



Kötü Amaçlı Yazılım Tespiti İçin Temel Bileşen Analizi ve Anova Özellik Seçiminin Karşılaştırmalı Analizi

Taha ETEM^{1*}  

¹Bilgisayar Mühendisliği, Mühendislik Fakültesi, Çankırı Karatekin Üniversitesi, Çankırı, Türkiye.
tahaetem@karatekin.edu.tr

Geliş Tarihi: 07.2.2025
Kabul Tarihi: 06.1.2026

Düzeltilme Tarihi: 09.11.2025

doi: <https://doi.org/10.62520/fujece.1635121>
Araştırma Makalesi

Alıntı: T. Ethem, "Kötü amaçlı yazılım tespiti için temel bileşen analizi ve anova özellik seçiminin karşılaştırmalı analizi"
Fırat Üni. Deny. ve Hes. Müh. Derg., vol. 5, no 1, pp. 299-315, Şubat 2026.

Öz

Bu çalışma, Android kötü amaçlı yazılım tespiti için Temel Bileşen Analizi (PCA) ve ANOVA tabanlı özellik seçimi yöntemlerinin karşılaştırmalı bir analizini sunarak, bunların sınıflandırma doğruluğu ve hesaplama verimliliği üzerindeki etkilerini değerlendirmektedir. Üç ön işleme senaryosu incelenmiştir: 241 özelliğe sahip orijinal veri setini kullanma, özellik çıkarma için PCA uygulama (varyans eşikleri nedeniyle tüm bileşenleri tutma) ve özellik setini 120'ye düşürmek için ANOVA kullanma. Destek Vektör Makineleri (SVM), Geniş Sinir Ağları ve Lojistik Regresyon sınıflandırıcıları, 5 katlı çapraz doğrulama yoluyla optimize edilen hiper parametrelerle bu veri setleri üzerinde eğitilmiştir. Sonuçlar, SVM'nin tüm senaryolarda tutarlı bir şekilde en yüksek doğruluğu elde ettiğini ve PCA ile %99,25'a ulaştığını göstermiştir. Ancak, PCA model boyutunu azaltmada başarısız olmuş ve orijinal veri setine kıyasla SVM için eğitim sürelerini artırmıştır. Bunun aksine, ANOVA özellik sayısını etkili bir şekilde azaltarak SVM eğitim süresini 4,81 saniyeye düşürürken %98,95 doğruluk oranı elde etmiştir. Bu bulgular, ANOVA'nın yüksek tespit performansını azaltılmış kaynak talepleriyle dengeleyen hesaplama açısından verimli bir yöntem olduğunu vurgulamaktadır. PCA doğruluğu marjinal olarak iyileştirirken, hesaplama maliyeti onu gerçek zamanlı uygulamalar için daha az pratik hale getirmektedir. Çalışma, ANOVA aracılığıyla özellik seçiminin Android kötü amaçlı yazılım tespiti için hem doğruluğu hem de verimliliği önceliklendirerek üstün bir takas sağladığı sonucuna varmıştır. Gelecekteki çalışmalar, genelleştirilebilirliği artırmak ve gelişen kötü amaçlı yazılım tehditlerini ele almak için gelişmiş özellik seçimi tekniklerini keşfetmeli ve modelleri çeşitli veri kümelerinde doğrulamalıdır.

Anahtar kelimeler: Kötü amaçlı yazılım tespiti, Özellik seçimi, Destek vektör makineleri, Android güvenliği

*Yazışılan yazar

İntihal Kontrol: Evet – Turnitin

Şikayet: fujece@firat.edu.tr

Telif Hakkı ve Lisans: Dergide yayın yapan yazarlar, CC BY-NC 4.0 kapsamında lisanslanan çalışmalarının telif hakkını saklı tutar.



Comparative Analysis of Principle Component Analysis and Anova Feature Selection in Malware Detection

Taha ETEM^{1*}  

¹Computer Engineering, Engineering Faculty, Çankırı Karatekin University, Çankırı, Türkiye.

¹tahaetem@karatekin.edu.tr

Received: 07.2.2025
Accepted: 06.01.2026

Revision: 09.11.2025

doi: <https://doi.org/10.62520/fujece.1635121>
Research Article

Citation: T. Etem, "Comparative analysis of principle component analysis and anova feature selection in malware detection", *Firat Univ. Jour. of Exper. and Comp. Eng.*, vol. 1, no 2, pp. 299-315, February 2026.

Abstract

This study presents a comparative analysis of Principal Component Analysis (PCA) and ANOVA-based feature selection methods for Android malware detection, evaluating their impact on classification accuracy and computational efficiency. Three preprocessing scenarios were examined: using the original dataset with 241 features, applying PCA for feature extraction (retaining all components due to variance thresholds), and employing ANOVA to reduce the feature set to 120. Support Vector Machines (SVM), Wide Neural Networks, and Logistic Regression classifiers were trained on these datasets, with hyperparameters optimized via 5-fold cross-validation. Results demonstrated that SVM consistently achieved the highest accuracy across all scenarios, peaking at 99.25% with PCA. However, PCA failed to reduce dimensionality of models and increased training times for SVM compared to the original dataset. In contrast, ANOVA effectively reduced the feature count, lowering SVM training time to 4.81 seconds while obtaining 98.95% accuracy. These findings highlight ANOVA as a computationally efficient method, balancing high detection performance with reduced resource demands. While PCA marginally improved accuracy, its computational cost renders it less practical for real-time applications. The study concludes that feature selection via ANOVA offers a superior trade-off for Android malware detection, prioritizing both accuracy and efficiency. Future work should explore advanced feature selection techniques and validate models on diverse datasets to enhance generalizability and address evolving malware threats.

Keywords: Malware detection, Feature selection, Support vector machines, Android security

* Corresponding author

1. Introduction

Malicious software is a heterogeneous collection of harmful code that is meant to disrupt or harm any computing systems and networks, as well as gain illegal access. Its development has turned out to be a topical issue in the field of cybersecurity. Malware has become widespread and it has been required to use more advanced detection techniques and methods of analysis. Malware taxonomy mostly focuses on its effect, purpose of operation and the methods used to create or deliver the malwares. There can be unique malware versions that can execute their devilish tasks without the permission of the user. Complexity of malware has increased due to the emergence of anti-detection systems that enable adaptation and avoidance of signature-based detection systems. In line with this, it has become necessary to have the correct identification of malware variants that can be added to traditional detection processes. Advanced advancements in machine-learning algorithms have shifted malware variants detection to program similarity. The strengths of this paradigm shift are in its capacity to better detect malicious software by exploiting structural affinities to known threats. The methods of malware detection are traditionally divided into two broad categories, signature-based and behaviour-based approaches. Signature-based detection uses existing patterns or signature of malicious artefacts to mark threats whereas behaviour-based detection watches the activity of software to determine the potentially harmful behaviour. [1, 2]. Also, cryptographic techniques play a crucial role in safeguarding sensitive data, ensuring secure communication, and protecting against unauthorized access or tampering in Android applications and systems [3-6].

Recent studies have emphasized the usefulness of feature selection based on system-call data on Android malware detection systems. Comparative studies were conducted on various techniques of feature-selection methods to determine their ability to reduce the size of system -call sets whilst maintaining or improving classification accuracy. The results indicate that a reduced set of the best selected features can reach desirable classification rates thus confirming the efficiency of the offered methodology in detecting malware.

The work of Aero is focused on the recognition and categorization of malware by using several machine-learning algorithms. The assessment involved Support Vector Machines (SVM), Decision Trees, and ensemble techniques of categorizing malware according to behavioural signature. Empirical findings show that ensemble methods had the best detection accuracy with a result of about 94%. The system is highly robust in the process of differentiating between benign and malicious software. In addition, the research demonstrates the urgent necessity of the unceasing refresh of the training information to be flexible to naturally evolving malware threats. [7].

Adewale introduces a deep learning framework for malware detection, integrating multiple layers of anomaly detection in cloud environments. The study employed a hybrid model combining Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to detect anomalies in network traffic data. The system demonstrated a high accuracy of approximately 92%. The model's performance validates the effectiveness of deep learning-based cybersecurity solutions for identifying malware in complex cloud infrastructures [8].

Cuiying et al. examine how machine-learning methods affect the detection of Android malware, with specific focus on the effect of the obfuscation methods on the detection accuracy. The authors contrast the ability of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to detect instances of obfuscated malware in a dataset that had been annotated by antivirus engines. The results of their research show that obfuscation decreases the detection performance by about 15%, and CNN models have an accuracy of around 88 percent in situations when no obfuscation is used. These findings highlight the need to have advanced countermeasures to counter the obfuscation techniques in malware detection [9].

Rajest and Regin analyse the use of the Catboost classifier as one of the graduate-boosting algorithms to detect Android ransomware. The paper examines a sample of Android applications where behavioural characteristics are derived. The choice of Catboost is explained by the fact that it is more accurate and capable of addressing the categorical data. The resulting model achieves a high accuracy rate of 96 % which is higher than classic classifiers like the Random Forests and the Decision Trees. These results indicate the effectiveness of Catboost as a mobile malware detector [10].

Totham et al. propose a machine learning-based approach for detecting ransomware through adaptive encryption pattern recognition. Their system monitors file encryption patterns that signal potential ransomware attacks. A Recurrent Neural Network (RNN) model was implemented to learn patterns associated with malicious encryption activities in real time. The model demonstrated an accuracy of approximately 93% in ransomware detection, effectively identifying threats at an early stage. This study highlights the effectiveness of dynamic, behaviour-based machine learning models in enhancing ransomware detection capabilities [11].

Haq and Khuthaylah introduce a feature extraction technique combined with a soft voting ensemble to enhance malware detection on Android platforms. The study underscores the significance of carefully selecting relevant features from a vast set of extracted data. The proposed soft voting mechanism integrates predictions from multiple models, achieving a high detection accuracy of around 95%. This improvement in precision plays a crucial role in differentiating malware from legitimate applications. The findings emphasize the importance of refined feature selection in enhancing detection accuracy [12].

Kunku et al. present a novel ransomware detection method that leverages dynamic feature selection based on ransomware behaviour. The study applied various feature extraction techniques to accurately identify ransomware signatures and evaluated the approach using classifiers like Random Forest and k-Nearest Neighbors. The proposed method achieved an accuracy of 93%, demonstrating the effectiveness of feature selection in distinguishing ransomware from other types of malwares. The study's approach introduces a flexible framework for dynamic malware detection through behavioural analysis [13].

Mubarak and Alasmari suggested a malware detection framework implemented on Linux and based on memory analysis and deep learning to extract their features. The pipeline used in the study included a data preprocessing phase, key artifact extraction phase, and model selection phase. The model was able to achieve an accuracy of 90% per cent in its ability to detect malicious activities by including extracted system artifacts, process memory, and file access patterns, and such results indicate that OS-level artifact analysis along with advanced feature extraction are an effective strategy to classify malware [14].

Gihavo et al. proposed utilizing machine learning to identify file traps at an early stage using file trap selection and machine learning. The analysis derived characteristics of file manipulation and used them with the classification algorithms to detect early signs of ransomware. The method yielded a precision of about 91, thereby highlighting its worth in identifying ransomware attacks before they cause great harm. Focus on extracting features in the initial stages makes this model an active-based ransomware detection system [15]. Dolesi et al. investigate the use of sequences of operation codes generated during program execution for ransomware detection. By extracting opcode features and employing a k-Nearest Neighbour classifier makes the model effectively identified ransomware activities. The model achieves an accuracy of 92.8%. Feature selection was critical in this study, as it reduced the required feature set to 400, improving computational efficiency without compromising detection performance. This research illustrates the efficiency of opcode-based features and feature selection for high-performance malware classification on Windows systems [16]. In this study, detection of malware on Android Malware Dataset by machine learning methods is performed. Original dataset (241 features), selected features by ANOVA (120 features) and extracted features by PCA (241 features) are examined in terms of accuracy and efficiency. After evaluations, extracted features by PCA reached the highest accuracy in all of the scenarios. However, when computational effectiveness is considered, it can be said that the method applied with ANOVA feature selection reduces training times while giving minimal loss in terms of accuracy. The paper is organized as follows: Section 2 provides an overview of the related work in the field of Android malware detection, focusing on the application of machine learning techniques and feature selection methods. Section 3 details the materials and methods used in the study, including the dataset, preprocessing scenarios, and the machine learning algorithms employed for classification. In Section 4, the results of the experiments are presented, followed by a discussion of the findings, including a comparison of the performance of the different feature selection methods. Finally, Section 5 concludes the paper, summarizing the key findings and outlining potential directions for future research in Android malware detection.

This study offers a novel comparison between Principal Component Analysis (PCA) and ANOVA-based feature selection methods for Android malware detection, focusing on both classification accuracy and computational efficiency. While PCA is often favoured for dimensionality reduction, this study highlights its limitations, particularly its failure to reduce model complexity effectively and the significant increase in training time. In contrast, ANOVA demonstrates a more efficient feature selection approach, reducing the feature set by 50% while maintaining high classification accuracy, thus offering a better balance between performance and computational cost. This comparative analysis fills a gap in the literature by providing a detailed evaluation of how these two techniques impact real-time malware detection, offering valuable insights for optimizing malware detection systems in resource-constrained environments.

2. Tezpur University Android Malware Dataset

The high rate of development of information technology-based services led to a significant increase in the damage that malware attacks can cause. In spite of the fact that many malware detection methodologies have been suggested, the lack of sufficiently comprehensive datasets often impedes progress in this field. To this, Tezpur University Malware Windows Dataset (TUMALWD) and Tezpur University Android Malware Dataset (TUANDROMD) are proposed in order to promote validation of the malware detection methods in both Windows and Android systems respectively [17].

The framework of TUANDROMD dataset creation is divided into three phases namely: (1) data collection and storage, (2) data analysis, and (3) feature engineering. This will involve the downloading of malware and benign applications, running of manual checks, and deriving the salient features. Due to the complexity of the Android malware, manual inspection had been performed using the tools like Androguard, ApkInspector, ApkAnalyser and Smali-CFGs. In this research, the TUANDROMD dataset in a pre-processed version is used, with a total amount of 4,465 instances. The skewed class distribution is also evident in the dataset, with 3,566 samples of malware and 899 samples of benign (or goodware) that are also reflective of distributions in the real world. There were 241 characteristics obtained to be classified. These features are highly essential to be detected, and we can classify them as follows:

- 214 permission-based features
- 27 API-based features

The target attribute for classification is a binary category, labelling each instance as either 'malware' or 'goodware'. The creation of TUANDROMD addresses a critical need for up-to-date and relevant malware datasets for Android platforms. By offering comprehensive permission-based and API-based features, this dataset serves as a valuable resource for researchers and cybersecurity professionals. It enables the development of advanced malware detection systems that can adapt to the ever-changing landscape of cyber threats.

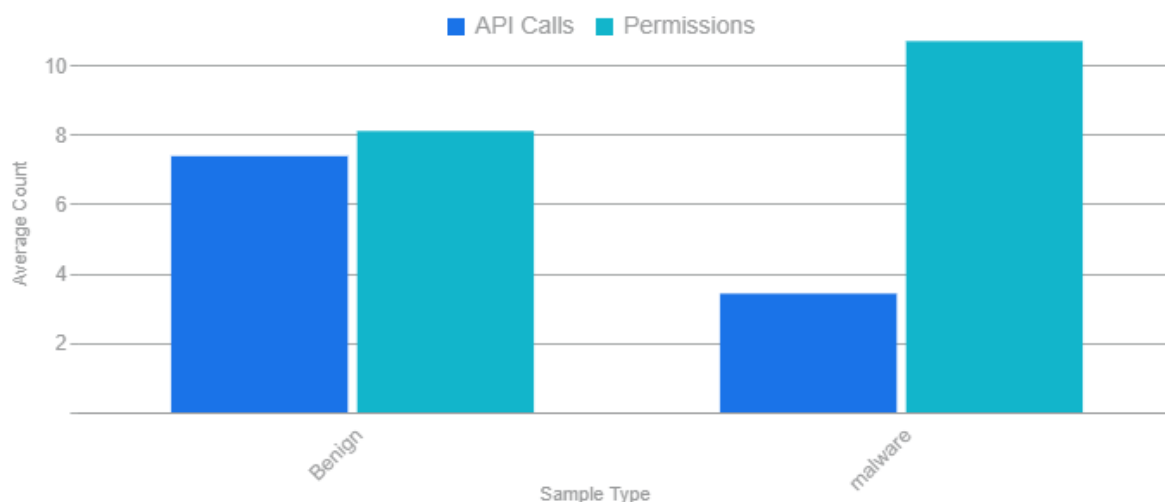


Figure 1. Average feature usage per sample

Figure 1 provides a simple but effective summary for dataset. Malware samples request more permissions on average (10.7) than benign samples (8.1). The trend is reversed for API calls. Benign samples use more of the monitored API calls on average (7.4) than malware samples (3.4). This suggests that malware is more "permission-hungry" while benign apps may have more complex functionality that uses these specific API calls.

The dataset records the modern trends of Android malware, which means that it is relevant to the current detection systems. The creation of the TUANDROMD serves a great demand of modern and relevant malware data in Android systems. These datasets are a treasure trove to researchers and cybersecurity experts by offering exhaustive host-based and network-based features on Windows malware and permission-based and API-based features on Android malware. They enable programmers to develop advanced malware detection solutions that can be modified upon the constantly changing cyber-threat situation.

3. Material and Methods

Machine-learning algorithms are used in invaluable classification and categorization of information that allows information to be grouped into preset categories. By analysing patterns, machine-learning models could produce inaccurate inferences in all types of fields, such as disease diagnosis in medicine, fraud detection, and automatic object recognition [18]. The ability to handle large and intricate datasets does not only improve efficiency and accuracy but also innovative ideas since it can uncover insights that would otherwise remain unnoticed by manual analysis [19].

To draw the line between Android applications, as being benign or malicious, various machine-learning algorithms were utilized, each of which has its own contribution to the malware detection process. Support Vector Machines (SVM) are particularly useful in large-dimensional feature spaces, which build optimal boundaries to the decision, even when the data is not linearly separable, therefore, making them adequate with the Malware classification in which subtle patterns distinguish between innocent and malicious behaviour. Neural networks offer the ability to acquire difficult patterns automatically, which is essential in malware detection where features associations are not necessarily obvious; they learn nonlinear relationships in the large volumes of data, thus, becoming an effective tool to classification with complex feature interactions. Logistic regression is a less complex but a highly effective binary classification method; its interpretability and ability to readily deal with linearly separable data make it a reliable option in cases where computational resources are limited. All these algorithms have been chosen due to their capability to address the complexity and the high dimensions of Android malware data and, thus, provide a good level of classification with varied sets of features.

Feature selection is a crucial preprocessing step in malware detection because the feature spaces derived from static and dynamic analyses tend to be high-dimensional and may contain redundant or irrelevant attributes. For example, Zhang et al. applied PCA in a PDF-malware detection scenario and showed that by reducing the number of features by about one-third they achieved nearly the same true positive rate while decreasing learning time by approximately 22 % [20]. Likewise, in Android malware/or network-traffic threat detection contexts PCA has been used to reduce dimensionality and thereby reduce computational burden [21]. In contrast, ANOVA-F (a univariate statistical scoring method) can be used to evaluate each feature's ability to discriminate between malware and benign classes: features with high F-scores indicate strong between-class variance relative to within-class variance and thus have higher discriminative power. For example, a recent study in obfuscated malware detection used ANOVA to rank features and then built a deep neural network on the selected subset, obtaining extremely high precision and recall [22]. Moreover, comparative work in intrusion-detection domains shows that applying ANOVA-F feature selection improved classification performance (accuracy, sensitivity) by around 5-10% on average [23].

By selecting only the most informative features, ANOVA reduces noisy/redundant attributes, lowers the dimensionality, and thus improves model generalisation, reduces overfitting risk, and speeds up training and inference. Accordingly, in our study we applied both PCA and ANOVA feature-selection under the same pipeline to compare not only classification accuracy but also computational cost, thereby assessing which method yields a better trade-off in the context of Android malware detection.

3.1. Support vector machines

Support Vector Machines (SVMs) are widely used supervised learning algorithms for classification and regression tasks. Based on statistical learning theory, SVMs aim to find the optimal boundary that best separates different classes within a dataset [24]. The key concept behind SVMs is the hyperplane, a mathematical boundary that divides data points into different categories. In two-dimensional space, this boundary is a line; in three-dimensional space, it is a plane. The main objective of a Support Vector Machine (SVM) in classification problem is to find a hyperplane that optimises the margin, the distance between the hyperplane and closest data points of both classes also called support vectors. This can be increased to increase the ability of the model to generalise to new and unobserved data, which further increases the prediction ability of the model.

SVMs use a trick known as the kernel trick, to project input data into a higher-dimensional feature space, in which the linear method of separation is more easily attained. The most popular kernel functions are the linear kernel, the polynomial kernel, the radial basis function (RBF) kernel and the sigmoid kernel. The choice of a proper kernel is of essential importance because it depends on it directly what power of discrimination the model will have. The other key feature of SVMs is the so-called soft margin that adds flexibility by allowing a part of the data points to be within the limits of the margin or to be misclassified data. This is controlled by the regularisation parameter C , which regulates the trade-off between margin enlargement and classification errors. The higher the value of C , the more the right classification and the smaller the value, the more the margin is stressed.

SVMs have significant weaknesses even though they are effective. The computation cost of training an SVM may be computationally costly, particularly with large data sets, as the cost increases with the size of the training set. Moreover, SVMs do not necessarily give probabilistic motivation, and this may be a disadvantage in applications that need probabilities; methods like Platt scaling have been utilized to transform the SVM results to probabilities but this introduces more complexity to the model. Moreover, SVMs require careful tuning of kernel functions and hyperparameters to attain the best performance which is usually achieved through cross-validation [25].

3.2. Neural networks

Neural networks are a fundamental class of machine learning models inspired by the human brain. They consist of layers of interconnected neurons that process information by transmitting signals from one layer

to another [26]. This architecture enables neural networks to learn complex patterns and representations from data, making them highly effective for tasks like classification, regression, and pattern recognition. Each neuron in a neural network receives input signals, applies weights to them, sums them up, and passes the result through an activation function to produce an output. Common activation functions include sigmoid, hyperbolic tangent (tanh), and Rectified Linear Unit (ReLU), each introducing non-linearity to help the model learn complex relationships.

Neural networks are structured in layers:

- The input layer receives raw data.
- One or more hidden layers extract features and patterns.
- The output layer generates the final prediction.

Training a neural network involves backpropagation, an optimization technique combined with gradient descent. Backpropagation calculates the loss gradient for each weight in the network and updates them to minimize prediction errors. The loss function measures the difference between the model's predictions and the actual values, guiding the model to improve over time. One major advantage of neural networks is automatic feature extraction, reducing the need for manual feature engineering. This is particularly beneficial in fields where defining relevant features is complex [27].

However, neural networks have some challenges:

- They require large amounts of labelled data for training, which can be resource intensive.
- Training deep networks demands significant computing power, often requiring specialized hardware such as GPUs or TPUs.
- They are prone to overfitting, where the model learns patterns too well, including noise and outliers, leading to poor generalization to new data. Techniques such as dropout, early stopping, and regularization help mitigate overfitting.
- Neural networks are often viewed as black boxes, meaning their decision-making process is difficult to interpret. This lack of transparency can be problematic in fields like healthcare. Explainable AI (xAI) is an emerging field focused on improving the interpretability of neural networks [28].

3.3. Logistic regression

The Logistic Regression is one of the most significant and versatile algorithms to perform classification. It has its advantages in being simple, interpretable, and efficient to compute, which makes it a product vital to both statistical modelling and machine learning. The Logistic Regression is used not in regression problems but in estimating the probability that relates to a categorical dependent variable. It models the association between a dichotomous dependent variable and one or more independent ones by approximate probabilities with the help of a logistic (sigmoid) model. The logistic function is the essence of the Logistic Regression: it transforms the input of any real value into a range between 0 and 1; a feature that makes it fit the dichotomous outcomes like probability well [29].

The interpretability of Logistic Regression is one of its major strengths. The coefficients represent the change in the long odds of the outcome in case the predictor variable (x) is augmented by one unit. Exponentiation of these coefficients produces odds ratios which measures the degree of influence of a predictor on the probability of a certain outcome. Logistic Regression assumes a linear relationship between the independent variables and the long odds of the dependent variable. Although such simplification is beneficial in the interpretation, it might not capture the non-linear, complicated interactions present in the data. The way out of this weakness is to add terms of interaction or polynomials transformations to the model, but this comes at the price of more complexity and the risk of overfitting.

Despite its widespread application, Logistic Regression has several limitations:

- **Multicollinearity:** This occurs when there are strong correlations between the independent variables and the estimates of the variance become inflated thus causing the instability of the model. The

multicollinearity can be detected and addressed with the help of Variance Inflation Factor (VIF) analysis and Principal Component Analysis (PCA).

- Sensitivity to Outliers: Outliers can imbalance the estimates of coefficient, giving disproportionate predictions. This can be mitigated by strong estimation, variable transformations or by filtering out extreme values.
- Class Imbalance: Classification can be highly inaccurate when there is a gross imbalance in the data, with one category overwhelming the other. In these cases, oversampling of the minority group, undersampling of the majority group, or the artificial training of data by SMOTE can be used to improve the performance of the model [30].

3.4. ANOVA feature selection

Analysis of Variance (ANOVA) is a type of statistical technique used in feature selection to determine the most salient features to make in a model that can be used to predict. ANOVA helps to assess the existence of statistically significant differences between the means of certain groups, thus helping to identify which attributes demonstrate a strong correlation with the object of interest. ANOVA is the most beneficial when the number of features in a dataset is huge because it helps to reduce the dimensions and allows the model to focus on the most relevant attributes. The F-statistic measures the ratio of:

- Variance between groups (how much the means of distinct groups differs).
- Variance within groups (how much variability exists within each group).
- A high F-value suggests that the variance between groups is significantly greater than the variance within groups, indicating that the feature has strong discriminatory power.

ANOVA feature selection in machine learning involves the computation of F-statistical value of a single feature in relation to the target variable. F-statistic is a ratio of variance of groups to variance of groups. The F-statistic is normally distributed according to the F-distribution the null hypothesis given that the group means are equal. One may determine whether to reject the null hypothesis by comparing the calculated F-value, which is compared to a critical value based on the F-distribution or by checking the p-value. When the p-value is low (usually less than 0.05) this is an indication that one of the group means are significantly different and thus is statistically significant and should be included in the model. When the use of ANOVA feature selection is properly implemented, machine learning models could improve their predictive accuracy by removing redundant or irrelevant features, which eventually leads to efficiency and accuracy [31].

ANOVA is developed to find out the linear relationship; therefore, it might not always efficiently identify non-linear associations between the target variable and the feature. To address this shortcoming, the transformations of logarithmic, square root or Box-Cox can be used to make the data equal and stable the variances. Non-linear relationships and interactions between variables may be expressed as interaction terms or as poly features.

ANOVA-based feature selection is a useful method in estimating important features in the continuous-predictor datasets where the target variable is categorical. It both makes model structures simpler and reduces overfitting, as well as contributes to interpretability. Practitioners should, however, be careful about the assumptions and limitations of the technique taking care to preprocess and validate them. By properly using feature selection, along with other methodologies as well as domain knowledge, it is possible to create more efficient and effective machine learning models [32].

3.5. Principal component analysis

Principal Component Analysis (PCA) is a fundamental tool in the fields of statistics and machine learning, especially in features extraction and dimensionality reduction. It reduces a high dimensional set of variables into a smaller one without substantially losing most of the intrinsic information. PCA transforms correlated variables into a set of mutually uncorrelated objects (also referred to as principal components) to simplify high-dimensional data without losing any important trends and patterns. Essentially, PCA deals with the

variance and covariance of the variables and is interested in the definition of the major components that capture the highest variance. These main elements are orthogonal to each other hence guaranteeing that all of these elements depict different directions of variance. The first main component represents the biggest variance, and the next major component represents the biggest unexplainable variance, but it is orthogonal to the first major component [33].

Mathematically, there are a number of critical steps in PCA. The first step is to centre the data by subtracting the means of the variables so as to have zero mean, this process is done because during PCA, the data is quite sensitive to the relative scales between the original variables. A covariance data of the data is then obtained to explain inter-variable relationships. Next, eigenvalues and eigen vectors of the covariance matrix are calculated. The eigenvectors indicate the directions of the new feature space, and the eigenvalues indicate the size of the variance in the directions. Components are then ranked in descending eigenvalue order, with the most explaining components being given priority. A threshold is usually used e.g. 95 percent of total variance and the principal components that are required to reach this level of threshold are used to reduce dimensionality by eliminating components that contribute insignificant variance to the model. Finally, the original data are mapped on top of the selected set of main components, which results in a transformed dataset with a smaller number of dimensions but a more informative set of information.

PCA is another powerful feature extraction procedure in the sense that a strong focus is laid on what is most informative in data. PCA removes noise and redundancy by filtering directions with the largest variance, which makes machine learning algorithms more efficient, particularly, when the high dimensional data results in a computational overhead and a reduced predictive accuracy. PCA dimensionality reduction also helps to reduce the curse of dimensionality, in which a large feature space leads to data sparsity and noise to the performance of algorithms in identifying meaningful patterns. Although PCA has these strengths, it does not lack limitations. The key disadvantage is that it assumes a linear relationship between variables and might therefore be hindered in capturing non-linear patterns. Also, due to its acute sensitivity to difference in variable scales, PCA may give biased results when one variable has a significantly large scale and thus contributes to the principal components disproportionately. It is, therefore, mandatory to standardise or normalise the data before performing PCA because standardisation makes all the variables contribute equally and avoids the control of the larger scale variables. The other difficulty is the issue of interpretability: principal components are linear combinations of original variables; hence it may be problematic to understand their real-world meaning. The process of interpretation requires analysis of loadings that outline the contribution of each original variable; the more the number of variables, the more complex and labour intensive the analysis becomes to perform successfully [34].

Selection of the number of principal components to retain is a very critical decision. Oversized retention can be unable to cluster enough of the data together to significantly reduce dimensionality, and undersized retention can afford to lose a lot of information. Methods like scree plots, which plot eigenvalues versus component number, help in determining the juncture at which adding extra components to the model will add insignificant variance to the model. An optimal number of components to keep is represented by the point where the plot levels off - the elbow point. Although PCA is based on the assumption that the components with the highest variance are most important, this cannot always be valid with respect to the goals of a particular machine learning activity. In some situations, less variant variables can be more relevant to the target variable. As such, unsupervised learning situations with PCA typically involve more use of the technique, where the emphasis is on the description of data, as opposed to the prediction of a specific phenomenon.

4. Results and Discussion

This paper aimed to create and test machine-learning models to predict Android apps as malware or benign. Three different preprocessing conditions were taken into account: (1) the initially unprocessed dataset without reducing the features, (2) the usage of Principal Component Analysis (PCA) to extract the features, and (3) the usage of an ANOVA test to locate the features. Three classification algorithms Fine Gaussian Support Vector Machine (SVM), Wide Neural Network and Binary Generalized Linear Model (GLM) Logistic Regression were used to analyse each scenario. The current section enlarges on the dataset,

preprocessing tools, feature reduction methods, classification methods, and assessment measures utilized in the research.

In the former case, the entire 241 features were used without dimensionality reduction or feature selection methods in training the classification models. This strategy was to test the performance of the algorithms provided with the entire pool of available information. In the latter case, the training data were standardized and PCA was used to test the data in order to decrease the number of dimensions and still maintain most of the variance in the original features. The main elements were ranked based on their eigenvalues discerningly. However, since all of the extracted features exceeded the threshold value, all 241 features were used in the training set. The transformed training data was then used to train the classification models. The same transformation was applied to the testing data before evaluation.

In the third scenario, the ANOVA-test was utilized to select the most significant features related to the target variable. The procedure included 120 features into the training process. This approach aimed to improve model performance by eliminating irrelevant or redundant features.

Hyperparameters for each model were optimized using 5-fold cross-validation on the training set. The optimal parameters were selected based on the highest average cross-validation accuracy. In table 1, hyperparameters of the classification models are given.

Table 1. Hyperparameters of the classification algorithms

Classification Algorithm	Hyperparameters
Fine Gaussian SVM	Multiclass coding: One-vs-One Kernel function: Gaussian Kernel scale: 3.9 Box constraint level: 1 Standardize data: Yes
Wide Neural Network	Activation: ReLU Number of fully connected layers: 1 First layer size: 100 Iteration limit: 1000 Regularization strength (Lambda): 0 Standardize data: Yes
Binary GLM Logistic Regression	Solver: Linear Penalty: L2 C= 0.001 Maximum Iteration: 1000

Table 2 indicates the findings of the initial dataset, the PCA dataset, and the feature set that was picked using ANOVA. Its results showed that the SVM model performed better than the other models in all the datasets in terms of accuracy. Precisely, the SVM used an accuracy of 99.25 per cent on the PCA-applied dataset, which is slightly higher than the 99.10 per cent and 98.95 per cent on the original dataset and ANOVA feature selection set, respectively. The neural network showed slightly less accuracy with a maximum of 98.65 2 on the dataset run through PCA. The lowest accuracy was of logistic regression that reached a maximum accuracy of 98.50 amongst the three models with a high accuracy of 98.50 on the PCA-applied data.

Table 2. Results of the classification algorithms

Dataset	Model Type	Accuracy %	Training Time (sec)	Selected Features
PCA-Applied Dataset	SVM	99.25	39.9052339	241/241
Original Dataset	SVM	99.10	8.6394085	241/241
ANOVA Feature Selection	SVM	98,95	4.8080937	120/241
PCA-Applied Dataset	Neural Network	98.65	117.3555743	241/241
Original Dataset	Neural Network	98.50	7.,6136271	241/241
ANOVA Feature Selection	Neural Network	99.35	580952943	120/241
PCA-Applied Dataset	Logistic Regression	98.50	27.8751492	241/241
Original Dataset	Logistic Regression	98.05	26.0621809	241/241
ANOVA Feature Selection	Logistic Regression	97.90	20.6098871	120/241

The use of PCA led to a slight increase in accuracy of all the models. As an illustration, the accuracy in the use of the SVM rose to 99.10 99.25 percent on the original data set and 99.25 percent in PCA respectively. However, this diminutive improvement was accompanied by a significant increment in training times; SVM training time increased by 8.64s to 39.91s. Similar extensions of training times were found in the case of the neural-network model and logistic-regression model. It is worth noting that PCA was not used to reduce the dimensionality of the features since all the 241 principal components were retained to meet the variance-threshold requirement.

Conversely, feature-selection method based on ANOVA was successful in the minimization of features of 241 down to 120. Reduction in this dimension resulted into reduced training time of all models without compromising their accuracy. As an example, the SVM has maintained accuracy of 98.95 per cent and reduced training time by 8.64 to 4.81 per cent on the dataset with ANOVA-selected features. These findings indicate that the features that were left out were either useless or they did not play a role in the classification task.

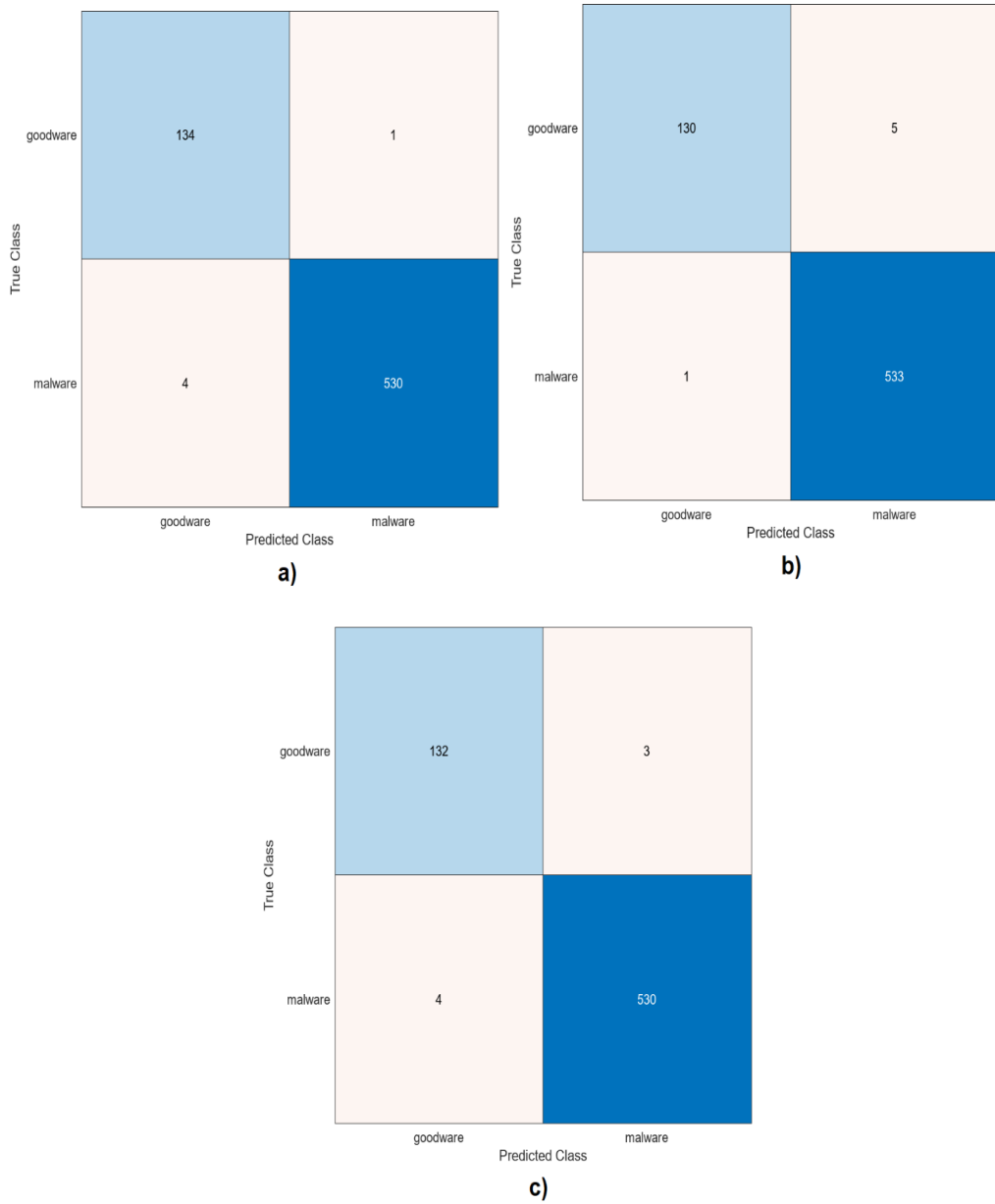


Figure 2. Confusion Matrices of the SVM models. (a) PCA-applied dataset, (b) Original dataset, (c) Anova feature selection

The breakdown of classification performance by each of the SVM models is provided in Figure 2 as represented by confusion matrices. The PCA model demonstrated the strongest results, with the ability to classify 3,535 malware samples (True Positives) and 896 benign ones (True Negatives), and only 1 False Positive and 4 False Negatives. These matrices show that PCA-applied dataset performed better to detect goodware applications.

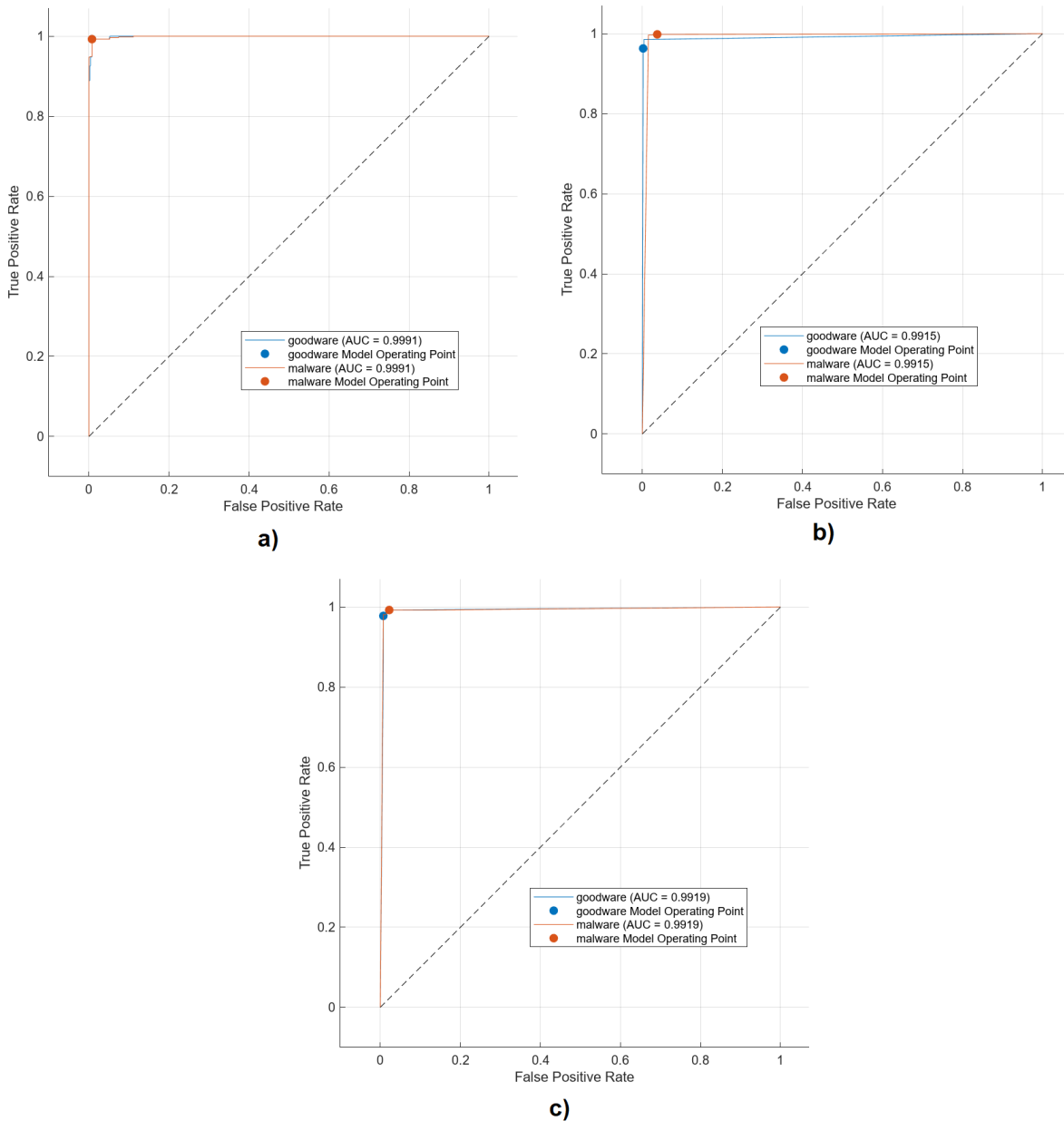


Figure 3. ROC curves of the SVM models. (a) PCA-applied dataset, (b) Original dataset, (c) Anova feature selection

Figure 3 illustrates the Receiver Operating Characteristic (ROC) curves for the SVM variations, plotting the True Positive Rate against the False Positive Rate. The Area Under the Curve (AUC) serves as a single metric for comparing the models' overall discriminative power. All three models show outstanding performance, with AUC values approaching a perfect score of 1.0. The proximity of all three curves to the top-left corner underscores the high sensitivity and specificity of the SVM classifier for malware detection using the TUANDROMD dataset.

In comparison, Alhogail & Alharbi achieved 97.82% accuracy for binary detection of Android malware using a chi-squared based feature selection and RF classifier on their dataset [35]. However, unlike our study, their work did not explicitly compare dimensionality-reduction approaches with a focus on computational cost. Our contribution thereby lies in providing a dual-axes evaluation (accuracy + efficiency) of PCA vs ANOVA under identical conditions.

5. Conclusions

Finally, the work also shows that Support Vector Machine models are more effective in distinguishing malware and benign Android applications with the highest predictive accuracy of the models compared in the paper. The feature selection of analysis of variance (ANOVA) is found as the pragmatic method of dimensionality reduction and cost-saving without compromising on performance. Conversely, the PCA did not provide many benefits in this specific context and instead caused a long period of training. Therefore, the focus should be on the method of the feature selection that identifies and exploits the most noticeable predictors to similar classification problems.

The future research directions should look into the implementation of more advanced machine-learning architectures, which could be more effective in gaining performance than is feasible with SVM. The research on alternative variance thresholds in PCA can help understand whether meaningful dimensionality reduction is possible to achieve, which may allow balancing accuracy and computational efficiency. Moreover, it is necessary to conduct stringent evaluation of overfitting risks, particularly in circumstances of high accuracy scores as indicated in this paper. While cross-validation mitigates this risk to some extent, validating the models on independent test datasets would strengthen the reliability of the results. Evaluating the models' inference times and more effective feature selection algorithms would also be beneficial, ensuring that they meet the real-time requirements of malware detection systems.

6. Author Contribution Statement

In the study, Author 1 contributed to forming the idea, making the design, literature review, contributed to the analysis of the results, provision of the materials and examination of the results.

7. Ethics Committee Approval and Conflict of Interest

There is no need for an ethics committee approval in the prepared article.
There is no conflict of interest with any person/institution in the prepared article.

8. Ethical Statement Regarding the Use of Artificial Intelligence

No artificial intelligence-based tools or applications have been utilized for the preparation of this study. All content of the study has been produced by the author in accordance with scientific research methods and academic ethical principles.

9. References

- [1] K. V. S. Bai and M. Thirumaran, "Hybrid deep learning and behavioral analysis for enhanced malware detection in banking," in *Proc. 8th Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA)*, 2024, pp. 1168–1173.
- [2] H. Y. Kwon, T. Kim, and M. K. Lee, "Advanced intrusion detection combining signature-based and behavior-based detection methods," *Electron.*, vol. 11, no. 6, p. 867, Mar. 2022.
- [3] S. Yakut, "Kayıplı resim sıkıştırma algoritmalarını temel alan rastgele sayı üretici," *Adıyaman Üniv. Müh. Bilim. Derg.*, vol. 9, no. 18, pp. 571–580, Dec. 2022.
- [4] S. Yakut, "Random number generator based on discrete cosine transform based lossy picture compression," *MTU J. Eng. Nat. Sci.*, vol. 2, no. 2, pp. 76–85, 2021.
- [5] S. Yakut, T. Tuncer, and A. B. Özer, "A new secure and efficient approach for TRNG and its post-processing algorithms," *Int. J. Bifurcation Chaos*, vol. 29, no. 15, May 2020.
- [6] S. Yakut, T. Tuncer, and A. B. Ozer, "Secure and efficient hybrid random number generator based on sponge constructions for cryptographic applications," *Elektron. Elektrotech.*, vol. 25, no. 4, pp. 40–46, Aug. 2019.
- [7] G. Areo, "Evaluating the efficacy of machine learning techniques in mitigating cybersecurity threats: A comprehensive analysis," 2024.
- [8] T. Adewale, "The role of deep learning in cloud-based cybersecurity solutions," 2024.
- [9] G. Cuiying *et al.*, "Uncovering and mitigating the impact of code obfuscation on dataset annotation with antivirus engines," in *Proc. 33rd ACM SIGSOFT Int. Symp. Softw. Test. Anal. (ISSTA)*, 2024, pp. 553–565.
- [10] S. S. Rajest and R. Regin, "Application of the CatBoost classifier for the detection of Android ransomware," *Central Asian J. Math. Theory Comput. Sci.*, vol. 5, no. 5, pp. 476–486, Nov. 2024.
- [11] D. Totham, V. Andersson, H. Thompson, and J. Whitmore, "Dynamic ransomware detection with adaptive encryption pattern recognition techniques," 2024.
- [12] M. A. Haq and M. Khuthaylah, "Leveraging machine learning for Android malware analysis: Insights from static and dynamic techniques," *Eng. Technol. Appl. Sci. Res.*, vol. 14, no. 4, pp. 15027–15032, Aug. 2024.
- [13] K. Kunku, A. N. K. Zaman, and K. Roy, "Ransomware detection and classification using machine learning," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, 2023, pp. 862–866.
- [14] G. Mubarak and S. Alasmari, "A framework to analyze OS systems artifacts from Linux machines," *Adv. Appl. Discrete Math.*, vol. 41, no. 8, pp. 603–640, Oct. 2024.
- [15] D. Gihavo, O. Ivanovich, A. Harrison, L. Merritt, and V. Schneider, "Automated file trap selection using machine learning for early detection of ransomware attacks," Authorea Preprints, Oct. 2024.
- [16] K. Dolesi, E. Steinbach, A. Velasquez, L. Whitaker, M. Baranov, and L. Atherton, "A machine learning approach to ransomware detection using opcode features and k-nearest neighbors on Windows," Authorea Preprints, Oct. 2024.
- [17] P. Borah, D. K. Bhattacharyya, and J. K. Kalita, "Malware dataset generation and evaluation," in *Proc. 4th IEEE Conf. Inf. Commun. Technol. (CICT)*, 2020.
- [18] A. Alhudhaif, B. Almaslukh, A. O. Aseeri, O. Guler, and K. Polat, "A novel nonlinear automated multi-class skin lesion detection system using soft-attention based convolutional neural networks," *Chaos Solitons Fractals*, vol. 170, p. 113409, May 2023.
- [19] F. Ucar, O. F. Alcin, B. Dandil, and F. Ata, "Machine learning based power quality event classification using wavelet-entropy and basic statistical features," in *Proc. 21st Int. Conf. Methods Models Autom. Robot. (MMAR)*, 2016, pp. 414–419.
- [20] J. Zhang, "Machine learning with feature selection using principal component analysis for malware detection: A case study," arXiv preprint, Feb. 2019.
- [21] D. Ö. Şahin, O. E. Kural, S. Akleylek, and E. Kılıç, "Permission-based Android malware analysis by using dimension reduction with PCA and LDA," *J. Inf. Secur. Appl.*, vol. 63, p. 102995, Dec. 2021.
- [22] M. Hadjila, M. Merzoug, W. Ferhi, D. Moussaoui, A. B. Boudaine, and M. H. Hachemi, "Obfuscated malware detection using deep neural network with ANOVA feature selection on CIC-MalMem-2022 dataset," *Sci. Tech. J. Inf. Technol. Mech. Opt.*, vol. 24, no. 5, pp. 849–857, Sep. 2024.

- [23] M. J. Siraj, T. Ahmad, and R. M. Ijtihadie, “Analyzing ANOVA F-test and sequential feature selection for intrusion detection systems,” *Int. J. Adv. Soft Comput. Appl.*, vol. 14, no. 2, pp. 185–194, 2022.
- [24] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, pp. 273–297, 1995.
- [25] S. Buyrukoğlu, “New hybrid data mining model for prediction of Salmonella presence in agricultural waters based on ensemble feature selection and machine learning algorithms,” *J. Food Saf.*, vol. 41, no. 4, p. e12903, Aug. 2021.
- [26] B. Karakaya, V. Çelik, and A. Gülten, “Chaotic cellular neural network-based true random number generator,” *Int. J. Circuit Theory Appl.*, vol. 45, no. 11, pp. 1885–1897, 2017.
- [27] O. Güler, “Turbofan motorlarının kestirimci bakımında makine öğrenimi algoritmaları performanslarının karşılaştırılması,” *Niğde Ömer Halisdemir Üniv. Müh. Bilim. Derg.*, vol. 13, no. 1, pp. 99–106, Jan. 2024.
- [28] R. K. Sheu, M. S. Pardeshi, K. C. Pai, L. C. Chen, C. L. Wu, and W. C. Chen, “Interpretable classification of pneumonia infection using explainable AI (XAI-ICP),” *IEEE Access*, vol. 11, pp. 28896–28919, 2023.
- [29] F. Türk, “Investigation of machine learning algorithms on heart disease through dominant feature detection and feature selection,” *Signal Image Video Process.*, vol. 18, no. 4, pp. 3943–3955, Jun. 2024.
- [30] S. Ahmadov and A. Boyacı, “Multilingual text mining based open source emotional intelligence,” *Turkish J. Sci. Technol.*, vol. 17, no. 2, pp. 161–166, Sep. 2022.
- [31] H. A. Kwaider and E. Avaroğlu, “Threats detection in IoT network,” *Turkish J. Sci. Technol.*, vol. 18, no. 1, pp. 113–122, Mar. 2023.
- [32] A. Bilen and A. B. Özer, “Siber saldırılar için rastgele orman algoritması kullanılarak öznelik seçimi,” *Firat Üniv. Fen Bilim. Derg.*, vol. 34, no. 1, pp. 31–37, Mar. 2022.
- [33] A. İmak, “Automatic classification of defective photovoltaic module cells based on a novel CNN-PCA-SVM deep hybrid model in electroluminescence images,” *Turkish J. Sci. Technol.*, vol. 19, no. 2, pp. 497–508, Sep. 2024.
- [34] İ. Riaz, N. Mushtaq, M. M. Jillani, and U. Navaz, “Performance analysis of Pakistan Super League players using principal component analysis approach,” *Sci. J. Mehmet Akif Ersoy Univ.*, vol. 2, no. 4, pp. 127–135, Dec. 2019.
- [35] A. Alhogail and R. A. Alharbi, “Effective ML-based Android malware detection and categorization,” *Electron.*, vol. 14, no. 8, p. 1486, Apr. 2025.