UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Posgrados

Feature Selection in Malware Classification

Proyecto de Titulación

Bryan Esteban Calisto Yerovi

Felipe Grijalva, Ph.D.

Director de Trabajo de Titulación

Trabajo de titulación de posgrado presentado como requisito para la obtención del título de Magíster en Inteligencia Artificial

Quito, 01 de diciembre 2024

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ
COLEGIO DE POSGRADOS

HOJA DE APROBACIÓN DE TRABAJO DE TITULACIÓN

Feature Selection in Malware Classification

Bryan Esteban Calisto Yerovi

Nombre del Director del Programa:	Felipe Grijalva
Título académico:	Ph.D. en Ingeniería Eléctrica
Director del programa de:	Inteligencia Artificial

Nombre del Decano del colegio Académico:	Eduardo Alba
Título académico:	Doctor en Ciencias Matemáticas
Decano del Colegio:	Ciencias e Ingenierías

Nombre del Decano del Colegio de Posgrados:	Dario Niebieskikwiat
Título académico:	Doctor en Física

Quito, diciembre 2024

© DERECHOS DE AUTOR

Nombre del estudiante:	Bryan Esteban Calisto Yerovi
Código de estudiante:	00339249
C.I.:	1500932445
Lugar y fecha:	Quito, 01 de diciembre 2024

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following graduation project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

DEDICATORIA

Dedico este trabajo a mi familia y a todas y todos quienes que me inspiraron en esta aventura académica

AGRADECIMIENTOS

Ofrezco mis agradecimientos a mi tutor Felipe Grijalva y los autores de investigaciones que me han servido como pilares para desarrollar este trabajo y a las que, por supuesto, doy el credito en este trabajo.

RESUMEN

Los ataques de malware han aumentado en cantidad y calidad en los últimos años, lo que cuestiona la necesidad de mejorar las técnicas que utilizan los defensores para intentar hacer frente a las innovaciones y esfuerzos de los atacantes. El aprendizaje automático ya se ha aplicado en este campo: detección y clasificación de malware, detección de intrusiones basada en anomalías y análisis de amenazas, solo por nombrar algunas aplicaciones. Este documento se centra en la clasificación de malware, explorando el trabajo previo en el conjunto de datos CIC-MalMem-2022 para construir dos clasificadores de malware: 4 clases y 16 clases, donde las 16 clases son subclases de las 4 clases, mezclando varios clasificadores con varias técnicas de selección de características que dieron como resultado herramientas de preprocesamiento efectivas para el conjunto de datos. Los resultados obtenidos demostraron que los algoritmos de eliminación de características recursivas, información mutua y Boruta son técnicas de selección de características efectivas para este conjunto de datos, lo que permite la eliminación de características redundantes, favoreciendo la eficiencia y la simplicidad del modelo e incluso aumentando las puntuaciones de las métricas de clasificación como F1. También realizamos una prueba de hipótesis para validar que existe una diferencia real entre nuestros resultados con selección de características y las versiones sin selección de características. Finalmente, ejecutamos un experimento haciendo predicciones con los modelos de 16 clases, agrupando las probabilidades resultantes por clase padre y obteniendo las métricas de clasificación comparando los resultados con los obtenidos con las 4 clases padre originales. El objetivo es verificar si podemos ganar rendimiento en la clasificación utilizando una capa complementaria de inferencia que ve los datos desde una perspectiva diferente, más atómica.

Palabras clave: Ciberseguridad, clasificación de malware, algoritmos de selección de características.

ABSTRACT

Malware attacks have increased in quantity and quality in the last years, questioning the necessity of enhancing the techniques that defenders use to try to cope with attackers innovations and efforts. Machine learning has already been applied in this field: malware detection and classification, anomaly-based intrusion detection and threat analysis, just to name a few applications. This document focuses on malware classification, exploring upon previous work on the CIC-MalMem-2022 dataset to build two malware classifiers - 4-class and 16-class, where the 16 classes are sub-classes of the 4-class - mixing various classifiers with various feature selection techniques which resulted in effective pre-processing tools for the dataset. The obtained results demonstrated that the Recursive Feature Elimination, Mutual Information and Boruta algorithms are effective feature selection techniques for this dataset, allowing redundant feature elimination, favoring efficiency and model simplicity and even increasing scores of classification metrics like F1. We also perform hypothesis test to validate there is actual difference between our feature selection based results and the all-features versions. Finally, we run an experiment by doing predictions with the 16-class models, grouping the resulting probabilities by parent class and obtaining the classification metrics for them with the test data and comparing them with the 4 parent classes'. The goal is to see if we can gain classification performance by using a complimentary layer of inference that sees the data in a different, more atomic perspective.

Key words: Cybersecurity, Malware Classification, Feature Selection Algorithms

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Feature Selection in Malware Classification

Bryan Calisto, Felipe Grijalva

***Abstract*—Malware attacks have increased in quantity and quality in the last years, questioning the necessity of enhancing the techniques that defenders use to try to cope with attackers innovations and efforts. Machine learning has already been applied in this field: malware detection and classification, anomaly-based intrusion detection and threat analysis, just to name a few applications. This document focuses on malware classification, exploring upon previous work on the CIC-MalMem-2022 dataset to build two malware classifiers - 4-class and 16-class, where the 16 classes are sub-classes of the 4-class - mixing various classifiers with various feature selection techniques which resulted in effective pre-processing tools for the dataset. The obtained results demonstrated that the Recursive Feature Elimination, Mutual Information and Boruta algorithms are effective feature selection techniques for this dataset, allowing redundant feature elimination, favoring efficiency and model simplicity and even increasing scores of classification metrics like F1. We also perform hypothesis test to validate there is actual difference between our feature selection based results and the all-features versions. Finally, we run an experiment by doing predictions with the 16-class models, grouping the resulting probabilities by parent class and obtaining the classification metrics for them with the test data and comparing them with the 4 parent classes'. The goal is to see if we can gain classification performance by using a complimentary layer of inference that sees the data in a different, more atomic perspective.**

***Index Terms*—Cybersecurity, Malware Classification, Feature Selection Algorithms.**

I. Introduction

THE cybersecurity field is a broadly explored field where creativity primes on the attacker and defender sides. Systems are continuously enhanced to decrease their susceptibility to being compromised by attackers; defenders always are trying to innovate to protect lifelong flaws in fundamental software. Attackers do the opposite, aiming to bypass software protections to get ownership of data or entire systems for their own interests. The malicious software that attackers use is generically called *malware*. There are a lot of types of malware: *rootkits*, *spyware*, *viruses*, worms, *ransomware* and *trojan horses*, just to name a few [1]. Each type has its own common characteristics, but some characteristics are shared too. Some are destructive, some favor stealthy, some are intelligent, some are deceptive, some are very clever, some are very hard to get rid of, some are hard to detect and practically invisible [2]. There is a broad range of techniques and philosophies that lead the development of malware together with economic, social, political and a variety of objectives. Every now and then a brand new technique is developed and the defenders must be able to respond with protections against those threats as soon and as effectively as possible. Their job can easily become overwhelming as bad behavior patterns can be hard to detect in malware that has been intentionally built to make difficult its detection and the detectors themselves can be less effective when using only conventional techniques like signature checking because it is trivial for the motivated attacker to generate dynamic clever malware that bypasses these kind of static protections, specially in expert systems manually setup for this task by programmers [1].

In recent years, machine learning has been used as a means to detect and categorize malware in order to determine strategies to apply against it [3]. An approach to evaluate the characteristics of a piece of running software is to capture a memory dump of it to obtain resource usage information (e.g. network sockets, file handles, threads), sub-processes information and any other aspect that's available through the operating system's API to collect. Then, those features can be fed into a machine learning model that can classify the software as a sample that belongs to a specific malware category or as benign [4].

This research elaborates upon previous work on the CIC-MalMem-2022 dataset done by Cevallos-Salas et al. [4]. We build two classifiers: a 4-categories and a 16-families classifier. Both are trained by using SMOTE to augment data inspired by Cevallos-Salas et al. [4] demonstrated effectiveness. We introduce the usage of multiple feature selection methods to pre-process the data with the goal of getting rid of possible highly correlated and unimportant features to save memory resources, enhance classification performance metrics and improve training and inference times. Statistical hypothesis tests are performed to compare the all-features versions against each feature selection version. The experiments are run with three classification algorithms: a fully connected neural network, a support vector machine and a random forest algorithm.

II. Prior Works

Multiple machine learning techniques have been applied to the CIC-MalMem-2022 dataset.

Cevallos-Salas, et al. [4] use classic machine learning algorithms like Decision Trees, Random Forest and Support Vector Machines, as well as artificial neural networks. They use SMOTE for data balancing achieving increased accuracy values [5].

Talukder, et al. [6] use a combination of machine and deep learning algorithms together with pre-processing techniques like feature selection with XGBoost with the

goal of eliminating superfluos features, reducing dimensions and complexity and improving the models accuracy overall to surpass other approaches. They use SMOTE too [5].

Kraskov, et al. [7] describe a filter feature selection method based on mutual information that considers the joint and marginal probability distributions of two random variables to measure the amount of information obtained from one random variable by observing another random variable.

Guyon, et al. [8] show a feature selection method that wraps the estimator and recursively fits and re-fits it, ranking the features and eliminating the less important one (highest rank) after each fit. This process repeats until an predetermined number of features is reached.

Jankowski, et al. [9] present the boruta algorithm. It is based on random forest and it works by creating what they call a "shadow feature" for each original feature. The shadow features are copies of the existing features, but shuffled between rows. Then the wrapped model is fit to both versions, original and shadow-merged, feature importance is calculated and the importance of each original feature is compared with the highest importance from the shadow features; if the original feature has higher score, it is kept. These comparisons are made with hypothesis testing. The algorithm iterates until all the features have been evaluated or until a predetermine amount of iterations.

LASSO (Least Absolute Shrinkage and Selection Operator) is a commonly known regularization method that is able to set coefficients to zero due to the constraints it puts into the conventional residual squared error calculations by adding the L1 norm of the model coefficients, penalizing large coefficients [10].

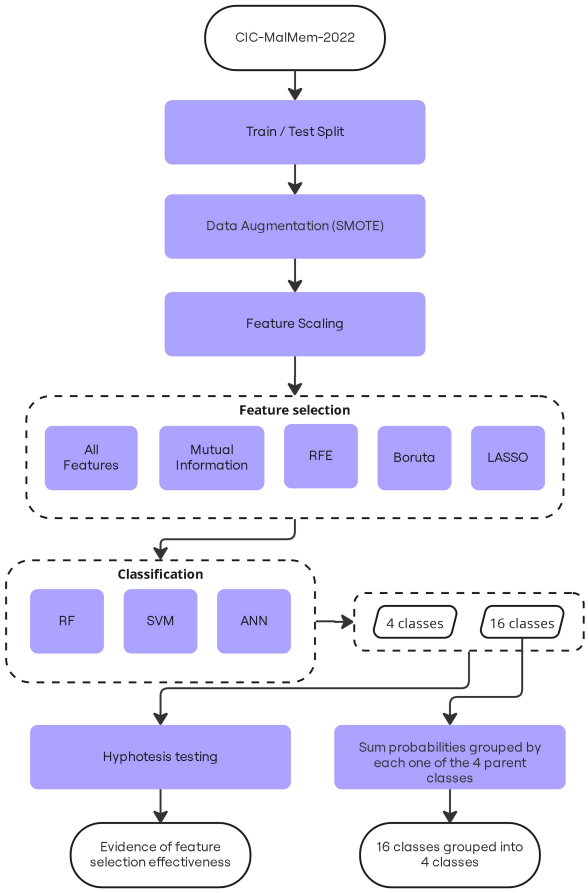

Figure 1. Experiments pipeline block diagram

III. Materials and Methods

A. Dataset

The dataset we use is CIC-MalMem-2022 [11]. It was authored by researchers from the University of Canada. It consists of 58,596 records with 29,298 benign and 29,298 malicious; the malicious subset is formed by records corresponding to Trojan Horse, Spyware and Ransomware types. More details about the malware samples used to build the dataset is detailed in Table I. The features consist of mostly numerical data related to the memory dumps that were taken from running pieces of software (benign and malicious) in a controlled environment running Windows 10. Later, the memory dumps would be processed with the VolMemLyzer tool [12] in a Kali Linux environment to extract the final features from them.

The data contains 57 features where the Class and Category fields determine whether the software is benign or malicious and the corresponding family it belongs to, respectively. Some of the numerical features present in the dataset are pslist.nproc (amount of subprocesses), pslist.avg_threads (average threads in the process), pslist.nprocs64bit (amount of 64 bit subprocesses), pslist.avg_handlers (average handlers, i.e. open resources like files, ports, etc), dlllist.ndlls (amount of DLLs), dlllist.avg_dlls_per_proc (average amount of DLLs per subprocess), handles.avg_handles_per_proc (average handlers, i.e. open resources like files, ports, etc), handles.nport (amount of ports), handles.nfile (amount of files), handles.nevent (amount of event handles used for event synchronization), handles.ndesktop (amount of desktop handles associated to graphical user interfaces), handles.nkey (amount of registry key handles), handles.nthread (amount of thread handles), handles.ndirectory (amount of directory handles), handles.nsemaphore (amount of semaphore handles used for synchronization), psxview.not_in_pslist (number of subprocesses not present in the pslist program output, which means they are potentially hidden), psxview.not_in_ethread_pool (number of subprocesses not found in the ethread pool, i.e. data structures that store thread information [13]), psxview.not_in_session (number of processes not associated with any session), svcscan.kernel_drivers (number of kernel-mode drivers), svcscan.fs_drivers (number of file system drivers.).

B. Experimental Setup

Multiple experiments were made by combining feature selection techniques with classification algorithms (Random

Table I
MALWARE SAMPLES USED TO GENERATE THE DATASET

Malware Category	Malware Family	Count
Trojan Horse	Zeus	195
	Emotet	196
	Refroso	200
	Scar	200
	Reconyc	157
Spyware	180Solutions	200
	Coolwebsearch	200
	Gator	200
	Transponder	241
	TIBS	141
Ransomware	Conti	200
	MAZE	195
	Pysa	171
	Ako	200
	Shade	220

Forest, Support Vector Machine and a Fully Connected Artificial Neural Network) together with some data preprocessing techniques like train-test split, SMOTE and feature scaling and methods to evaluate the performance of our model and the quality of the used feature selection algorithm. For instance, one experiment consists of splitting the data into training and test subsets, applying SMOTE to balance the training samples, applying standard scaling on the training features, fitting the feature selection algorithm to the training data and removing the non-selected features, passing those selected features into the classifier to train it, using the trained classifier to get the predicted classes for the test subset and finally getting the F1, recall, precision and AUC-ROC classification macro-average metrics for them. This same structure is utilized for all the feature selection and classification algorithms combinations (including the combination of absence of feature selection step and classifier), for 4 classes and for 16 classes, respectively, following a k-fold cross-validation structure with multiple hyperparameters for each feature selection and classification algorithm. After all the experiments were run, we applied a statistical hypothesis test by means of Wilcoxon signed-rank test [14] to determine whether there was a statistically significant difference between the F1 scores of the all-features version against each of the feature selection algorithms for a same classifier; each of all the k-folds' F1 scores were used for the Wilcoxon test. A block diagram illustrating the pipeline flow of these experiments can be seen in Figure 1 .

The classification algorithms we use are Random Forest, Support Vector Machine and a fully-connected neural network. We chose these algorithms to have variety of behavioral nature to test our feature selection methods and because these algorithms were also used in previous work from Cevallos, et al. [4] which we wanted to research upon.

The feature selection algorithms we use are Mutual Information, Recursive Feature Elimination, Boruta and LASSO. We chose this ones to comply with some of the most popular feature selection algorithm categories: filter, wrapper, hybrid and LASSO categories [15].

Mutual information is a filter method as it ranks features based on importance metrics, independent of any downstream classification algorithm (which fits perfectly with our goals). Then the least important features are filtered out from the dataset [7].

Recursive Feature Elimination is a wrapper method as it trains the classifier, assigns weights to the features indicating their importance in making predictions, then it removes the least important feature and it repeats the operation (training the classifier, evaluating feature importance and so on), but with the reduced dataset, until reaching an specific number of selected features determined by hyperparameter. As it can be seen, it wraps the classifier (e.g. Random Forest in our experiments), controlling the data that it passes to it and evaluating its results while operating [8].

Boruta plays the role of hybrid method as it acts as a filter method without relying on the underlying classifier, but at the same time it wraps a Random Forest classifier (independent of the actual classifier used in each experiment) that it uses for its operation [9].

The dataset was divided into training and test subsets, keeping 80 percent for the training group and the rest for test. The dataset is shuffled before splitting, to get a balanced distribution of classes in the training set. SMOTE was used as a data balancing technique to overcome this limitation [5].

For the feature scaling phase we run standardarization (i.e. z-score normalization) by substracting the mean and dividing to the standard deviation for each feature's values [16].

For the cross-validation, we use a combination of k-fold with 10 folds and a grid search to find the best parameters and perform cross-validation on different subsets of our training data, ensuring we have balanced target variable ocurrences in each fold, thanks to the stratified characteristic of the k-fold setup [17].

The hyperparameter usage for the experiments is detailed in Table II.

In addition to the mentioned experiments we also wanted to try a variation of the data aggregation technique seen in the work of [18] to enhance the inference performance of the 4-class classification relying on the results of inferencing on the 16 sub-classes. Generalizing [18] to our problem in hand, the decomposition of the inference classes (the 4 Benign, Ransomware, Spyware and Trojan classes) into training classes (the 15 malware families plus benign) and the subsequent training on those 16 more atomic, locally related, classes, which cause the 16-class classifier model to be fit differently than the 4-class version, allows to develop a more versatile embedded-like model which combines the 4-class and 16-class models into one pipeline

Table II
Algorithms and their hyperparameters

Algorithm	Hyperparameters
Mutual Information	- K features: 40, 50.
Recursive Feature Elimination	- N features to select: 40, 50.
Boruta	- Random Forest N estimators: 100. - Random Forest criterion: entropy. - Random Forest max depth: 10.
LASSO	- LinearSVC penalty: l1. - Max iterations: 1000.
Random Forest	- N estimators: 100. - Criterion: entropy. - Max depth: 10.
Support Vector Machine	- Kernel: linear. - Gamma: 1. - C: 1. - Max iterations: 1000
Neural Network	- # hidden layers: 1, 2, 3. - # neurons: 64, 128 (all the hidden layers have the same number of neurons). - Regularizer lambda: 1e-4.

to enhance inference effectiveness by adding an additional metric -the sum of the 16-class predicted probabilities grouped into each of the 4-class (e.g. if we consider the Ransomware as our inference class, sum the predicted probabilities for the Ransomware-Conti, Ransomware-MAZE, Ransomware-Pyza, Ransomware-Ako, Ransomware-Shade training classes)- that tells the probability that the evaluated data corresponds to Ransomware, but based on a different model (16-class). We perform this experiment for the best 4-class model (based on F1 score) only and its corresponding 16-class model, where we compare the test data subset's F1-score, precession, recall and AUC-ROC scores of the original 4-class model and the 16-class model, grouping the latter's predicted probabilities previous to obtain its classification metrics.

C. Tools

The tools used to develop the experiments were scikit-learn, imbalanced-learn, scikeras, Boruta, scipy, pandas and numpy. The models were trained without using a GPU due to the limitations of scikit-learn. The GridSearchCV was run with unlimited jobs to allow parallelized and faster training.

All of our experiments code can be seen in https://github.com/bryancalisto/feat-sel-in-malware-classif .

IV. Results and Discussion

The main goal in our study is not to compare scores between different classifiers, but between different feature selection methods (or the absence of feature selection) with the same classifier, so our results analysis is based on that premise. Additionally, even though we captured multiple classification metrics, we are specially interested in F1 and will use it for our analysis. However, we also present the results for the other metrics as well as a complement.

A. 4-class classification

Table III
F1 Metric Across Models and Feature Selection Methods in Test Set (4-class)

Feature Selection	Model	F1
All Features	RF	0.725856
	NN	0.602816
	SVM	0.420526
Mutual Information	RF	0.726526
	NN	0.640779
	SVM	0.430813
Recursive	RF	0.732269
	NN	0.683575
	SVM	0.444486
Boruta	RF	0.720154
	NN	0.613263
	SVM	0.471411
LASSO	RF	0.725121
	NN	0.674727
	SVM	0.420526

Table IV
Recall and Precision Metrics Across Models and Feature Selection Methods in Test Set (4-class)

Feature Selection	Model	Recall	Precision
All Features	RF	0.727311	0.741130
	NN	0.629287	0.697934
	SVM	0.519167	0.552629
Mutual Information	RF	0.727514	0.739113
	NN	0.652594	0.681140
	SVM	0.517640	0.560247
Recursive	RF	0.732990	0.742910
	NN	0.686293	0.708917
	SVM	0.526450	0.570583
Boruta	RF	0.721634	0.733689
	NN	0.635701	0.692483
	SVM	0.540250	0.575724
LASSO	RF	0.726251	0.737169
	NN	0.676002	0.675237
	SVM	0.519167	0.552629

Table V
Accuracy and AUC-ROC Metrics Across Models and Feature Selection Methods in Test Set (4-class)

Feature Selection	Model	Accuracy	AUC-ROC
All Features	RF	0.815700	0.945923
	NN	0.748464	0.923855
	SVM	0.673038	0.840189
Mutual Information	RF	0.815785	0.946680
	NN	0.764164	0.921840
	SVM	0.672355	0.836179
Recursive	RF	0.819625	0.947284
	NN	0.787201	0.934807
	SVM	0.678669	0.839803
Boruta	RF	0.811945	0.944982
	NN	0.752901	0.922223
	SVM	0.688567	0.836468
LASSO	RF	0.815017	0.946101
	NN	0.781655	0.925626
	SVM	0.673038	0.841938

After executing all the experiments for the 4-class classification, we came up with the scores for the test dataset shown in Table III, Table IV and Table V.

For Random Forest we can see there is minimal differences between all the models scores in the test dataset. The highest F1 score was obtained by the model with Recursive Feature Elimination followed by the model with Mutual

Table VI
F1 SCORES, NUMBER OF USED FEATURES AND STATISTICAL SIGNIFICANCE SUMMARY (4-CLASS)

	All feats	MI	RFE	Boruta	LASSO
RF	0.722172	0.726347* (40/55)	0.729295* (40/55)	0.724516* (47/55)	0.725397 (52/55)
SVM	0.427190	0.438032 (50/55)	0.422077 (50/55)	0.439070 (47/55)	0.427190 (52/55)
NN	0.628682	0.643054 (50/55)	0.646367 (40/55)	0.635604 (47/55)	0.574272* (20/55)

Information, LASSO, Boruta and finally the model without feature selection.

Table VII
RANDOM FOREST F1 HYPOTHESIS TESTS RESULTS (4-CLASS)

Feature Selection Method	**p-value**
All Features (pivot)	–
Mutual Information	0.01953125*
Recursive Feature Elimination	0.00390625*
Boruta	0.048828125*
LASSO	0.064453125

The hypothesis test results for this classifier (Table VII) tells us there is no statistical difference between the model without feature selection F1 and the model that uses LASSO F1, so it's slightly higher score is not definitive. Nevertheless, that model selected and used 52 out of 55 features, marking an advantage over the all-features model in terms of, at least, efficiency. The model with Recursive Feature Elimination, with Mutual information and with Boruta differed statistically significantly from the all-features model. They have not only a higher F1, but also used less features than the latter (40 for RFE and MI and 47 for Boruta), which demonstrate these feature selection methods prove effective for 4-class classification in our dataset [15]. Using these methods leverages considerably less features to obtain a simpler, more efficient and accurate model [15]. The amount of selected features per method for the 4-class models is shown in Table VI in parentheses.

For SVM we see there is no statistically significant difference between the models with feature selection and the model without it (Table VIII) F1 scores, but, still, the models managed to use less features, which results in a simpler and efficient to train model [15]. For the models that applied Mutual Information and Recursive Feature Elimination, 50 features were selected. Boruta selected 47 features and LASSO selected 52.

Table VIII
SVM F1 HYPOTHESIS TESTS RESULTS (4-CLASS)

Feature Selection Method	**p-value**
All Features (pivot)	–
Mutual Information	0.625
Recursive Feature Elimination	1.0
Boruta	0.625
LASSO	1.0

The Neural Nework best models were the ones with 2 hidden layers (the max allowed according to hyperparameters) and 128 neurons per layer (the max allowed according to hyperparameters).

For the Neural Network we see slightly higher F1 scores for all the feature selection methods, except for LASSO, which fell notably under the all-features mark. However, according to the hypothesis test, the only method that presents a statistically significant difference when compared against the all-features version is LASSO. Mutual Information, Recursive Feature Elimination and Boruta selected 50, 40 and 47 features, respectively, while not showing a statistical difference with respect to all-features on F1, which positions them as good alternatives for selecting features with this type of classifier.

Table IX
NEURAL NETWORK F1 HYPOTHESIS TESTS RESULTS (4-CLASS)

Feature Selection Method	**p-value**
All Features (pivot)	–
Mutual Information	0.16015625
Recursive Feature Elimination	0.10546875
Boruta	0.556640625
LASSO	0.013671875*

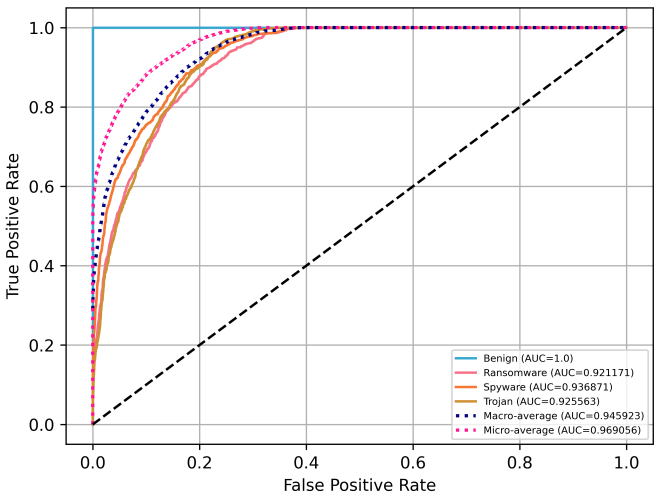

Figure 2. AUC-ROC for RF with all features (4-class)

According to the obtained AUC-ROC values presented in Table V, all the models are above the random prediction threshold, being Random Forest the best models, followed by the Neural Network models and the SVM. The Random Forest AUC-ROC scores for the 4 different classes, including macro and micro averages, are shown in Fig IV-A, 3, 4, 5 and 6.

B. 16-class classification

The 16-class classification is expected to have lower scores than the 4-class due to the presence of less data of each class [4].

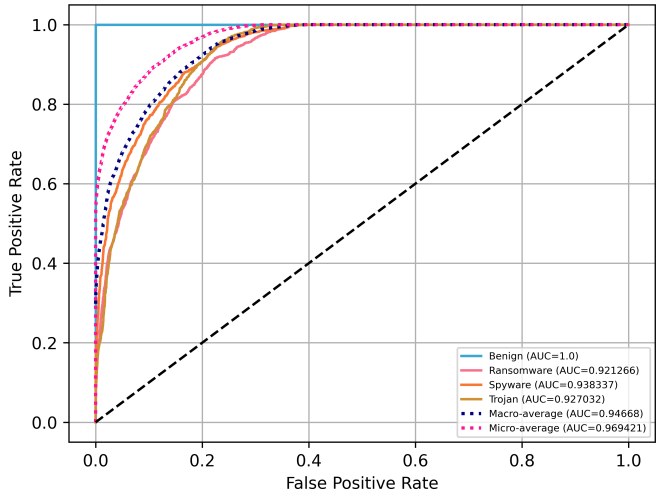

Figure 3. AUC-ROC for RF with Mutual Information (4-class)

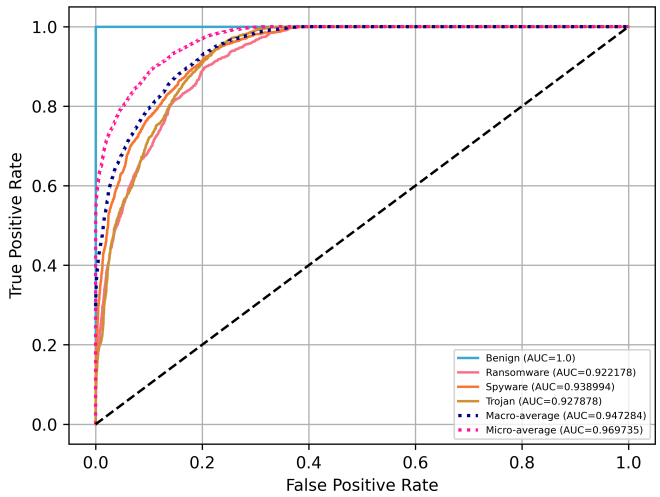

Figure 4. AUC-ROC for RF with RFE (4-class)

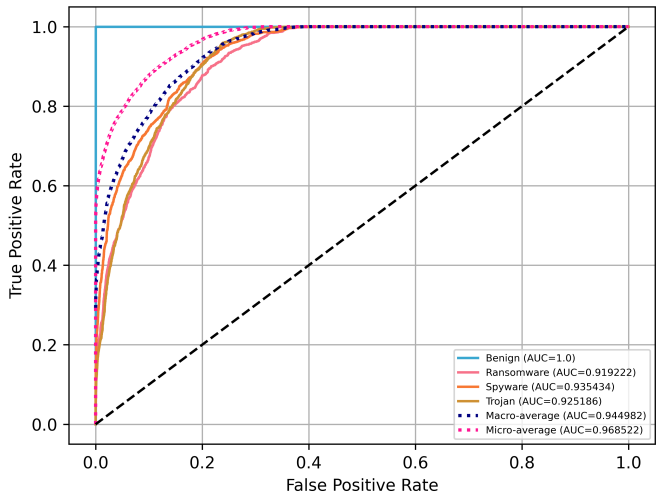

Figure 5. AUC-ROC for RF with Boruta (4-class)

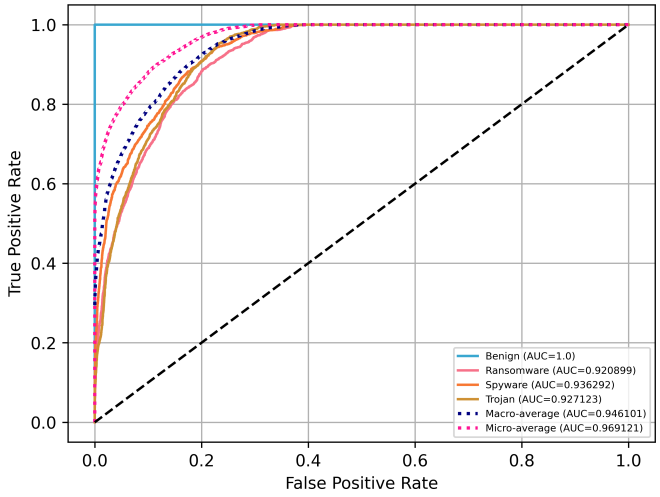

Figure 6. AUC-ROC for RF with LASSO (4-class)

Table X
RF REMOVED FEATURES COUNTS (4-CLASS)

	Feature	Count
0	pslist.nprocs64bit	4
1	handles.nport	4
2	svcscan.interactive_process_services	4
3	psxview.not_in_eprocess_pool	3
4	psxview.not_in_eprocess_pool_false_avg	3
5	svcscan.fs_drivers	3
6	callbacks.nanonymous	3
7	callbacks.ngeneric	3
8	psxview.not_in_pslist	2
9	psxview.not_in_ethread_pool	2
10	psxview.not_in_pspcid_list	2
11	psxview.not_in_csrss_handles	2
12	psxview.not_in_session	2
13	modules.nmodules	2
14	psxview.not_in_deskthrd	1
15	psxview.not_in_pslist_false_avg	1

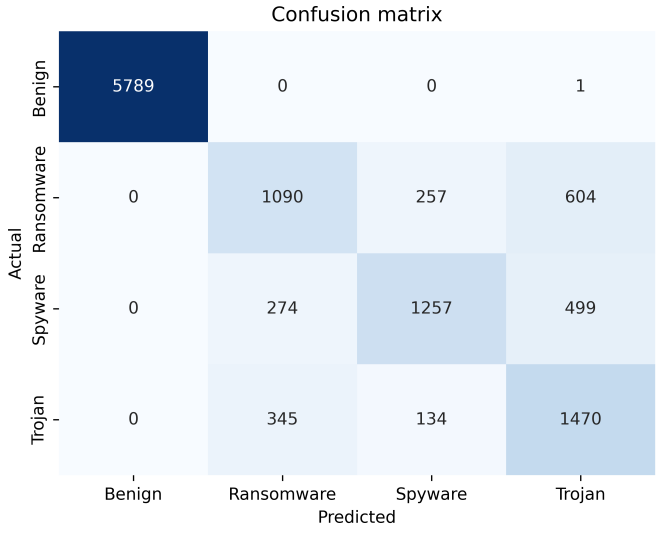

Figure 7. Confusion Matrix for RF with RFE (4-class)

After executing all the experiments for the 16-class classification, we came up with the scores for the test dataset

Table XI
F1 Metric Across Models and Feature Selection Methods in Test Set (16-class)

Feature Selection	Model	F1
All Features	RF	0.412597
	NN	0.351370
	SVM	0.125989
Mutual Information	RF	0.416351
	NN	0.364812
	SVM	0.114743
Recursive	RF	0.416839
	NN	0.405771
	SVM	0.132116
Boruta	RF	0.404174
	NN	0.393597
	SVM	0.133433
LASSO	RF	0.414658
	NN	0.358329
	SVM	0.125989

Table XII
Recall and Precision Metrics Across Models and Feature Selection Methods in Test Set (16-class)

Feature Selection	Model	Recall	Precision
All Features	RF	0.430129	0.442082
	NN	0.353597	0.435550
	SVM	0.155490	0.226638
Mutual Information	RF	0.432787	0.445293
	NN	0.366505	0.447298
	SVM	0.146662	0.226485
Recursive	RF	0.432251	0.444261
	NN	0.413525	0.444199
	SVM	0.158313	0.233446
Boruta	RF	0.421060	0.429688
	NN	0.398698	0.445034
	SVM	0.159543	0.230190
LASSO	RF	0.431286	0.442612
	NN	0.355404	0.442045
	SVM	0.155490	0.226638

Table XIII
Accuracy and AUC-ROC Metrics Across Models and Feature Selection Methods in Test Set (16-class)

Feature Selection	Model	Accuracy	AUC-ROC
All Features	RF	0.689249	0.927255
	NN	0.647440	0.908472
	SVM	0.544369	0.823399
Mutual Information	RF	0.690529	0.928260
	NN	0.653925	0.914767
	SVM	0.540700	0.827369
Recursive	RF	0.690700	0.927782
	NN	0.679096	0.927028
	SVM	0.542406	0.819787
Boruta	RF	0.684642	0.926734
	NN	0.670734	0.924390
	SVM	0.544625	0.821569
LASSO	RF	0.690017	0.927768
	NN	0.647696	0.912434
	SVM	0.544369	0.821888

shown in Table XI, Table XII and Table XIII.

For Random Forest we can see there is no statistically significant difference between the F1 scores of the models that use feature selection and the all-features model. MI, RFE, Boruta and LASSO all selected less features than available; 40, 50, 51 and 52, respectively, which means they can get results equivalent to the all-features model, but more efficiently [15]. The p-values can be seen in Table XV.

Table XIV
F1 scores, number of used features and statistical significance summary (16-class)

	All feats	MI	RFE	Boruta	LASSO
RF	0.406580	0.407123 (40/55)	0.406296 (50/55)	0.405808 (51/55)	0.404551 (52/55)
SVM	0.122998	0.121775 (50/55)	0.123194 (50/55)	0.123272 (51/55)	0.120626 (52/55)
NN	0.365893	0.379989 (50/55)	0.379290 (40/55)	0.375245 (51/55)	0.322607* (20/55)

Table XV
Random Forest F1 Hypothesis Tests Results (16-class)

Feature Selection Method	p-value
All Features (pivot)	–
Mutual Information	0.625
Recursive Feature Elimination	0.556640625
Boruta	0.6953125
LASSO	0.193359375

The SVM models, similarly to the Random Forest models, registered no statistically significant differences in F1. MI and RFE selected 50 features, Boruta 51 and LASSO 52, meaning we can get equivalent results to the all-features model by using less features [15]. Can see the p-values in Table XVI.

Table XVI
SVM F1 Hypothesis Tests Results (16-class)

Feature Selection Method	p-value
All Features (pivot)	–
Mutual Information	0.76953125
Recursive Feature Elimination	0.921875
Boruta	0.556640625
LASSO	0.375

The Neural Network models, similar to their corresponding 4-class version, present statistically significance difference only in LASSO, and the F1 score is considerably low compared with the all-features; it selected 20 features out of 55 due to a possibly too high alpha [19]. MI, RFE and Boruta selected 50, 40 and 51 features, respectively, which allow obtaining a more efficient and simpler model with equivalent F1 scores [15]. The p-values can be seen in Table XVII.

Table XVII
Neural Network F1 Hypothesis Tests Results (16-class)

Feature Selection Method	p-value
All Features (pivot)	–
Mutual Information	0.193359375
Recursive Feature Elimination	0.275390625
Boruta	0.275390625
LASSO	0.037109375*

According to the obtained AUC-ROC values presented in Table XIII, all the models are above the random

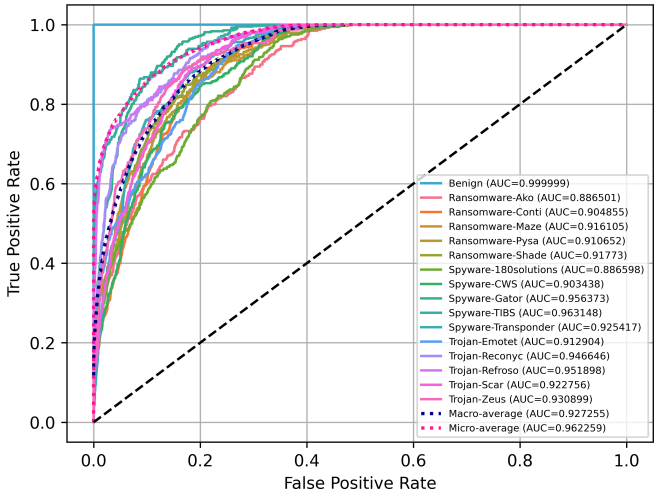

Figure 8. AUC-ROC for RF with all features (16-class)

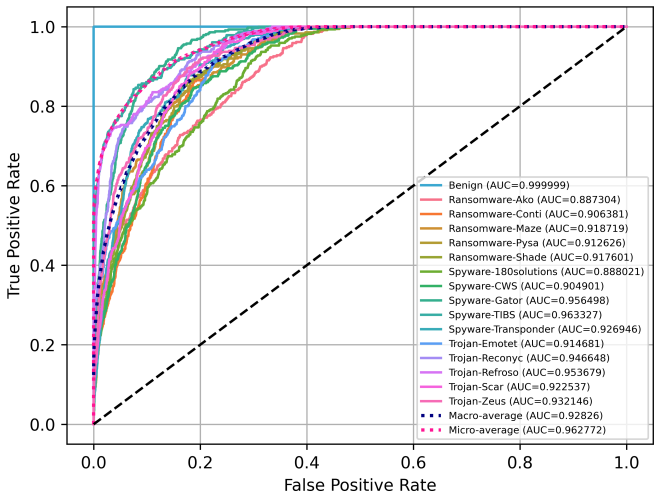

Figure 9. AUC-ROC for RF with Mutual Information (16-class)

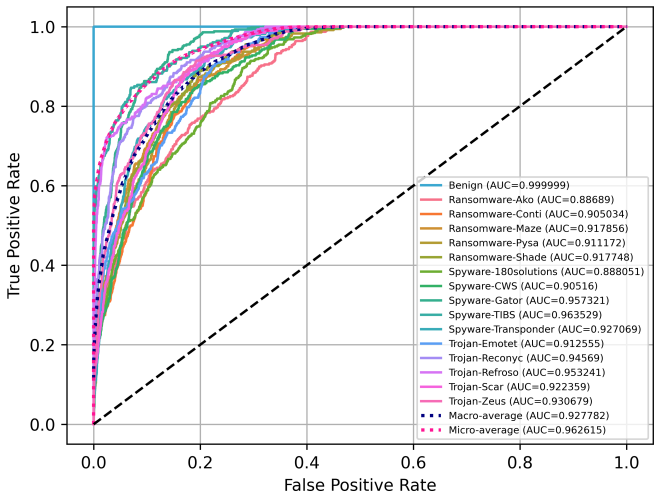

Figure 10. AUC-ROC for RF with RFE (16-class)

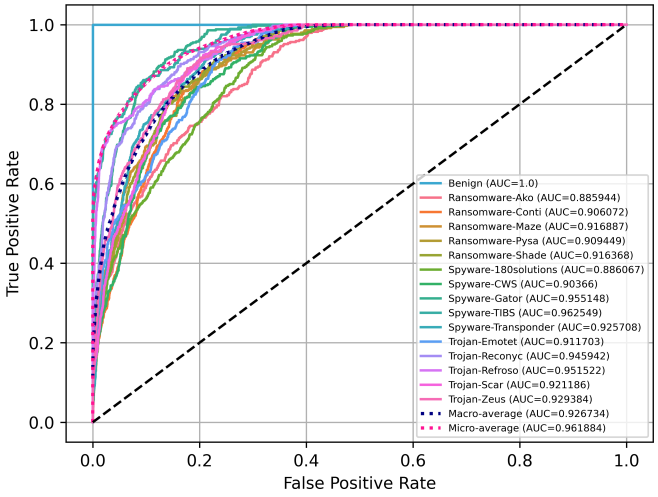

Figure 11. AUC-ROC for RF with Boruta (16-class)

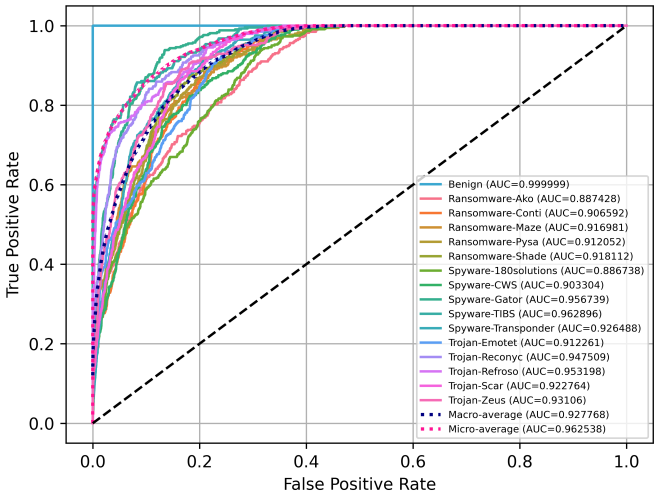

Figure 12. AUC-ROC for RF with LASSO (16-class)

prediction threshold, being Random Forest the best models, followed by the Neural Network models and the SVM. The Random Forest AUC-ROC scores for the 16 different classes, including macro and micro averages, are shown in Fig 8, 9, 10, 11 and 12.

C. Grouping 16 classes probabilities into 4 classes

The best 4-class model was Random Forest with Recursive Feature Elimination. We took the 16-class equivalent model and used it for our inference probability grouping experiment. The results of grouping the 16-class test dataset predicted probabilities by category, getting their classification metrics and comparing them to the ones from the original 4-class can be seen in Table XIX.

F1, Recall, Precision and Accuracy are slightly lower in the grouped scores. In general we expected to have lower classification scores in the grouped version as the 16-class classifier is expected to have lower scores due to the higher amount of classes and the reduced number of

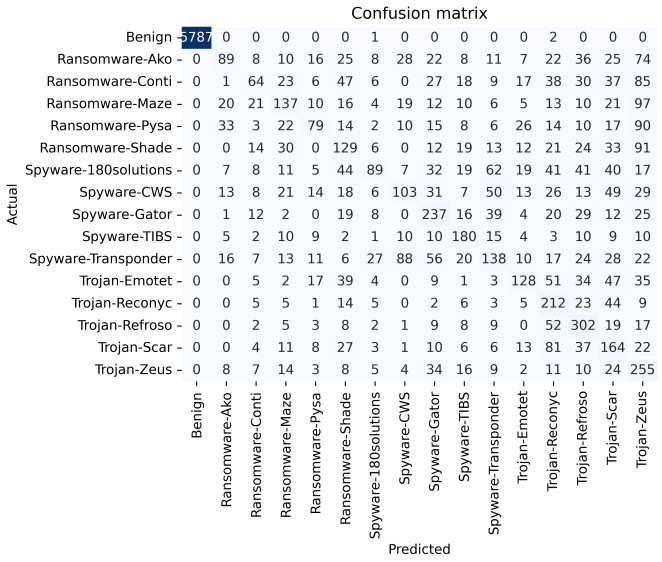

Figure 13. Confusion Matrix for RF with Mutual Information (16-class)

Table XVIII
RF removed features counts (16-class)

	Feature	Count
0	pslist.nprocs64bit	4
1	handles.nport	4
2	svcscan.interactive_process_services	4
3	psxview.not_in_eprocess_pool	3
4	psxview.not_in_eprocess_pool_false_avg	2
5	psxview.not_in_pslist	1
6	psxview.not_in_ethread_pool	1
7	psxview.not_in_pspcid_list	1
8	psxview.not_in_csrss_handles	1
9	psxview.not_in_session	1
10	psxview.not_in_deskthrd	1
11	modules.nmodules	1
12	svcscan.fs_drivers	1
13	callbacks.nanonymous	1
14	callbacks.ngeneric	1

instances of each class which compromises the training process quality [20]. AUC-ROC got a bit higher value, which represents a potential contribution from the 16-class predictions grouped and used as a complement for the 4-class classifier when inferencing, but this is not conclusive as we are not doing statistical comparison between the AUC-ROC scores. A more exhaustive investigation is needed to confirm the veracity of these conclusions.

Table XIX
Classification metrics comparison between the 16 classes grouped into 4 classes and the 4 original classes

	16 grouped into 4	4 original
F1	0.724013	0.732269
Recall	0.725003	0.73299
Precision	0.730546	0.74291
Accuracy	0.814164	0.819625
AUC-ROC	0.9482	0.947284

D. Removed Features

Table X and Table XVIII show counts for the removed features of 4-class and 16-class Random Forest, respectively. Both share similar removed features at the top of the counts, like pslist.nprocs64bit, handles.nport and svcscan.interactive_process_services. Specifically, these 3 features have the characteristic of being zero for most of the dataset records. There is also a pattern with psxview features where they are apparently removed due to having high correlation with other features [15]. The high correlation can be seen in the correlation matrix included in our source code notebooks and it is not included here due to visualization limitations. For instance, psxview.not_in_pslist_false_avg, which is a removed feature in the 4-class RF models (see Table X), has high correlation with psxview.not_in_pslist, psxview.not_in_pspcid_list, psxview.not_in_session, psxview.not_in_pspcid_list_false_avg and psxview.not_in_session_false_avg.

V. Conclusions

The obtained results revealed feature selection techniques are definitely effective in the CIC-MalMem-2022 dataset. Specifically, Recursive Feature Elimination and Mutual information showed rather useful with Random Forest, with the first one not only removing 15 out of the 55 features, but also increasing the F1 score in the 4-class, and the second, although it did not increase the target scores significantly, it removed 15 out of 55 features without causing significant changes in the scores with respect to the all-features version which helps from an efficiency and simplicity perspective. Most of the removed features were high correlated features and others were features with low variance, where most of the rows had the same values (e.g. zero). The SVM and NN had similar feature selection patterns, but did not get statistically significant differences in classification scores, except for NN with LASSO which resulted in lower classification performance due to regularization alpha hyperparameter influence. Grouping probabilities did not result in notable benefits; nevertheless, as stated, there is still more research to be done to confirm whether it is beneficial or not.

References

[1] P. Vinod, R. Jaipur, V. Laxmi, and M. Gaur, "Survey on malware detection methods," in *Proceedings of the 3rd Hackers' Workshop on computer and internet security (IITKHACK'09)*, 2009, pp. 74–79.

[2] J. Singh and J. Singh, "Challenge of malware analysis: malware obfuscation techniques," *International Journal of Information Security Science*, vol. 7, no. 3, pp. 100–110, 2018.

[3] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences*, vol. 9, no. 20, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/20/4396

[4] D. Cevallos-Salas, F. Grijalva, J. Estrada-Jiménez, D. Benítez, and R. Andrade, "Obfuscated privacy malware classifiers based on memory dumping analysis," *IEEE Access*, vol. 12, pp. 17 481–17 498, 2024.

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, p. 321–357, Jun. 2002. [Online]. Available: http://dx.doi.org/10.1613/jair.953

[6] M. A. Talukder, F. Hasan, M. Islam, M. A. Uddin, A. Akhter, M. Yousuf, F. Alharbi, and M. A. Moni, "A dependable hybrid machine learning model for network intrusion detection," *Journal of Information Security and Applications*, vol. 72, p. 103405, 02 2023.

[7] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical Review E*, vol. 69, no. 6, Jun. 2004. [Online]. Available: http://dx.doi.org/10.1103/PhysRevE.69.066138

[8] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1, pp. 389–422, Jan 2002. [Online]. Available: https://doi.org/10.1023/A:1012487302797

[9] M. Kursa, A. Jankowski, and W. Rudnicki, "Boruta - a system for feature selection," *Fundam. Inform.*, vol. 101, pp. 271–285, 01 2010.

[10] R. Muthukrishnan and R. Rohini, "Lasso: A feature selection technique in predictive modeling for machine learning," in *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*, 2016, pp. 18–20.

[11] T. Carrier, P. Victor, A. Tekeoglu, and A. Habibi Lashkari, "Detecting obfuscated malware using memory feature engineering," 01 2022, pp. 177–188.

[12] A. H. Lashkari, B. Li, T. L. Carrier, and G. Kaur, "Volmemlyzer: Volatile memory analyzer for malware classification using feature engineering," in *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS)*, 2021, pp. 1–8.

[13] barrygolden, "Windows Kernel Opaque Structures - Windows drivers — learn.microsoft.com," https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/eprocess#ethread, [Accessed 14-11-2024].

[14] W. J. Conover, *Practical nonparametric statistics*. john wiley & sons, 1999, vol. 350.

[15] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Comput. Surv.*, vol. 50, no. 6, Dec. 2017. [Online]. Available: https://doi.org/10.1145/3136625

[16] D. Anggoro and W. Supriyanti, "Improving accuracy by applying z-score normalization in linear regression and polynomial regression model for real estate data," *International Journal of Emerging Trends Technology in Computer Science*, pp. 549–555, 11 2019.

[17] D. Anguita, L. Ghelardoni, A. Ghio, L. Oneto, S. Ridella *et al.*, "The'k'in k-fold cross validation." in *ESANN*, vol. 102, 2012, pp. 441–446.

[18] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, Feb 2017.

[19] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996. [Online]. Available: http://www.jstor.org/stable/2346178

[20] R. Andonie, "Extreme data mining: Inference from small datasets," *International Journal of Computers, Communication & Control*, 2010.