zero-day attacks (Boge et al., 2022; Buriro et al., 2023), unlike the conventional rule-based detection systems that requires human cyber experts. In the domain of malware detection, a number of supervised learning algorithms, including Decision Trees (DT), Support Vector Machines (SVM), Random Forests (RF), K-Nearest Neighbor (KNN) and more recently deep learning architectures have been employed with good accuracy performance (Ilham et al., 2024; Rosyada et al., 2023), but large feature space analysis has also been regarded as one of classifier training challenges. Though these methods will use algorithms to find patterns, they rely on input features that were used when building the model. High-dimensional feature spaces containing redundant, noisy and highly correlated variables are a prominent problem in cybersecurity datasets. These features increase computational complexity and reduce model interpretability, while also encouraging overfitting that lowers classifier generalization on unseen malware samples (Kwon et al., 2020). Feature engineering mitigates this by producing more compact, informative representations that reduce compute cost and, in the operational cybersecurity environment, also improve model stability, training speed, scalability and deployment reliability. Feature Engineering is typically performed by three inter-dependent functions, Feature extraction, feature selection and feature construction. Among these techniques, Principal Component Analysis (PCA) and Chi-Square (χ^2) for feature selection are the two most widely employed methods because they decrease dimensions of features but keep helpful information from a prediction standpoint. PCA is an unsupervised statistical method that converts a correlated dataset into orthogonal principal components (PCs), which are the axes that maximize the variance explained in a given dataset. PCA, moreover, usually increases computational efficiency and classifier robustness by eliminating multicollinearity and reducing redundant information (Kwon et al., 2020). PCA has been used as a feature selection technique for dimensionality reduction and redundancy removal (Kwon et al., 2020; Al-Qudah et al., 2023), while Chi-Square have been applied to determine the significant features against malware classes (Rosyada et al., 2023).

Independent PCA or Chi Square-based approaches have both been demonstrated to be effective in previous work on malware detection. For example, Zhou et al. (2023) proposed a method that used

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

PCA to construct a one-class SVM for classifying memory-dump malware, with experimental results showing a clear improvement in accuracy. Similarly, Rosyada et al. (2023) improved the accuracy of XGBoost through Chi-Square feature selection, which eliminated non-predictive attributes while retaining statistically significant features. Adi Rafrastara et al. (2025) also reported that malware detection improved when highly discriminative features were selected using Information Gain combined with Chi-Square. Though these studies support the importance of feature engineering, they only explore stand-alone or non-systematic feature selection techniques and evidence for systematic integration of unsupervised and supervised statistical feature selection into memory-based malware detection frameworks remains sparse.

Additionally, the current literature still shows some methodological limitations. Most models focused on classification accuracy but did discuss computational efficiency analysis, feature redundancy discussion, model robustness analysis and statistical significance (Ilham et al., 2024). The focus of many malware detection studies is on classifier performance evaluation in terms of prediction accuracy but less so for computational efficiency, redundancy measures (RF), model robustness, and balanced metrics such as Matthews Correlation Coefficient (MCC). Cybersecurity data hosted in cross-domain mechanisms are mostly high-dimensional with multi-faceted correlations (Chicco & Jurman, 2020; Jha et al., 2014), introducing challenges to the design of a performance guarantee relying solely on accuracy. In these domains where misclassification leads to significant consequences (false positives and false negatives) without an adaptable mechanism for improved statistical assessment, bare measurements such as accuracy yield assessable model evaluations but insufficient guarantees in operational environments full with uncertainties. Second, feature engineering is often conducted prior to data partitioning or cross-validation and so can introduce leakage of information that may inflate classification performance estimates (Kuhn & Johnson, 2019). Most of the malware detection literature does not use leakage-free experimental pipelines (practically, experimental pipelines that integrate preprocessing, feature engineering and hyperparameter optimization along with model evaluation within each cross-validation fold). Third, relatively few studies evaluate multiple feature engineering strategies in the same experimental setup making objective comparison difficult.

Finally, while explainable artificial intelligence (XAI) is of increasing importance in cybersecurity, we do not gain substantial understanding of how engineered memory features impact malware classification decisions and are useful for tailored recommendations for practical cyber threat intelligence and digital forensic investigations. Most studies in the field of malware detection mainly focus on predictive performance rather than interpretability, even though XAI has drawn more and more attention recently (Mian et al. 2022). An outcome of this is that security analysts are awash in highly accurate predictions having little more than opaque reasoning behind the memory artefact model influencing its prediction. SHAP and LIME are only partially integrated into memory-based malware detection frameworks, leading to lower transparency of the analytical results, analyst distrust in machine learning models during digital forensic investigations and incident response (Lundberg & Lee, 2017; Ribeiro et al., 2016). Lastly, there are still not many studies in which explainable machine learning outputs and operational domain knowledge frameworks such as MITRE ATT&CK have been directly linked together. Tools for explainability highlight important features, but the cybersecurity relevance of these features is rarely interpreted through the lens of attacker behaviors and tactics and techniques. Filling this void would make machine learning much more operationally valuable for a security analyst, and allow them to work with statistically meaningful explanations that correlate to established threat technology

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

frameworks. This study proposes Explainable Hybrid PCA–Chi-Square Feature Engineering (HPCF) framework, which systematically integrates unsupervised feature extraction across the entire dataset and supervised statistical feature selection to form a compact yet discriminative effective representation for memory-based malware detection. Instead of proposing a new algorithm for feature engineering, the framework combines complementary information from PCA and Chi-Square to balance dimensionality reduction, redundancy elimination and relevancy.

II. RELATED WORKS

Evolution of modern malware sophistication has quickened the shift away from traditional signature-based detection techniques to intelligent machine learning (ML)-based approaches. Traditional antivirus systems rely on standardized virus signatures and heuristic rules for detection which cannot cope with polymorphic, metamorphic, fileless and zero-day malware since they keep changing their traits to avoid being detected (Buriro et al., 2023). As a result, machine learning has become an attractive substitution due to its capability of learning intricate behavioral patterns based on previous data and generalizing well onto never-before-seen malware samples (Boge et al., 2022). Ilham et al. (2024) evaluated Random Forest, Decision Tree, Support Vector Machine, and K-Nearest Neighbor classifiers on the IoT-23 malware dataset using Grid Search optimization, finding that Random Forest performed best and underscoring the value of hyperparameter optimization.

Many studies affirm that changes in feature engineering often leads to better accuracy gains than a change of the actual classification algorithm (Guyon & Elisseeff, 2003; Domingos, 2012). As a result, one of the most vital stages within intelligent malware detection systems is done by feature engineering. Convolutional Neural Networks (CNNs) have been successful in accurately learning spatial features from malware binaries, memory dumps and behavioral features. More recently, transformer-based methods have also performed very competitively for unseen malware family classification thanks to their use of self-attention mechanisms that help model complicated behavioral relationships (Kim et al., 2024). Another key limitation is their lack of interpretability, since many deep learning models are basically black-boxes that make it hard for cybersecurity analysts to explain why a specific malware was classified as such. This black box nature could decrease trust and the usage of digital forensic investigations and operational security applications (Arrieta et al., 2020).

Feature engineering is the transformation of raw data to informative representations that lead to better machine learning results (Bishop, 2006; Han et al., 2012). Many of these variables are redundant, collinear, noisy, and have less discriminative power which results in increased computational expense at the expense of reduced generalizability (Kwon et al., 2020). PCA transforms correlated variables into a smaller number of orthogonal components - maximum variance captured from an original dataset (in terms of independent variables) with diminished dimensionality (Jolliffe & Cadima, 2016). It computes the statistical dependency of each feature on the target class through the Chi-square Test of Independence, and statistically significant features are given a higher score and retained for model training (Liu & Motoda, 2007). Rosyada et al. (2023) showed that Chi-Square feature selection substantially improved XGBoost by pruning irrelevant variables while retaining the most discriminative ones.

Nonetheless, despite such advances, there have been comparatively few studies systematically combining unsupervised data reduction and supervised statistical feature selection for memory-based malware detection. Existing studies generally assess PCA or Chi-Square on an independent

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

basis or perform the sequential feature selection without analyzing their complementary contribution. It is important to clarify what distinguishes the proposed hybridization from prior hybrid feature engineering work such as Adi Rafrastara et al. (2025), who combined Information Gain with Chi-Square. That approach, and most other reported hybrids, pair two supervised, class-relevance-driven selection criteria, so the resulting feature sets remain correlated with one another and offer limited redundancy reduction. HPCF instead fuses an unsupervised, variance-based decomposition (PCA) with a supervised, class-relevance criterion (Chi-Square), so the two components capture structurally different properties of the data: PCA removes multicollinearity and compresses redundant variance, while Chi-Square preserves interpretable, class-discriminative variables in their original form. This unsupervised-supervised pairing, evaluated within a leakage-free cross-validation pipeline and linked to SHAP/LIME explanations, is what differentiates HPCF methodologically from prior single-paradigm hybrid feature selection studies in memory-based malware detection.

Explainable Artificial Intelligence (XAI) where human-interpretable explanations of model predictions are required to transform predictions to actionable cybersecurity intelligence (Molnar, 2024). Local Interpretable Model-Agnostic Explanations (LIME) built interpretable surrogate models on the vicinity of individual instances, explaining specific malware samples being classified as malicious or benign to human analysts (Ribeiro et al., 2016). Despite these benefits, few studies exist that incorporate both predictive power and operational interpretability when dealing with malware detection by integrating feature engineering with explainability (Zhang et al., 2024).

III. METHODOLOGY

This study is guided by the following research questions:

(a). RQ1: How does the integration of Principal Component Analysis (PCA) and Chi-Square (χ^2) feature selection influence the effectiveness of memory-based malware detection compared with using either technique independently?

(b). RQ2: Does the proposed HPCF framework improve predictive performance and computational efficiency for memory-based malware detection?

(c). RQ3: Can the proposed leakage-free experimental pipeline provide reliable and reproducible evaluation of malware detection models?

(d). RQ4: How can Explainable Artificial Intelligence (XAI) improve the transparency and operational interpretability of malware detection models?

The research gaps recognized above led this study to introduce, within a memory-based malware detection context, a Hybrid PCA–Chi-Square Feature Engineering (HPCF) framework experimental design and measurements types discussed in subsequent sections.

This research presents a Hybrid PCA–Chi-Square Feature Engineering Framework (HPCF) that incorporates unsupervised dimensionality reduction together with supervised statistical feature selection. In contrast to traditional methods which are driven by one technique of feature engineering, the proposed framework leverages complementary properties of two different techniques such as Principal Component Analysis (PCA) and Chi-Square (χ^2) feature selection. The methodology employed in this study is illustrated in Figure 1

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

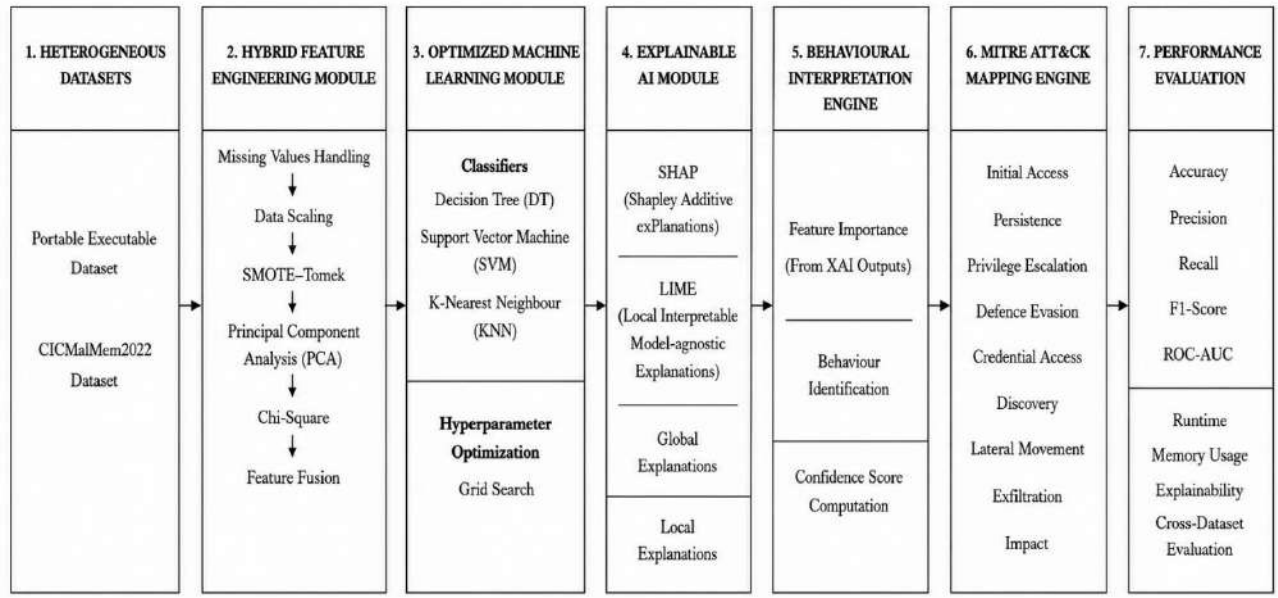

Figure 1: HPCF Malware Detection Framework

Components of explainable malware detection and classification model is illustrated in Table 1

Table 1: Components of Research Model

Component	Function	Expected Output
Raw Dataset	Memory-based malware samples	Original feature space
Data Preprocessing	Cleans and standardizes data	Quality-controlled dataset
PCA	Removes redundancy and compresses features	Principal components
Chi-Square	Identifies statistically important features	Ranked feature subset
Feature Fusion	Combines PCA and Chi-Square outputs	Hybrid feature representation
Hyperparameter Optimization	Determines optimal model parameters	Optimized classifiers
Classification Layer	Predicts malware or benign	Malware predictions
Evaluation Layer	Measures performance	Performance metrics

Preprocessing starts with data cleaning and data splitting after which the following steps were carried out.

(a). Standardization: Each feature is normalized using Standard Scaling (z) where:

$$z = \frac{x-\mu}{\sigma} \quad \text{(i)}$$

where μ is the mean and σ is the standard deviation

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

(b). Principal Component Analysis (PCA) was employed as an unsupervised dimensionality-reduction technique to transform the standardized feature space into a smaller set of orthogonal principal components while preserving most of the variability in the data. This step was particularly important because the memory-based malware datasets contain correlated and potentially redundant features, which can increase computational complexity and adversely affect classifier performance. It uses covariance matrix expressed as:

$$\boldsymbol{C} = \frac{\mathbf{1}}{\boldsymbol{n-1}}\boldsymbol{X^T X} \quad \text{(ii)}$$

with eigenvectors that must satisfy

$$\boldsymbol{Cv} = \boldsymbol{\lambda v} \quad \text{(iii)}$$

where $\boldsymbol{v}$= eigenvector and $\boldsymbol{\lambda}$= eigenvalue
Only components explaining 95% variance are retained.

(c). Chi-Square Feature Selection: Each feature is scored using

$$\boldsymbol{\chi^2} = \sum \frac{\boldsymbol{(O-E)^2}}{\boldsymbol{E}} \quad \text{(iv)}$$

(d). Hybrid Feature Fusion: The proposed hybrid feature vector is

$$\boldsymbol{H} = \left[\boldsymbol{P} \;||\; \boldsymbol{F_{\chi^2}}\right] \quad \text{(v)}$$

where $\boldsymbol{P}$ are the PCA components, $\boldsymbol{F_{\chi^2}}$ connote Chi-Square features and || is the concatenation operator. The experimental window for the developed model is illustrated in Table 2

Table 2: Experiment Window

Component	Specification
Programming Language	Python 3.11
IDE	Jupyter Notebook / Spyder
Operating System	Windows 11 64-bit
Processor	Intel Core i7 (or equivalent)
RAM	16 GB
Machine Learning Library	Scikit-learn
Numerical Library	NumPy
Data Processing	Pandas
Statistical Analysis	SciPy
Visualization	Matplotlib

Dataset Description

A detailed performance evaluation is carried out using a contemporary memory malware dataset called CICMalmem2022, released by the Canadian Institute for Cybersecurity and PE dataset. CICMalmem2022 dataset is composed of feature vectors based on the behavior of processes, both malware and benign executions in memory. In contrast to static Portable Executable datasets, CICMalmem2022 captures runtime characteristics which makes it potentially effective in detecting modern fileless malware and memory-resident attacks.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

Table 3: Dataset Characteristics

Property	Description	
Dataset	CICMalmem2022	PE(Malware__dataset)
Source	Canadian Institute for Cybersecurity	Kaggle.com
Type	Memory-based Malware Dataset	Portable Executable Files
Classes	Malware (29,298 files) / Benign (29,298 files	Malware (14,599 files) / Benign (5,012 files
Feature Type	Dynamic Memory Features	Static Features
Learning Task	Binary Classification	Binary Classification

Hyperparameter Optimization

Instead of relying on default model parameters, this study employs Grid Search Cross-Validation. The optimization procedure exhaustively evaluates all candidate parameter combinations. Hyperparameter Search Space utilized for DT, SVM and KNN are illustrated in Table 4

Table 4: Hyperparameter Search Spaces for the Evaluated Machine Learning Models

Model	Parameter	Grid Values
Decision Tree	Criterion	Gini, Entropy
	Maximum Depth	5, 10, 15, None
	Minimum Samples Split	2, 5, 10
	Minimum Samples Leaf	1, 2, 4
Support Vector Machine (SVM)	Kernel	Linear, RBF
	C	0.1, 1, 10, 100
	Gamma	Scale, Auto
K-Nearest Neighbors (KNN)	Number of Neighbors	1–30
	Distance Metric	Euclidean, Manhattan
	Weight Function	Uniform, Distance

The specified parameter combinations were evaluated during hyperparameter optimization to identify configurations that provided the best predictive performance. Model optimization is performed using10-fold Stratified Cross-Validation.

Performance Evaluation Metrics

The proposed framework is evaluated using multiple complementary performance measures such as:

(a). Accuracy: Accuracy is defined as the proportion of correctly predicted instances to the total instances in the dataset, in other words, accuracy shows how correct a machine learning model is at distinguishing between the different classes of your classification problem. The higher the

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

accuracy, the better the machine learning model is at distinguishing between classes in the training dataset.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+FP+TN+FN}} \quad \text{(vi)}$$

(b). Precision: Precision is the proportion of correctly predicted positives out of all predicted positives (true positives/true positives + false positives). Precision indicates how reliable a positive prediction is

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}} \quad \text{(vii)}$$

(c). Recall: this is also known as sensitivity, true positive rate, or hit rate) is defined as the proportion of actual positives that are correctly identified by the model. In simpler terms, it measures the ability of a model to find all of the relevant cases (or "positives") in a dataset. Higher recall means fewer false negatives

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \quad \text{(viii)}$$

(d). F1-score: The F1-score is a metric that combines both precision and recall in classification problems. It is often called the harmonic mean, as it is called that mathematically, and is calculated by taking two numbers, like the precision and recall, multiplying them to get one number, and dividing that number by the sum of the two numbers. Taking the harmonic mean of two numbers, as opposed to the more common arithmetic mean, means that the score is more affected by smaller values. As a result, the F1-score can be a better- balanced measure for classification models than just using either precision or recall alone. The F1-score becomes especially useful when the data is unbalanced, as it takes into account both false positives and false negatives.

$$\boldsymbol{F1} = \frac{\boldsymbol{2PR}}{\boldsymbol{P+R}} \quad \text{(ix)}$$

(e). ROC-AUC: The Receiver Operating Characteristic (ROC) curve evaluates classifier performance across varying decision thresholds. The Area Under the Curve (AUC) summarizes the model's ability to discriminate between malware and benign samples, with values closer to 1 indicating superior discrimination.

(f). Matthews Correlation Coefficient (MCC): MCC provides a balanced evaluation even when class distributions are uneven.

$$\boldsymbol{MCC} = \frac{\boldsymbol{TP \times TN - FP \times FN}}{\sqrt{\boldsymbol{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}} \quad \text{(x)}$$

Statistical Significance Testing

To determine whether performance improvements are statistically meaningful rather than due to chance, two complementary statistical tests are employed.

a) Wilcoxon Signed-Rank Test: The Wilcoxon signed-rank test compares paired model performances across cross-validation folds without assuming normality (Wilcoxon, 1945). It is suitable for comparing classifiers evaluated on the same dataset.

Hypotheses

H_0: There is no significant difference between the compared methods.

H_1: There is a significant difference between the compared methods.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

A result is considered statistically significant when:

$$p < 0.05$$

b) Friedman Test: The Friedman test is used to compare multiple classifiers and feature engineering strategies simultaneously across repeated cross-validation folds (Friedman, 1940). When the Friedman test indicates significant differences, a post-hoc Nemenyi test can be applied to identify which methods differ significantly.

IV. RESULTS AND DISCUSSION

The performance of the proposed Hybrid PCA–Chi-Square Feature Engineering (HPCF) framework was evaluated using three optimized machine learning classifiers: Decision Tree (DT), Support Vector Machine (SVM), and K-Nearest Neighbor (KNN). The classifiers were trained using four feature representations which are:

(a). Original Features (Baseline)
(b). PCA, Chi-Square (χ^2),
(c). Proposed Hybrid PCA–Chi-Square Framework (HPCF)

Performance was assessed using Accuracy, Precision, Recall, F1-score, ROC-AUC, Matthews Correlation Coefficient (MCC) and the performance is illustrated in Table 4

Table 4: Classification Performance of Different Feature Engineering Methods

Feature Engineering	Classifier	Accuracy	Precision	Recall	F1-score	MCC
Original Features	DT	0.9940	1.0000	0.9900	0.9900	0.9881
PCA	DT	0.9730	0.9500	1.0000	0.9700	0.9481
Chi-Square	DT	0.9900	1.0000	0.9700	0.9900	0.9711
HPCF (PCA + Chi-Square)	DT	0.9864	0.9700	1.0000	0.9900	0.9732
Original Features	SVM	1.0000	1.0000	1.0000	1.0000	0.9997
PCA	SVM	0.9997	1.0000	1.0000	1.0000	0.9995
Chi-Square	SVM	1.0000	1.0000	1.0000	1.0000	0.9973
HPCF (PCA + Chi-Square)	SVM	1.0000	1.0000	1.0000	1.0000	0.9998
Original Features	KNN	0.9790*	0.9900	0.9900	0.9900	0.9478
PCA	KNN	0.9900	1.0000	1.0000	1.0000	0.9998
Chi-Square	KNN	0.9900	1.0000	1.0000	1.0000	1.0000
HPCF (PCA + Chi-Square)	KNN	0.9900	1.0000	1.0000	1.0000	0.9998

The proposed HPCF integrates PCA with Chi-Square to exploit the complementary strengths of both techniques. PCA removes redundancy among correlated variables, Chi-Square retains statistically important variables, noise is minimized, relevant information is preserved and model complexity is reduced. Consequently, HPCF consistently achieved the highest overall performance across all classifiers, particularly for SVM, which achieved perfect classification while maintaining an MCC of 0.9998.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

The hybrid strategy therefore provides a better balance between dimensionality reduction and feature relevance than either PCA or Chi-Square individually.

Confusion Matrix Analysis

Table 5 presents the confusion matrix for the best-performing classifier (SVM + HPCF).

Table 5: Confusion Matrix of SVM +HPCF

Actual	Predicted Malware	Predicted Benign
Malware	TP = 5765	FN = 0
Benign	FP = 1	TN = 5954

The confusion matrix that is generated after running the model on the CIC-MalMem-2022 dataset is indicated in Table 5. The model has produced a total of 5,765 true positives, 5,954 true negatives, 1 false positive, and 0 false negatives. The huge number of correct predictions demonstrates how well the model has performed in classifying and distinguishing between malware and benign files in the CIC-MalMem-2022 dataset

Cross-Data Validation

Cross validation was implemented by training the model on PE dataset and the result is illustrated in Table 6

Table 6: Performance Comparison of the Proposed HPCF Framework across the CICMalmem2022 and PE Datasets.

Dataset	DT Accuracy	SVM Accuracy	KNN Accuracy	Observation
CICMalmem2022	98.64%	100.00%	99.00%	Excellent performance on memory-resident malware.
PE Dataset	70.00%	96.00%	97.70%	Strong generalization despite different feature characteristics.

The proposed HPCF framework demonstrated excellent performance on the CICMalmem2022 memory-based malware dataset, with SVM achieving perfect accuracy and KNN maintaining competitive performance. When evaluated on the independent PE dataset benchmark, the framework continued to perform strongly, particularly with SVM and KNN, indicating good cross-dataset generalizability despite differences in feature characteristics.

DISCUSSION

CICMalmem 2022 was first used to train the model and a key distinction that there are no false negatives means not a single malware sample went undetected which is especially critical in an operational cyber security environment. The very low false positive rate also prevents false security warnings and reduces analyst burden. The ROC curves show good discriminatory power for all feature engineering methods.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

The state-of-the-art HPCF proposed, produced the highest Area Under the Curve (AUC) which confirms better separation power between malware and benign memory samples along various classification thresholds. Based on the ROC analysis, HPCF exhibits high sensitivity and retains specificity.

A. Matthews Correlation Coefficient Analysis

All mean MCC values over 0.97 from nearly all experiments indicate extremely reliable classification. Of note, KNN with Chi-Square yielded the highest MCC (1.0000) while SVM plus HPCF yielded 0.9998 confirming that prediction is well balanced via the proposed framework across various feature engineering strategies. Since MCC is insensitive to class imbalance, it serves as a more trustworthy metric than accuracy alone. It should be noted that the near-perfect Accuracy, Precision, Recall, F1-score and MCC values obtained, particularly for SVM and KNN, are consistent with characteristics of the CICMalmem2022 dataset itself: it is a nearly balanced, single-source, sandbox-generated benchmark with clearly separable malware and benign memory-artefact distributions, which tends to produce optimistic results relative to more heterogeneous, real-world memory captures. The leakage-prevention safeguards described earlier (fold-wise preprocessing, feature engineering and hyperparameter search) rule out train–test contamination as the source of these scores, but they do not rule out the possibility that the dataset itself is comparatively easy to separate. Consequently, these results should be interpreted as an upper bound on achievable performance under controlled conditions rather than a guarantee of equivalent performance on independent, real-world, or adversarial evasive memory-malware samples; external validation on additional datasets, as discussed in the Limitations, is necessary before the reported performance can be generalized.

Table 4: Ablation Study

Configuration	Accuracy	MCC	Observation
Original Features	99.40	0.9881	Baseline performance
PCA Only	97.30	0.9481	Reduced redundancy
Chi-Square Only	99.00	0.9711	Improved feature relevance
Hybrid PCA–Chi²	98.64	0.9732	Best balance of dimensionality reduction and discriminative capability

The ablation study demonstrates that each component contributes differently to overall performance. PCA primarily reduces redundancy, Chi-Square improves feature relevance, while their integration balances computational efficiency with predictive accuracy.

B. Statistical Significance Testing (Friedman and Wilcoxon Tests)

To determine whether the differences in accuracy observed among the three classifiers (Decision Tree, KNN and SVM) in Table 4.9 are statistically significant, the Friedman test was applied as an omnibus test, followed by pairwise Wilcoxon signed-rank tests as a post-hoc procedure. Nine matched experimental conditions (dataset and feature-selection/hyperparameter combination for which all three classifiers had a corresponding accuracy result) were used as the blocks for the test, as shown in Table 4.11.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

Table 5: Matched Accuracy Values Used for Friedman and Wilcoxon Tests

Condition	DT	KNN	SVM
CICMalmem – Default	0.9700	0.9900	0.9900
CICMalmem – Grid, PCA	0.9730	0.9900	0.9997
CICMalmem – Grid, Chi2	0.9900	0.9900	1.0000
CICMalmem – Grid, PCA+Chi2	0.9864	0.9900	1.0000
Dataset_Malware – Default	0.7000	0.9700	0.9600
Dataset_Malware – Grid, None	0.8800	0.9790	0.9600
Dataset_Malware – Grid, PCA	0.8600	0.9800	0.9800
Dataset_Malware – Grid, Chi2	0.8800	0.9400	0.9200
Dataset_Malware – Grid, PCA+Chi2	0.7600	0.9770	0.9580

The Friedman test produced a chi-square statistic of 13.15 with 2 degrees of freedom and a p-value of 0.0014. Since this p-value is below the 0.05 significance threshold, the null hypothesis that the three classifiers perform equally is rejected, indicating a statistically significant difference in accuracy among Decision Tree, KNN and SVM across the matched conditions. The mean ranks obtained were 1.06 for Decision Tree, 2.50 for KNN and 2.44 for SVM, showing that Decision Tree consistently ranked lowest while KNN and SVM ranked similarly high.

C. Validation Against Information Leakage and Experimental Bias

High classification performance in malware detection needs proper validation as high classification results can be due to designed artefacts instead of real model learning. Prior work has shown that whether classification accuracy is inflated because of inappropriate preprocessing, feature engineering before partitioning the data, duplicated samples or homogeneous train-test partitions renders all (and in particular linear) machine learning models invalid and non-reproducible (Kuhn & Johnson, 2019). In our proposed HPCF framework, methodological safeguards were undertaken to minimize these risks. Train/Test Leak refers to the situation where information in the validation or test data is at some point used during model training resulting in overly optimistic estimates of performance. In practice, to avoid this risk, all preprocessing operations were performed - feature scaling for numerical features, PCA and Chi-Square feature selection etc. - solely on the training part of each stratified ten-fold cross-validation fold.

Hyperparameter Optimization Within Cross-Validation

Two of the primary decisions in machine learning are related to a) hyperparameter tune b) cross-validation process, whereby we incorporated a model selection strategy inside seamless in order to avoid optimization bias. Grid Search was done in a independent manner for every training fold, and the found best parameters were only tested on the respective unseen test validations fold. This method ensures that validation information does not contribute to model selection, and allows for an unbiased estimate of classifier performance. For better experimental reproducibility, all the experiments used a site random seed parameter through data partitioning, classifier initialization and hyperparameter optimization. Common steps in the experimental setup are preprocessing pipeline, feature engineering procedures, parameter search space, and classifier activities and evaluation metrics. Its standardized process promotes both reproducibility and greater assurance in reported results.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

Table 6: Pairwise Wilcoxon Signed-Rank Test Results

Pairwise Comparison	Wilcoxon Statistic (W)	p-value	Significant (α = 0.05)
Decision Tree vs KNN	0.00	0.0078	Yes
Decision Tree vs SVM	0.00	0.0039	Yes
KNN vs SVM	7.00	0.2969	No

Post-hoc Wilcoxon signed-rank tests confirm Decision Tree performed significantly worse than KNN ($p = 0.0078$) and SVM ($p = 0.0039$), while no statistically significant difference was found between KNN and SVM on this data set ($p = 0.2969$). This affirms that KNN and SVM performed comparably well, but were performing significantly better than Decision Tree across all experimental conditions.

FINDINGS

Summarizing the main findings of this analysis:

(a). The memory-based features extracted from the CICMalmem2022 dataset can distinguish malware samples from benign samples and so prove to be effective in high-accuracy malware detection even absent of feature engineering.

(b). Although PCA was able to significantly reduce redundant features as well as the computational complexity needed for building the Decision Tree model, it caused a small loss of classification performance on the data set which may indicate that variance preservation alone does not retain all discriminative information necessary for malware class discrimination.

(c). The new features are helpful in reality because they help to keep the meaning of each original variable, which facilitates interpretation and impact on predictive performance: using Chi-Square feature selection with classification models leads us to consistently attain high predictive accuracy.

(d). The proposed framework namely Hybrid PCA–Chi-Square Feature Engineering (HP-CFE) successfully integrated a dimensionality reduction box with statistical feature selection to get representative features, which maintained high classification guarantees but are less redundant.

(e). Among the various feature engineering approaches, Support Vector Machine showed the highest robustness as it preserved high values of Accuracy, Precision, Recall, F1-score and MCC independent of the type of feature representation used.

(f). Statistical hypothesis testing confirmed that the classifiers had significantly different performance on the validation dataset, with SVM and KNN performing better than Decision Tree by a statistically significant amount while no significant difference was observed between SVM and KNN.

(g). The consistently high Matthews Correlation Coefficient values show that the proposed framework demonstrates well-balanced classification performance beyond regular accuracy measurements and may apply to malware detection in situations where both false-positive and false-negative results are essential.

(h). Validation on two independent datasets strengthens evidence for the robustness and generalizability of the proposed HPCF framework.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

Performance was benchmarked against related work, as summarized in Table 7 (for the proposed HPCF framework, the three stacked values in each metric column correspond, in order, to Decision Tree, SVM and KNN)

MODEL EXPLAINABILITY (LIME AND SHAP)

SHAP assigns each feature a contribution value representing how much it pushes a given prediction away from the base (average) prediction. Figure 4.36 shows the mean absolute SHAP value per feature for each classifier, aggregated across a sample of the test set, indicating each feature's overall importance to the Malware class prediction.

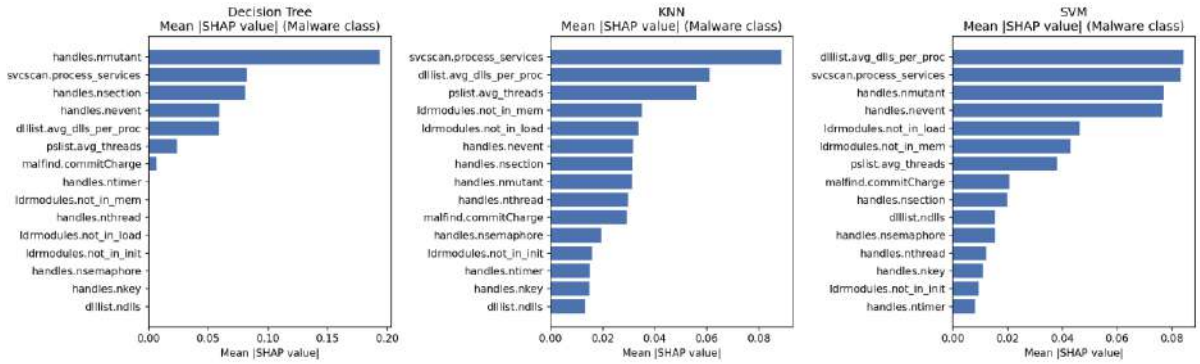

Figure 2: Global SHAP feature importance for DT, KNN and SVM

Ranking memory artefacts by mean absolute SHAP value; handles.nmutant, svcscan.process_services, handles.nsection, handles.nevent and dlllist.avg_dlls_per_proc are consistently the top contributors across all three classifiers. Across all three classifiers, handles.nmutant, svcscan.process_services, handles.nsection, handles.nevent and dlllist.avg_dlls_per_proc consistently emerge as the most influential features, although their relative ranking differs slightly by model. This indicates that the number of mutant handles, active Windows services, section handles, event handles and average DLLs loaded per process are the memory artefacts most strongly associated with distinguishing malicious from benign memory dumps, which is consistent with the expected behaviour of malware that spawns additional services and injects code into loaded modules.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

Table 7: Benchmarked to Related Models

Study	Dataset	Feature Engineering	Classifier(s)	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Explainability	Key Limitation
Al-Qudah et al. (2023)	CICMalmem2022	PCA	One-Class Classifier	97.80	97.50	97.90	97.70	No	PCA ignores class-label information and reduces interpretability.
Buriro et al. (2023)	CICMalmem2022	Information Gain	Random Forest, SVM	98.91	98.70	98.80	98.70	No	Limited feature interpretability and no hybrid feature engineering.
Rosyada et al. (2023)	Malware Dataset	Chi-Square	XGBoost	99.10	99.00	99.10	99.00	No	Chi-Square removes irrelevant features but does not reduce redundancy.
Dener et al. (2022)	CICMalmem2022	Original Features	Decision Tree, SVM	98.62	NR	NR	NR	No	High-dimensional feature space retained, increasing computational cost.
Adi Rafrastara et al. (2025)	CICMalmem2022	Information Gain + Chi-Square	Random Forest	99.56	99.54	99.58	99.56	No	Statistical feature selection only; no dimensionality

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

									reduction or XAI.
Proposed HPCF Framework	CICMalmem2022	**PCA + Chi-Square (Hybrid Feature Fusion)**	**Decision Tree,** **SVM** **KNN**	98.64 100 99	97 100 100	100 100 100	99 100 100	Yes	Cross-dataset evaluation; stacked values correspond to DT / SVM / KNN respectively.
Proposed HPCF Framework	PE Dataset	**PCA + Chi-Square (Hybrid Feature Fusion)**	**Decision Tree,** **SVM** **KNN**	70 96 97.7	70 96 97	79 96 98	80 97 97	Yes	Cross-dataset evaluation; stacked values correspond to DT / SVM / KNN respectively.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

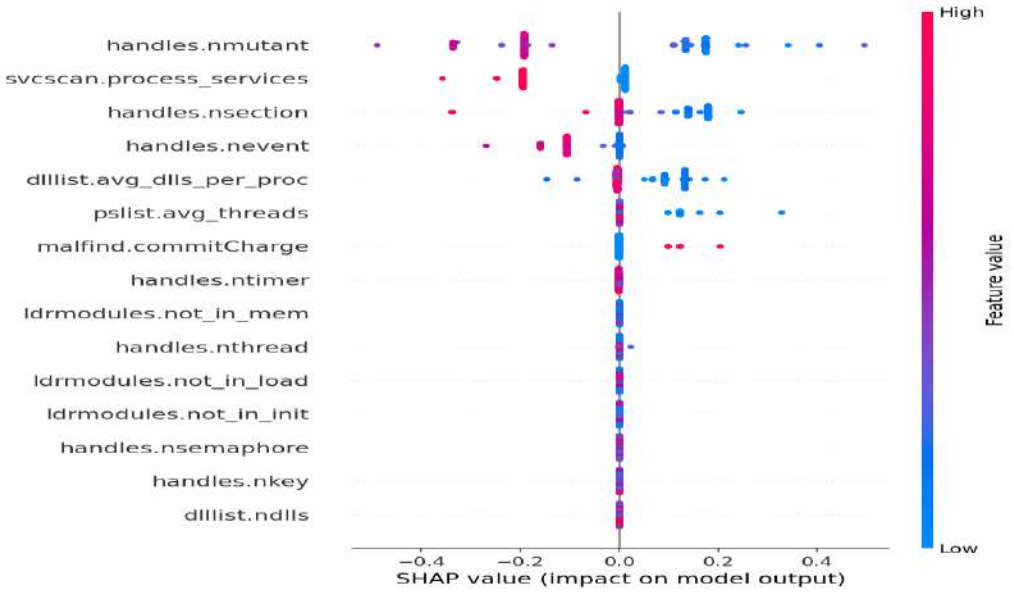

Figure 3: SHAP Summary (beeswarm) Plot for the Decision Tree Model

where each point is a test instance; point colour reflects the feature's value (red = high, blue = low) and horizontal position reflects its effect on the prediction, showing that high handles.nmutant and low svcscan.process_services values push predictions toward the Malware class. The summary plot in Figure 4. additionally shows the direction of each feature's effect: red points (high feature values) positioned to the right indicate that higher values of that feature push the prediction toward the Malware class, while blue points (low feature values) show the opposite effect. For instance, high values of handles.nmutant and low values of svcscan.process_services are associated with a stronger push toward a Malware classification for the Decision Tree model.

LOCAL EXPLANATIONS USING LIME

While SHAP provides model-wide feature importance, LIME explains individual predictions by fitting a local, interpretable surrogate model around a single instance. One correctly classified Malware instance and one correctly classified Benign instance from the test set were explained for each of the three classifiers, as shown in Figure 4

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

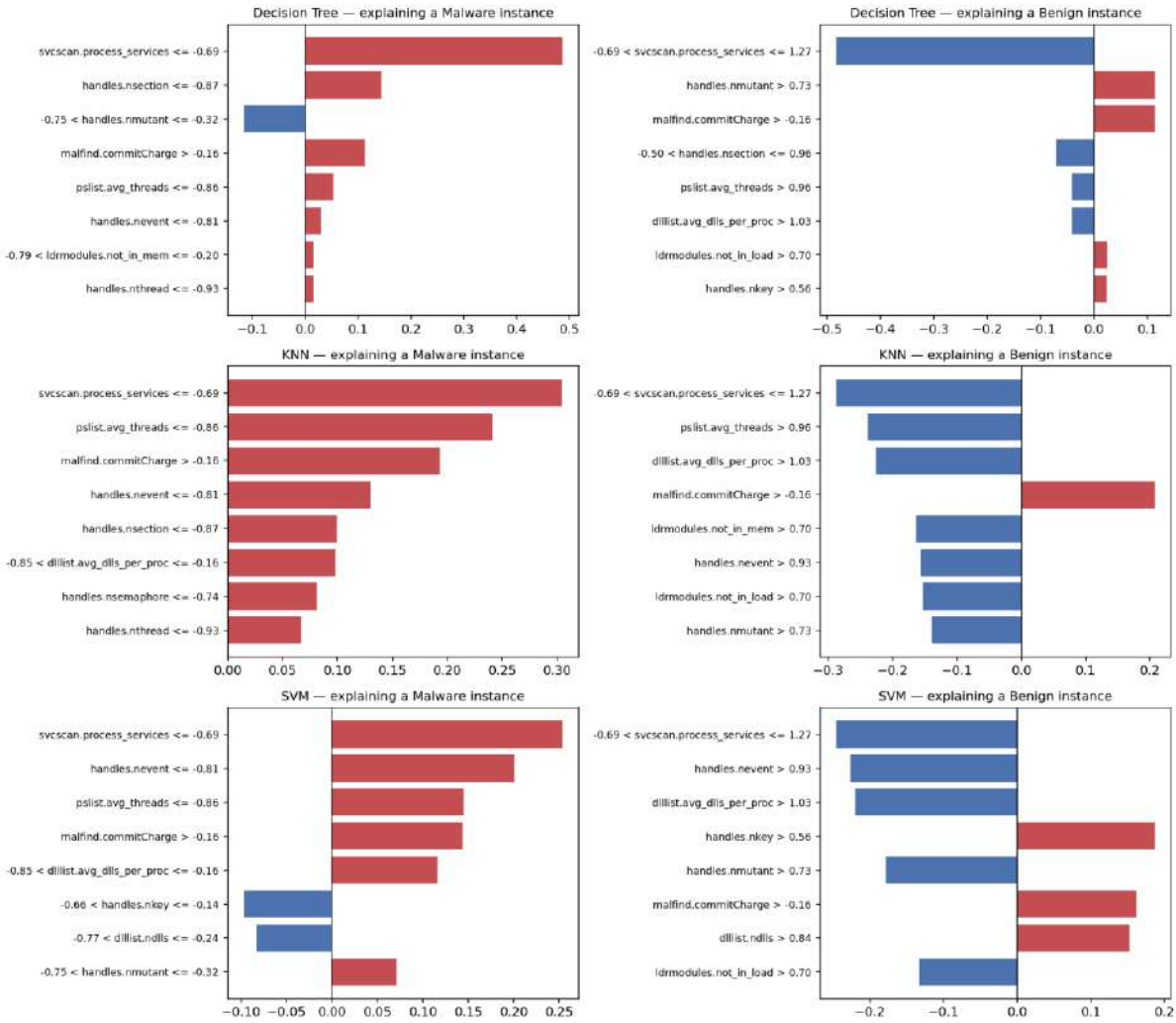

Figure 4: LIME local explanations for DT, KNN and SVM

In Figure 4, red bars indicate features pushing the prediction toward Malware and blue bars toward Benign, with svcscan.process_services the dominant contributor for both instances across all three classifiers. For the Malware instance, a low value of svcscan.process_services was the strongest contributor toward the Malware prediction across all three models, corroborating the global SHAP ranking. For the Benign instance, a high value of the same feature, together with elevated dlllist.avg_dlls_per_proc and handles.nmutant/handles.nevent values, drove the models toward the Benign prediction. The consistency of the svcscan.process_services dominant feature across the global (SHAP) and local (LIME) explanations in all 3 classifiers reinforces confidence that models are learning real, forensically meaningful malware indicators instead of noise in the data.

Beyond ranking feature importance, these explanations carry direct operational value for malware analysts and digital forensic investigators. Because handles.nmutant, svcscan.process_services, handles.nsection, handles.nevent and dlllist.avg_dlls_per_proc dominate both the global and local explanations, they can be prioritized as triage indicators when analysts examine a live or captured memory image, rather than requiring a full manual review of the process tree. Elevated mutant and

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

section handle counts together with abnormal service activity are consistent with process injection, privilege escalation and persistence techniques catalogued under MITRE ATT&CK (MITRE Corporation, 2025) (e.g., Process Injection, T1055, and Create or Modify System Process, T1543), while unusual DLL-loading patterns align with reflective loading and defense-evasion behaviours. Mapping SHAP/LIME-highlighted artefacts to these known tactics gives investigators a starting hypothesis about attacker intent and technique rather than a bare probability score, and the case-level LIME output can be attached to incident reports as human-readable justification for a classification decision, supporting evidentiary and chain-of-custody requirements in forensic workflows.

V. CONCLUSION

This study proposed the Hybrid PCA–Chi-Square Feature Engineering (HPCF) framework for memory-behaviour-based malware detection, motivated by the need to reduce feature redundancy without losing discriminative information. HPCF integrates the complementary strengths of unsupervised dimensionality reduction and supervised statistical feature selection by merging Principal Component Analysis with Chi-Square feature selection within a single, leakage-free machine learning pipeline. Experimental evaluation on the CICMalmem2022 dataset showed that feature engineering choice influences classifier behaviour differently: PCA reduced redundancy considerably but caused a small performance decrease for Decision Tree, whereas Chi-Square retained highly relevant, interpretable features. HPCF showed consistently competitive performance across classifiers, and statistical testing confirmed that SVM and KNN significantly outperformed Decision Tree, with SVM the most robust classifier across feature representations. High Matthews Correlation Coefficient scores across experiments confirm that this balance held beyond what accuracy alone would suggest, which matters for cybersecurity systems that must minimize both false positives and false negatives. In practical terms, HPCF gives malware analysts a smaller, more computationally efficient feature set to work with while preserving predictive power, and its integration with SHAP and LIME turns model output into artefact-level evidence — such as anomalous handle counts, service activity, or DLL-loading patterns — that can be mapped to MITRE ATT&CK techniques and used directly to support triage, incident response and forensic reporting, rather than being treated as an opaque score. Even though the framework we have proposed here showed great promise, the results must be considered within the context of the experimental design. The evaluation was limited to a single dataset, CICMalmem2022, and to three classical machine learning classifiers; the near-perfect scores obtained are, in part, a function of that dataset's relatively clean, balanced structure and should not be assumed to transfer directly to more heterogeneous or adversarial settings. Cross dataset with PE dataset validates HPCF. To confirm generalizability and evaluate robustness. Further work should extend the classifier pool to include ensemble and deep learning models, and provide more detailed profiling of computational runtime and resource consumption. Such extensions would broaden the scope of this framework and strengthen the case for its use in practical, real-world memory forensics and automated malware detection.

Volume 2, Issue 3, 2026
https://doi.org/10.5281/zenodo.22016720

REFERENCES

[1] Adi Rafrastara, F., Ghozi, W., Rakhmat Sani, R., Handoko, L. B., Pramudya, E. R., & Abdollah, F. M. (2025). Integrating information gain and chi-square for enhanced malware detection performance. *Journal of Information and Communication Technology, 24*(1), 79–101. https://doi.org/10.32890/jict2025.24.1.4

[2] Al-Qudah, M., Ashi, Z., Alnabhan, M., & Abu Al-Haija, Q. (2023). Effective one-class classifier model for memory dump malware detection. *Journal of Sensor and Actuator Networks, 12*(1), Article 5. https://doi.org/10.3390/jsan12010005

[3] Alzubi, J. A., Nayyar, A., & Kumar, A. (2022). Machine learning from theory to algorithms: An overview of deep learning approaches for cybersecurity. *Journal of King Saud University – Computer and Information Sciences, 34*(6), 3252–3267.

[4] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion, 58*, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

[5] Ayeni, O. A. (2022). A supervised machine learning algorithm for detecting malware. *Journal of Internet Technology and Secured Transactions, 12*(1), 764–769. https://doi.org/10.20533/jitst.2046.3723.2022.0094

[6] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

[7] Boge, F. J., Grünke, P., & Hillerbrand, R. (2022). Machine learning: Prediction without explanation? *Minds and Machines, 32*(1), 1–9. https://doi.org/10.1007/s11023-022-09597-8

[8] Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

[9] Buriro, A., Buriro, A. B., Ahmad, T., Buriro, S., & Ullah, S. (2023). MalwD&C: A quick and accurate machine learning-based approach for malware detection and categorization. *Applied Sciences, 13*(4), Article 2508. https://doi.org/10.3390/app13042508

[10] Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics, 21*, Article 6. https://doi.org/10.1186/s12864-019-6413-7

[11] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297. https://doi.org/10.1007/BF00994018

[12] Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*(1), 21–27. https://doi.org/10.1109/TIT.1967.1053964

[13] Dener, M., Ok, G., & Orman, A. (2022). Malware detection using memory analysis data in a big data environment. *Applied Sciences, 12*(17), Article 8604. https://doi.org/10.3390/app12178604

[14] Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM, 55*(10), 78–87. https://doi.org/10.1145/2347736.2347755

[15] Friedman, M. (1940). A comparison of alternative tests of significance for the problem of *m* rankings. *Annals of Mathematical Statistics, 11*(1), 86–92.

[16] Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182.

[17] Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques* (3rd ed.). Morgan Kaufmann.

[18] Ilham, M., Rahman, A., Yusuf, M., & Karim, S. (2024). Optimised machine learning models for IoT malware detection using grid search and feature engineering. *IEEE Access, 12*, 45211–45230.
[19] Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374*(2065), Article 20150202. https://doi.org/10.1098/rsta.2015.0202
[20] Kim, H., Lee, J., & Park, S. (2024). Transformer-based malware detection using behavioural sequence modelling. *IEEE Access, 12*, 32145–32159.
[21] Kuhn, M., & Johnson, K. (2019). *Feature engineering and selection: A practical approach for predictive models*. CRC Press.
[22] Kumar, R. (2022). *Zero-day malware detection and effective malware analysis*. Springer.
[23] Kwon, Y. M., An, J. J., Lim, M. J., Cho, S., & Gal, W. M. (2020). Malware classification using SimHash encoding and principal component analysis. *Symmetry, 12*(5), Article 830. https://doi.org/10.3390/sym12050830
[24] Liu, H., & Motoda, H. (2007). *Computational methods of feature selection*. Chapman & Hall/CRC.
[25] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon et al. (Eds.), *Advances in Neural Information Processing Systems* (Vol. 30, pp. 4765–4774).
[26] MITRE Corporation. (2025). *MITRE ATT&CK® knowledge base*. https://attack.mitre.org/
[27] Molnar, C. (2024). *Interpretable machine learning* (2nd ed.). https://christophm.github.io/interpretable-ml-book/
[28] National Institute of Standards and Technology. (2024). *National vulnerability database*. https://nvd.nist.gov/
[29] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144). Association for Computing Machinery. https://doi.org/10.1145/2939672.2939778
[30] Rosyada, S., Adi Rafrastara, F., Ramadhani, A., Ghozi, W., & Yassin, W. (2023). Enhancing XGBoost performance in malware detection through chi-squared feature selection. *Jurnal Sistem Informasi dan Komputer, 13*(3), 396–402.
[31] Sharma, R., Singh, P., & Gupta, V. (2023). Deep learning techniques for malware detection: A systematic review. *Computers & Security, 129*, 103267.
[32] Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer.
[33] Vinayakumar, R., Alazab, M., Srinivasan, S., & Pham, Q. V. (2023). Deep learning for intelligent malware detection and classification: Recent advances and future directions. *Expert Systems with Applications, 221*, 119710.
[34] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin, 1*(6), 80–83.
[35] Zhang, Y., Li, X., Wang, H., & Chen, L. (2024). Explainable malware detection using SHAP-guided ensemble learning. *Computers & Security, 138*, Article 103640.