BİLGİSAYAR BİLİMLERİ VE TEKNOLOJİLERİ DERGİSİ
JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGIES
http://dergipark.org.tr/tr/pub/bibted

Araştırma Makalesi

Makine Öğrenimi Yöntemlerini Kullanarak Kötü Amaçlı Yazılımların Statik Analiz ile Tespiti

Nisa VURAN SARI*[1], Mehmet ACI[2]

[1]Mersin Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği, Mersin, Türkiye
[2] Mersin Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği, Mersin, Türkiye

Anahtar Kelimeler:
Siber Güvenlik
Zararlı Yazılım Tespiti
Zararlı Yazılım Analizi
Makine Öğrenmesi

ÖZ

Siber saldırılardaki artış internet ve bilişim teknolojileri kullanımını da tehdit etmeye başlamıştır. Bu durum, siber saldırılardan sorumlu kötü amaçlı yazılımları tespit etmenin önemini vurgulamaktadır. Günümüzde, kötü amaçlı yazılımları algılamak için makine öğrenmesi yöntemlerinin geliştirilmesi üzerine çalışmalar bulunmaktadır. Kötü amaçlı yazılım dedektörleri, kötü amaçlı yazılımlara karşı savunmada birincil araçlardır. Böyle bir dedektöderün kalitesi, kullandığı tekniklerle belirlenir. Makine öğrenmesi, derin öğrenme ve statik ve dinamik analiz gibi zararlı yazılım analiz yöntemleri bu teknikler arasında yer almaktadır. Bu çalışma kötü amaçlı yazılım analizi ve sınıflandırma tekniklerini sunmaktadır. Kötü amaçlı yazılım tespiti için, K-En Yakın Komşular, Saf Bayes, Karar Ağaçları ve Rastgele Orman gibi iyi bilinen makine öğrenmesi algoritmaları kullanılmıştır. Çalışma, Karar Ağaçları sınıflandırma tekniği kullanımının %97,75 sınıflandırma ile en iyi doğruluğu ürettiğini, Saf Bayes'in ise %53 ile en düşük doğruluğu ürettiğini göstermektedir.

Detection of Malware by Static Analysis Using Machine Learning Methods

Keywords:
Cyber Security
Malware Detection
Malware Analysis
Machine Learning

ABSTRACT

The increase in cyber-attacks has also started to threaten the use of internet and information technologies. This situation emphasizes the importance of detecting malicious software that is responsible for cyber-attacks. Nowadays, there are studies on the development of machine learning methods for malicious software detection. Malicious software detectors are the primary tools in defense against malicious software. The quality of such a detector is determined by the techniques it uses. Malware analysis methods such as machine learning, deep learning, and static and dynamic analysis are among these techniques. This study presents malware analysis and classification techniques. For malware detection, well-known algorithms for machine learning including such K-Nearest Neighbors, Naive Bayes, Decision Trees, and Random Forest were used. The research shows that the use of Random Forest classification technique produces the best accuracy with 97.75% classification, while Naive Bayes produces the lowest accuracy of 53%.

***Sorumlu Yazar**

*(nvuran@mersin.edu.trl) ORCID ID 0000–0001–7042–3031
(maci@mersin.edu.tr) ORCID ID 0000–0002–7245–8673

e-ISSN: 2717-8579
Geliş Tarihi: 06/06/2023; Kabul Tarihi: 08/08/2023

1. INTRODUCTION

In recent years, there has been an increase in the development of malware. Today, cyber attackers use malware for attacks on information systems. Internet environments such as e-mails, malicious websites, downloaded software are the main environments for performing a malware attack on information systems. Detection and analysis of these malicious software is important for information security and healthy internet usage. In general, 3 basic techniques (static, dynamic and hybrid) are examined for the analysis of malicious software. Analyzes extract different features to classify and detect malicious and harmless files. Static and dynamic methods can be used to automate and speed up the steps in malware analysis, detection and classification. Extracting different malware features through analytical techniques is critical to the success of malware detection. Data from static or dynamic methods can be used to detect malware or classify malware by families using machine learning techniques (i.e., clustering or classification).

Analyzing malware without running it is called static analysis. Patterns used in static analysis include string signature, syntactic library call, byte array n-grams, opcode (operation code) frequency distribution, control flow diagram, and so on (Gandotra et al., 2014). Analysis of the behavior (interaction with the system) of a malware while running in a controlled environment such as a virtual machine, sandbox, simulator, emulator, and so on is called dynamic analysis. Before running the malware sample, appropriate monitoring tools such as Process Monitor, Capture BAT, Process Hacker, Process Explorer, Regshot and Wireshark are installed and enabled (Gandotra et al., 2014). Despite all its benefits, the biggest disadvantage of dynamic analysis is the performance cost. When dealing with large datasets containing hundreds of binaries, dynamic analysis cannot scale effectively (Hassen et al., 2017). Since scalability is a need of this research, static analysis is used to extract features from the malicious programs.

Although traditional malware classification methods were used before, with the rapid spread and complexity of malware, there is a need to develop new methods. Machine learning methods are also powerful technologies that transform the effectiveness of cyber security research. In this study, Random Forest (RF), Decision Tree (DT), K-Nearest Neighbor (KNN) and Gaussian Naive Bayes (NB) machine learning methods were used to classify malware. Calculated accuracy, f1-score, precision, and recall metrics are compared to determine success of the methods. Among the methods used, it is observed that the RF algorithm gave the most successful result with an accuracy rate of 97.75%.

Malware has posed a great threat to computer systems from past to present. New techniques and methods have been tested by adding new malware detection studies that have been carried out since the past. It has been observed that artificial intelligence-assisted methods such as machine learning and deep learning methods improve malware detections and outperform existing methods, with the efficiency of existing methods decreasing against new threats. In the continuation of this section, a summary of the studies on malware detection in the literature is presented.

Tian et al. (Tian et al., 2009) presented malware classification research using classification methods based on sequence information. Sequences were acquired from 1367 samples, including unpackaged trojans, viruses, and clean files. Several classification techniques, including tree-based classifiers, KNN, statistical algorithms, and AdaBoost were used to analyze the information identifying the sequences presented in each sample. Using k-fold cross validation on unpackaged malicious and benign files, the RF classifier achieved an accuracy rate of 97%.

Santos et al. (Santos et al., 2013) offered a new hybrid malware detector that merges the frequency of occurrence of operational codes (statically collected) with information about an executable's execution track (dynamically obtained). This hybrid strategy has been found to increase the performance of both methods when performed individually. KNN, DT, RF, SVM (Support Vector Machine), and NB methods were used in this work. Static, dynamic, and hybrid techniques were evaluated separately. The SVM (Normalized Polynomial Kernel) achieved the highest accuracy for each approximation (Dynamic: %77.26, Hybrid: 96.60%, and Static: 95.90%).

Patil et al. (Patil and Deng, 2020) used a method to extract various feature sets from malware data such as system calls, opcodes, and bytecodes. Work has been done on Microsoft's malware dataset available on the Kaggle website. The dataset contains 10868 malicious files from nine different malware families. The study examines the effectiveness of machine learning algorithms (i.e., DT, RF, NB, KNN, Support Vector Classifier (SVC), Stochastic Gradient Descent (SGD), Logistic Regression (LR)) and deep learning-based (i.e., Deep Neural Network (DNN)) models. Each of the features in the dataset is examined and the findings show that the feature vector for system calls achieves the highest accuracy.

Azeez et al. (Azeez et al., 2021) proposed a method based on collective learning. After applying the CNN classifier, various machine learning algorithms were applied for the final stage. For comparison, RF, NB, DT, GB and Adaboost algorithms were used. The RF algorithm yielded the highest result with an accuracy rate of 99.24%.

A literature review of existing malware detection classification researchs utilizing machine learning techniques was published by Harshalatha et al. (Harshalatha and Mohanasundaram, 2020).

Random Forest (RF), Support Vector Machine (SVM), BayesNet and Multi-Layer Perceptron (MLP) are the classifier models utilized in this study. First, all malware samples were put through a 10 fold cross validation test, and the findings of the classifier showed that RF performed the best with 98% accuracy. It was shown that using the RF classifier for various datasets caused the same RF accuracy to drop by 12%.

Tahtacı and Canbay (Tahtacı and Canbay, 2020) investigated Android-based files with a machine learning model along with N-Gram features. The models were combined with methods to determine the threshold of variance and obtain features information. With the APKTool tool, 3000 Android Package (APK) files were converted to source code and the opcodes of the programs were obtained. DT, KNN, NB, LR, RF and SVM machine learning algorithms are trained using N-Gram features. The best test result is obtained after reducing the number of features in 3-gram to two (100% test score was obtained in the KNN and SVM models).

Table 1. Summary of the related studies on malware detection and their results.

Author	Year	Classifiers	Best Classifier	Accuracy
Tian et al.	2009	RF, NB, DT, IB1 and SVM	RF	0.97
Santos et al.	2013	KNN, DT, RF, SVM and NB	SVM	Static:0.9590 Dynamic:0.7726 Hybrid: 0.9660
Patil et al.	2020	DT, SGD, RF, SVC, LR, DNN, NB, KNN	DNN	>0.95
Tahtacı and Canbay	2020	DT, KNN, NB, LR, RF and SVM	KNN and SVM	1.0
Harshalatha et al.	2020	RF, BayesNet, MLP ve SVM	RF	0.98
Azeez et al.	2021	CNN, RF, NB, DT, GB and Adaboost	RF	0.99

The article continues as follows. The second chapter highlights technique, which includes machine learning-based malware classification and detection methods and dataset preprocessing. The third chapter analyzes machine learning malware detection algorithms and evaluates the findings. This topic is concluded in Chapter 4 by addressing the research aspects.

2. METHODS

The main purpose of this study is to detect malicious software with a data set created by static analysis method. This section contains information about the dataset and machine learning algorithms used for malware analysis.

2.1. Dataset and Data Preprocessing

The data set was obtained from C-Prot Turkey. It consists of 1000 pieces of software labeled as malicious (malware) and 1000 pieces of software labeled as benign (non-malware). In order to extract the data set features used in the study, the features from the files of programs were obtained by applying the static analysis method. The sample fragment of the dataset is shown in table 1. The file size, digital signature status, libraries and functions used in the software were used as dataset features of each software. In this study, the data set was divided into two sub-datasets and 80% (1600 pieces) of the data were used for training and 20% (400 pieces) for testing.

Table 1. A sample dataset before the dataset preprocessing step.

SIZE	DIGITAL SIGNATURE	LIBRARIES	FUNCTIONS	LABEL
323800	1	KERNEL32.dll,USER32.d...	GetSystemTime,PostMessage,FreeLib…	1
6595084	0	MSVCR100.dll,IMM32.dl...	MD5,SHA,SHA1,HMAC,GetSe...	1
139084	0	lib1.dll,bcrypt.dll,KERNEL3...	BCryptOpenAlgorithmProvider,BC...	0
119160	0	kernel32.dll,user32.dll	ExitProcess,VirtualAlloc,VirtualFree…	1
560906	0	uxtheme.dll,gdi32.dll,kern...	CreateWindow,CreateWindowEx,Ge…	1
162733	0	zlib1.dll,bcrypt.dll,KERN…	BCryptOpenAlgorithmProvider,BCryp…	0
511552	1	VCRUNTIME140.dll,KER…	SHA,InitializeConditionVariable…	0

Among the features in the dataset, each of the functions and libraries in the functions and libraries (features) column has been made a separate feature, so that each function and library name is also considered a dataset feature. As a result of this process, 195 unique libraries and 4,327 unique functions were obtained. Including size, digital signature, label, and these unique functions and libraries were also used as features in the dataset. Thus, a total of 4,525 features (4,525 columns) were obtained. Table 2 shows sample dataset after the dataset preprocessing step.

The data preprocessing step transforms the data into a format that machine learning algorithms can analyze efficiently (Markel et al., 2014). In order for the models to work efficiently in machine learning algorithms, categorical inputs need to be converted into numerical expressions. For this reason, normalizing all the features in the dataset to binary values as "0" or "1" provides an advantage in the training process of the model. In this study values of 1 and 0 (value of 1 presence of the feature in the software, 0 value of the absence of the feature in the software) were assigned for each feature according to the content of the software.

Table 2. A sample dataset after the dataset preprocessing step.

SIZE	DIGITAL SIGNATURE	.ctor	KERNEL32.dll	zlib1.dll	GetSystemTime	ExitProcess	LABEL
323800	1	0	1	0	0	0	1
6595084	0	1	0	0	0	0	1
139084	0	0	0	0	1	0	0
119160	0	0	0	0	0	1	1
560906	0	0	0	0	0	0	1
162733	0	0	1	0	0	0	0
511552	1	0	1	1	0	0	0

2.2. Machine Learning Algorithms

With the proliferation of new and unseen malware families, there is a need to develop new methods to detect malware. Machine learning is one of these methods. In this study, well-known machine learning algorithms such as KNN, RF, DT and Gaussian NB were trained and tested to detect malware. The theoretical details of the machine learning methods used in this study are given below.

2.2.1. Gaussian naive bayes

The NB classifier is a Bayesian-based probabilistic classification mechanism. The main goal of classification is to find the best match between a set of new data and a set of classifications within a specific problem domain (Yang, 2018). The relationship in is stated by Bayes' theorem with respect to the class variables f_1 and f_n and the dependent feature vector (1).

C represents the given target, and f represents the features.

$$p(C|\boldsymbol{f_1}, \dots \boldsymbol{f_n}) = \frac{P(C)p(\boldsymbol{f_1},\dots \boldsymbol{f_n}|C)}{p(\boldsymbol{f_1},\dots \boldsymbol{f_n})} \qquad (1)$$

In simple Bayesian classification, the Gaussian distribution is a method of using continuous features. If the feature has continuous values, they are from a Gaussian or normal distribution.

$$\boldsymbol{p}(\boldsymbol{x_i} \mid \boldsymbol{y}) = \frac{\mathbf{1}}{\sqrt{\mathbf{2}\pi\boldsymbol{\sigma_y^2}}}\ \boldsymbol{exp}\left(-\frac{(\boldsymbol{x_i}-\boldsymbol{\mu_y})^2}{\mathbf{2}\boldsymbol{\sigma_y^2}}\right) \qquad (2)$$

Parameters σ(y) and μ(y) shown in formula (2) are estimated using maximum probability.

2.2.2. K-nearest neighbors

Based on the supervised learning technique, KNN is one of the Machine Learning algorithms used for both classification and regression. Based on the similarity between the new state data and the existing states, it classifies the new state into the category to most similar to the existing classes. It computes the distance between the test data and all training points and attempts to predict which category the test data belongs to.

Algorithm selects the number of K points close to the test data. 'K' value computes the probability of test data belonging to training data classes, and then the class with the highest probability is chosen.

To find the nearest neighbors, various distance measurement methods are used (Chumachenko, 2017). The Hamming, Manhattan, Minkowski, and Euclidean distances are all popular. Minkowski distance was used as a distance measure in this study. It then discards the point from among the k nearest neighbors to the class (where k is an integer).

Where x=$(x_1, x_2, x_3, \ldots x_n)$ and y=$(y_1, y_2, y_3, \ldots y)$ and p is an integer.

$$\boldsymbol{D}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} (p_i - q_i)^{\frac{1}{p}} \quad (3)$$

2.2.3. Decision tree

A DT is a recursively expressed machine learning classifier. It consists of nodes that form a root tree (Rokach, Maimon, 2005). The root and internal nodes, branches, and leaf nodes of decision trees have a hierarchical tree structure. It starts with a node and grows a tree structure by adding branches when new results are obtained. The result is achieved by traversing the nodes using the entered value. Figure 1 depicts an example DT.

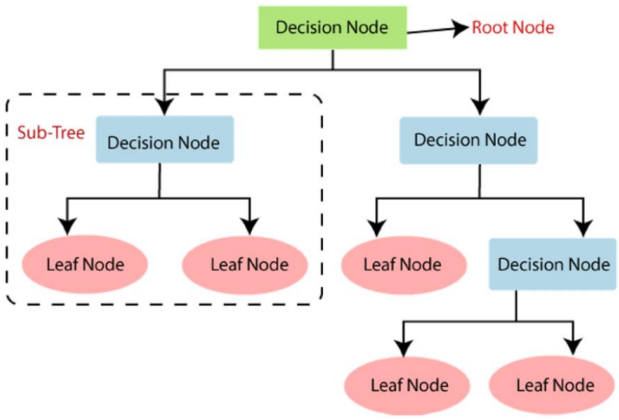

Figure 1. Representation of DT algorithm (Deshpande, 2021).

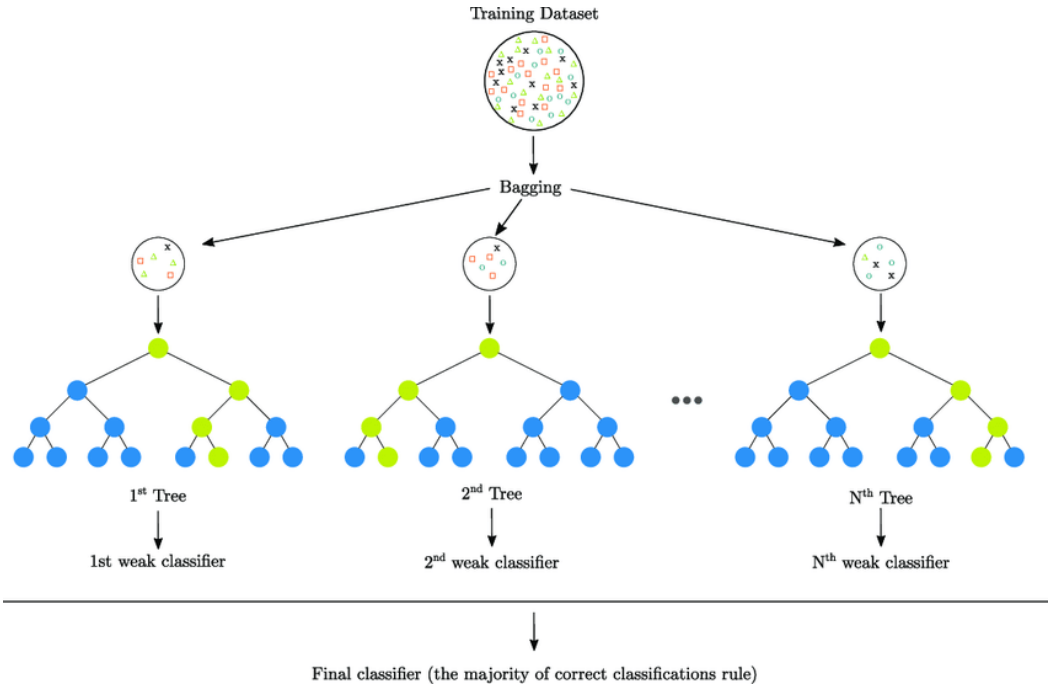

Figure 2. Representation of RF algorithm (Deshpande, 2021).

2.2.4. Random forest

Random forests are a combination of tree estimators that contain a set of DTs in different subsets of a given dataset, where each tree is sampled independently and depends on the values of a random vector with the same distribution for all trees in the forest (Breiman, 2001).

It does not depend on the output of a single decision tree, but aggregates the predictions generated from each tree and predicts the final result based on the majority of these predictions. Increasing the number of trees in the forest prevents overfitting and improves accuracy. Figure 2 presents the RF algorithm.

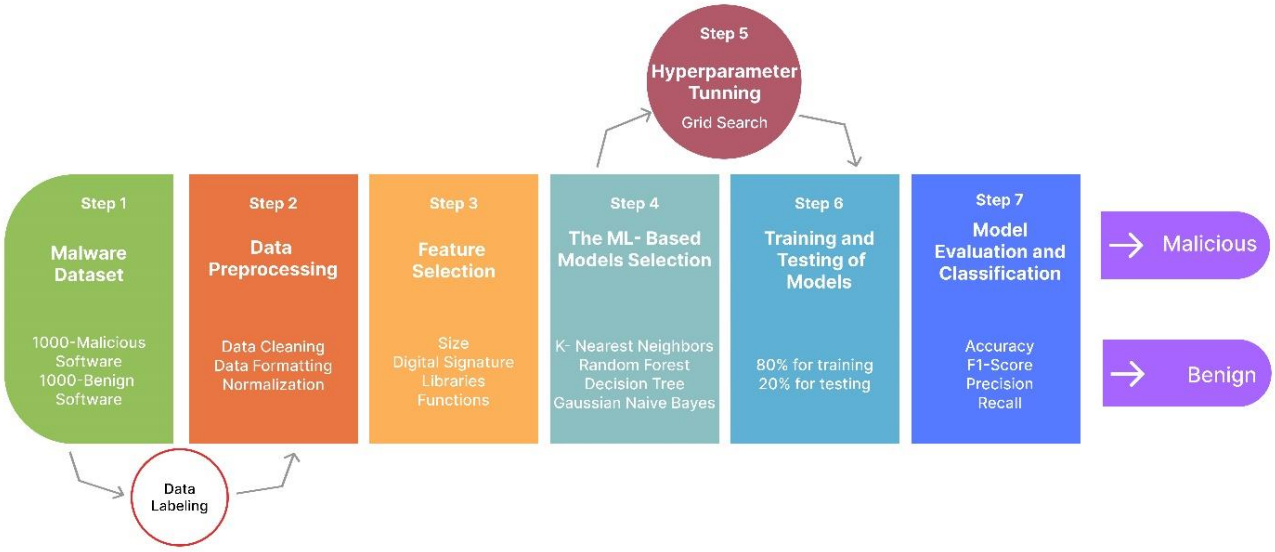

Figure 3. The overall design of malware analysis system used in the study.

3. FINDINGS

All classifiers used in the study were developed and tested with the python-based Scikit learn library (URL-1). The results of the models, trained with KNN, DT, RF and Gaussian NB algorithms, were evaluated with accuracy, f1-score, recall and precision metrics.

Grid search method was applied for hyperparameter optimization during the training of classifiers. The hyperparameters to be tested in the model and the determined value ranges are shown in Table 3. In the grid search method, a model is built with all combinations separately and the model is trained for determining the most successful hyperparameter set according to the specified metric. Grid search hyperparameter optimization technique has been observed to increase model performance. The overall design of malware analysis system used in the study is shown in figure 3.

According to the results; RF (97% - Highest) and DT (94%) algorithms achieved the highest accuracy rate, while NB (53% - Lowest) and KNN (59%) algorithms achieved lower results. RF and DT are more effective than KNN and Gaussian NB for a variety of reasons because they are tree-based algorithms. One argument for this is that the tree structure they produce tends to reduce the inaccuracy rate relative to the previous tree. It is thought that the NB algorithm gives weaker classification results compared to other classification algorithms due to the independent determination of the class values of the features. Figure 3 shows the comparison graph of accuracy, f1-score, precision and recall values of algorithms.

RF and DT algorithms gave the highest accuracy results. There are many reasons that RF algorithm gave higher accuracy than DT. One of the important reason is that random forest algorithm finds the root node and assigns the nodes randomly. Rather than feeding a decision tree with all the data, it can generate a random forest by feeding it piecemeal and multiple DTs.

From Table 5, it is seen that the f1-score, precision and recall values are in similar proportions with the accuracy values of the algorithms. In addition, the high precision and recall values of the DT and RF algorithms prove that they can accurately distinguish between malicious and benign software

Table 3. The hyperparameter values for machine learning algorithms using grid search method.

	Hyperparameter	**Value Range**	**Description**
KNN	N_neighbours Distance Metric	(1,50) Minkowski	Number of neighbors Metric to use for distance computation.
RF	Max_depth Max_features N_estimators **criterion**	(1,10) [3, 5, 10, 15] [100, 200, 500, 1000, 2000] gini	The maximum depth of the tree. The number of features when looking for the best split. The number of trees in the forest. The function to measure the quality of a split.
DT	Max_depth	(1,10)	The tree's maximum depth.

	Min_samples_split	(2,50)	The minimum number of samples required to split an internal node
	criterion	gini	The function to measure the quality of a split.
NB	**priors**	**default**=None	Prior probabilities of the classes.
	var_smoothing	**default**=1e-9	Portion of the largest variance of all features that is added to variances for calculation stability

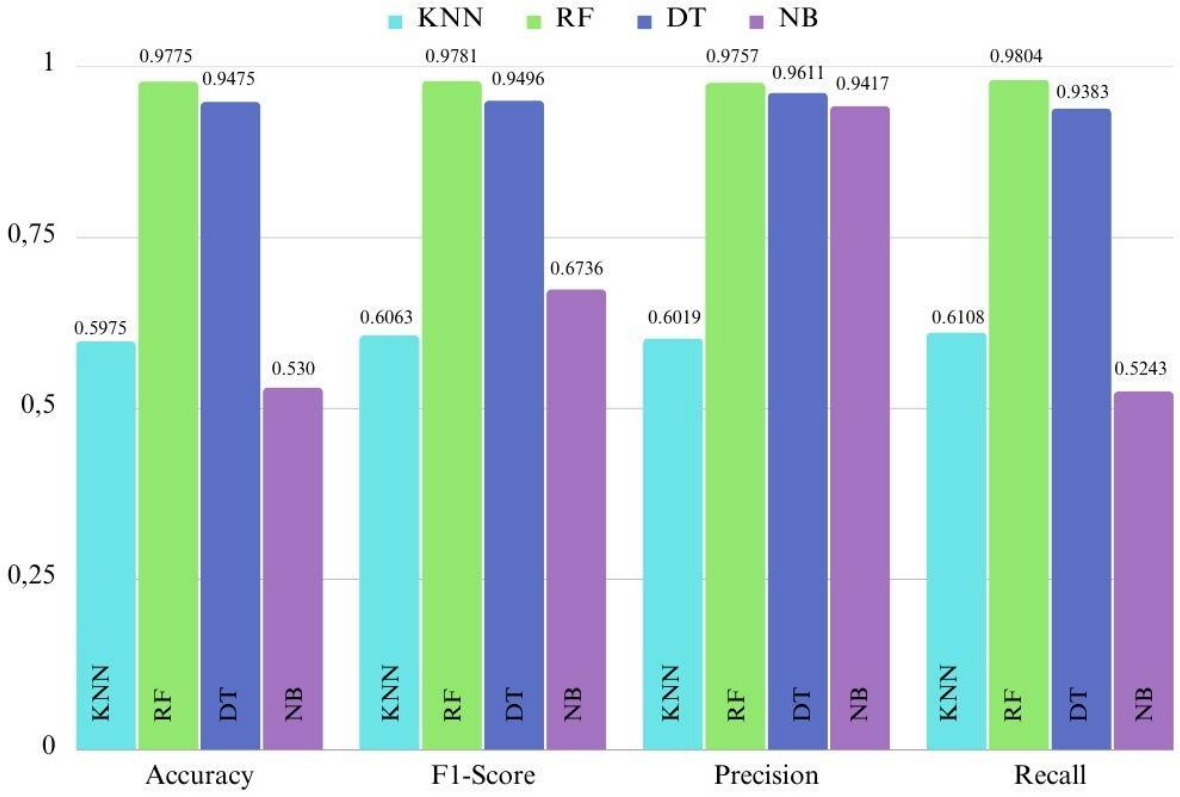

Figure 3. The comparison graph of accuracy, f1-score, precision and recall values of algorithms.

Table 5. Accuracy, f1-score, precision and recall values of algorithm.

Algorithm	Accuracy	F1-Score	Precision	Recall
KNN	0.5975	0.6063	0.6019	0.6108
RF	**0.9775**	**0.9781**	**0.9757**	**0.9804**
DT	0.9475	0.9496	0.9611	0.9383
NB	0.530	0.6736	0.9417	0.5243

The difference between the macro and micro average performance measures is that the macro average weights each class equally, while the micro average weights each feature equally. When Figure 5 and Table 5 are examined, it is seen that the balanced sample numbers of the classes in the data set cause the macro and micro averages to result in the same or very close results.

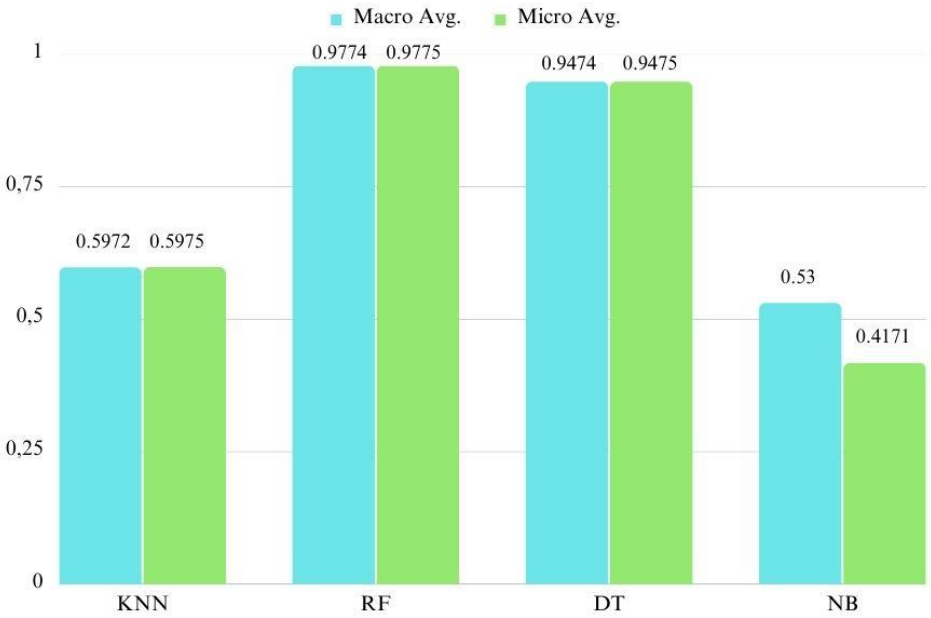

Figure 5. The comparison graph of macro average and micro average values of algorithms.

The RF algorithm gave the best results with 97.75% micro average and 97.74% macro average value.

Table 5. Macro average and micro average values of algorithm.

Algorithm	Macro Average	Micro Average
KNN	0.5972	0.5975
RF	**0.9774**	**0.9775**
DT	0.9474	0.9475
NB	0.53	0.4171

4. RESULTS

The main purpose of this paper is to detect malicious software using static analysis and machine learning methods (i.e., DT, KNN, RF, NB). The data obtained by static methods were processed and converted into a suitable format for training with machine learning algorithms, and then the models were trained with this data set and analyzes were carried out. The experimental results of the trained and tested machine learning algorithms are compared. It has been seen that tree-based DT and RF algorithms show high performance in malware analysis.

In future work, the dataset can be trained by developing deep learning-based models to improve accuracy, f1-score, precision and recall metrics and higher malware detection rates.

APPENDIX

We would like to thank C-Prot Turkey Company for sharing the dataset used in this study.

REFERENCES

Azeez, N. A., Odufuwa, O. E., Misra, S., Oluranti, J., & Damaševičius, R. (2021). Windows PE malware detection using ensemble learning. In *Informatics* (Vol. 8, No. 1, p. 10). MDPI.

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5-32.

Chumachenko, K. (2017). Machine learning methods for malware detection and classification.

Deshpande, N. M., Gite, S., & Aluvalu, R. (2021). A review of microscopic analysis of blood cells for disease detection with AI perspective. *PeerJ Computer Science*, *7*, e460.

Gandotra, E., Bansal, D., & Sofat, S. (2014). Malware analysis and classification: A survey. *Journal of Information Security*, *2014*.

Harshalatha, P., & Mohanasundaram, R. (2020). Classification Of Malware Detection Using Machine Learn-ing Algorithms: A Survey. *International Journal of Scientific & Technology Research*, *9*(02).

Hassen, M., Carvalho, M. M., & Chan, P. K. (2017, November). Malware classification using static analysis based features. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1-7). IEEE.

Maimon, O., & Rokach, L. (Eds.). (2005). Data mining and knowledge discovery handbook.

Markel, Z., & Bilzor, M. (2014, October). Building a machine learning classifier for malware detection. In *2014 second workshop on anti-malware testing research (WATeR)* (pp. 1-4). IEEE.

Patil, R., & Deng, W. (2020, March). Malware analysis using machine learning and deep learning techniques. In *2020 SoutheastCon* (Vol. 2, pp. 1-7). IEEE.

Santos, I., Devesa, J., Brezo, F., Nieves, J., & Bringas, P. G. (2013). Opem: A static-dynamic approach for machine-learning-based malware detection. In *International joint conference CISIS'12-ICEUTE´ 12-SOCO´ 12 special sessions* (pp. 271-280). Springer Berlin Heidelberg.

Sapountzoglou, N., Lago, J., & Raison, B. (2020). Fault diagnosis in low voltage smart distribution grids using gradient boosting trees. *Electric Power Systems Research*, *182*, 106254.

TAHTACI, B., & CANBAY, B. (2020, October). Android malware detection using machine learning. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1-6). IEEE.

Tian, R., Batten, L., Islam, R., & Versteeg, S. (2009, October). An automated classification system based on the strings of trojan and virus families. In *2009 4th International conference on malicious and unwanted software (MALWARE)* (pp. 23-30). IEEE.

Yang, F. J. (2018, December). An implementation of naive bayes classifier. In *2018 International conference on computational science and computational intelligence (CSCI)* (pp. 301-306). IEEE.

URL-1: https://scikit-learn.org/stable/
[Erişim Tarihi: 15.05.2023]