KADIR HAS UNIVERSITY
SCHOOL OF GRADUATE STUDIES
DEPARTMENT OF
ADMINISTRATIVE SCIENCES

ENSEMBLE OF FEATURE SELECTION MODELS FOR MALWARE DATASETS

FARUK CÜREBAL

MASTER OF SCIENCE

ISTANBUL, SEPTEMBER, 2022

ENSEMBLE OF FEATURE SELECTION MODELS FOR MALWARE DATASETS

FARUK CÜREBAL
ADVISOR: Prof. Dr. HASAN DAĞ

A thesis submitted to
the School of Graduate Studies of Kadir Has University
in partial fulfilment of the requirements for the degree of
Master of Science in
Cyber Security

Istanbul, September, 2022

APPROVAL

This thesis titled ENSEMBLE OF FEATURE SELECTION MODELS FOR MALWARE DATASETS submitted by FARUK CÜREBAL, in partial fulfillment of the requirements for the degree of Master of Science in Cyber Security is approved by

Prof. Dr. Hasan Dağ (Advisor) .....
Kadir Has University

Prof. Dr. Mustafa Bağrıyanık .....
Istanbul Technical University

Assist. Prof. Dr. E. Fatih Yetkin .....
Kadir Has University

I confirm that the signatures above belong to the aforementioned faculty members.

.....
Prof. Dr. Mehmet Timur Aydemir
Dean of School of Graduate Studies
Date of Approval: 06.09.2022

DECLARATION ON RESEARCH ETHICS AND PUBLISHING METHODS

I, FARUK CÜREBAL; hereby declare

- that this Master of Science Thesis that I have submitted is entirely my own work and I have cited and referenced all material and results that are not my own in accordance with the rules;
- that this Master of Science Thesis does not contain any material from any research submitted or accepted to obtain a degree or diploma at another educational institution;
- and that I commit and undertake to follow the "Kadir Has University Academic Codes and Conduct" prepared in accordance with the "Higher Education Council Codes of Conduct".

In addition, I acknowledge that any claim of irregularity that may arise in relation to this work will result in a disciplinary action in accordance with university legislation.

FARUK CÜREBAL

.....

06.09.2022

To my family

ACKNOWLEDGEMENT

I would like to thank my my thesis supervisor Prof. Dr. Hasan Dağ for his guidance throughout this project. I also would like to thank Prof. Dr. Mustafa Bağrıyanık and Assist. Prof. Dr. E. Fatih Yetkin for their invaluable feedbacks.

I am extremely thankful to Dr. Aykut Çayır, Kadir Has University, for all his technical and moral support throughout the project.

I would like to thank my friends Ferhat Demirkıran, İsmail Erbaş, Kağan Ceylan and Muhammet Kaya for their endless support. In my most desperate moments, I always found their support by my side.

I also express my deepest gratitude to my parents, Mustafa and Kadriye Cürebal, and my sisters Dilek and Fatma Cürebal for their lifelong supports. I could not have done this without them.

ENSEMBLE OF FEATURE SELECTION MODELS FOR MALWARE DATASETS

ABSTRACT

While the development of technology has made our lives easier, our dependence on it has also increased. Cybercriminals develop various types of malware to exploit this dependence. Thus, malware classification is essential for security researchers and incident response teams to take action against them and accelerate mitigation. In this study, we selected seven feature selection methods considering their popularity, effectiveness, and complexity: LOFO Importance (Leave One Feature Out) , FRUFS (Feature Relevance based Unsupervised Feature Selection), AGRM (A General Framework for Auto-Weighted Feature Selection with Global Redundancy Minimization), MI (Mutual Information), Chi-square test, mRMR (Minimum Redundancy and Maximum Relevance), BoostARoota. We performed all the experiments in this study using XGBoost (Extreme Gradient Boosting), RF (Random Forest), and HGB (Histogram-Based Gradient Boosting) machine learning classifiers and accuracy, F1-score, and AUC-score (Area under the ROC Curve) evaluation metrics. We measured the parameter sensitivities of these feature selection methods having adjustable parameters on two high-dimensional datasets: the Microsoft Malware Prediction dataset and the API Call Sequences dataset. These feature selection methods and parameters are FRUFS (model-c, random-state), BoostARoota (clf, iters), and LOFO (model). Only the 'model' parameter of the LOFO algorithm significantly affects the accuracy and F1-score evaluation metric results among the adjustable parameters. We then compared these seven feature selection algorithms using two high-dimensional malware datasets: the Microsoft Malware Prediction dataset and the API Import dataset. Overall results show that AGRM obtained better metric results than other feature selection methods. Behind AGRM, FRUFS, LOFO, MI, and mRMR achieved the best results in different metrics. Compared to MI and mRMR, LOFO is much less used in the malware domain, while FRUFS

has not been used before. Since AGRM performs better and FRUFS and LOFO are newer than other algorithms, we decided to continue our work with these three feature selection methods. Finally, we combined three selected feature selection methods, LOFO Importance, FRUFS, and AGRM, to find the most important features and work with fewer features by reducing the multidimensionality. We trained three feature subsets from these feature selection methods with three models, XGBoost, RF, and HGB classifiers, using a stacking ensemble on the Microsoft Malware Prediction dataset and the API Import dataset. From the nine prediction probabilities we obtained, we eliminated the prediction probabilities containing the same information by setting a threshold in the correlation matrix. We gave the final prediction probabilities we obtained to the SVM (Support Vector Machine) meta classifier. Our model obtained an average of 1.2% better classification accuracy than the selected three feature selection methods on one of the well know malware datasets (Microsoft Malware Prediction dataset). For the API Import dataset, our model obtained an average 8% better classification accuracy than LOFO and FRUFS feature selection algorithms, and AGRM could not be used in that comparison due to insufficient RAM. Therefore, our proposed model was trained with fewer features and got better results.

Keywords: Feature Selection, Ensemble, FRUFS, AGRM, LOFO, Malware Classification

KÖTÜCÜL YAZILIM VERİ KÜMELERİ İÇİN ÖZNİTELİK SEÇİM MODELLERİNİN TOPLULUĞU

ÖZET

Teknolojinin gelişmesi hayatımızı kolaylaştırırken, ona olan bağımlılığımız da arttı. Siber suçlular, bu bağımlılıktan yararlanmak için çeşitli kötü amaçlı yazılım türleri geliştirmektedirler. Bu nedenle, güvenlik araştırmacıları ve olay müdahale ekiplerinin bunlara karşı önlem alması ve sistemlere verilebilecek zararları en aza indirmesi için kötü amaçlı yazılım sınıflandırması çok önemlidir. Bu çalışmada, popülerlik, etkinlik ve karmaşıklıklarını göz önünde bulundurarak yedi öznitelik seçim yöntemi seçtik: LOFO Importance (Leave One Feature Out), FRUFS (Feature Relevance based Unsupervised Feature Selection), AGRM (A General Framework for Auto-Weighted Feature Selection with Global Redundancy Minimization), MI (Mutual Information), Ki-kare testi (Chi-square test), mRMR (Minimum Redundancy and Maximum Relevance) ve BoostARoota. Bu çalışmadaki tüm deneyleri XGBoost (Extreme Gradient Boosting), RF (Random Forest) ve HGB (Histogram-Based Gradient Boosting) makine öğrenimi sınıflandırıcıları ve doğruluk, F1-skor ve AUC-skor (ROC eğrisinin altında kalan alan) değerlendirme ölçütlerini kullanarak gerçekleştirdik. Bu öznitelik seçim yöntemlerinden ayarlanabilir parametre sahibi olanlarının parametre duyarlılıklarını, iki yüksek boyutlu veri kümesinde ölçtük: Microsoft Kötü Amaçlı Yazılım Tahmini veri kümesi ve API Çağrı Dizileri veri kümesi. Bu öznitellik seçim yöntemleri ve parametreleri: FRUFS (model-c, random-state), BoostARoota (clf, iters) ve LOFO'dur (model). LOFO algoritmasının sadece 'model' parametresi, ayarlanabilir parametreler arasında doğruluğu ve F1-skor değerlendirme metrik sonuçlarını önemli ölçüde etkiler. Daha sonra iki yüksek boyutlu kötü amaçlı yazılım veri kümesi kullanarak bu yedi öznitelik seçim algoritmasını karşılaştırdık: Microsoft Kötü Amaçlı Yazılım Tahmini veri kümesi ve API Import veri kümesi. Genel sonuçlar, AGRM'nin diğer öznitelik seçim yöntemlerinden daha iyi metrik sonuçlar elde ettiğini göstermektedir. AGRM'nin arkasında,

FRUFS, LOFO, MI ve mRMR, farklı metriklerde en iyi sonuçları elde etti. MI ve mRMR ile karşılaştırıldığında, LOFO kötü amaçlı yazılım alanında çok daha az kullanılırken, FRUFS daha önce hiç kullanılmadı. AGRM'nin daha iyi performans göstermesi, FRUFS ve LOFO'nun diğer algoritmalardan daha yeni olması nedeniyle çalışmalarımızı bu üç öznitelik seçim yöntemiyle sürdürmeye karar verdik. Son olarak, en önemli öznitelikleri bulmak ve çok boyutluluğu azaltarak daha az öznitelikle çalışmak için seçilen üç öznitelik seçim yöntemini, LOFO, FRUFS ve AGRM'yi kombine ettik. Microsoft Kötü Amaçlı Yazılım Tahmini veri kümesinde ve API İçe Aktarma veri kümesinde bir yığınlama topluluğu kullanarak, XGBoost, RF ve HGB sınıflandırıcıları olmak üzere üç modelle bu öznitelik seçme yöntemlerinden üç öznitelik alt kümesini eğittik. Elde ettiğimiz dokuz tahmin olasılığından aynı bilgiyi içeren tahmin olasılıklarını korelasyon matrisinde bir eşik belirleyerek elimine ettik. Elde ettiğimiz son tahmin olasılıklarını SVM (Support Vector Machine) meta sınıflandırıcısına verdik. Modelimiz, iyi bilinen kötü amaçlı yazılım veri kümelerinden birinde (Microsoft Kötü Amaçlı Yazılım Tahmini veri kümesi) seçilen üç öznitelik seçim yönteminden ortalama 1,2% daha iyi sınıflandırma doğruluğu elde etti. API Import veri seti için modelimiz, LOFO ve FRUFS öznitelik seçim algoritmalarından ortalama 8% daha iyi sınıflandırma doğruluğu elde etti ve yetersiz RAM nedeniyle AGRM bu karşılaştırmada kullanılamadı. Bu nedenle önerilen modelimiz daha az öznitelikle eğitilmiş ve daha iyi sonuçlar elde edilmiştir.

Anahtar Sözcükler: Öznitelik Seçimi, Topluluk, FRUFS, AGRM, LOFO, Kötücül Yazılım Sınıflandırma

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

AGRM	A General Framework for Auto-Weighted Feature Selection via Global Redundancy Minimization
AUC	Area under the ROC Curve
Chi	Chi-square test
CPU	Central processing unit
FRUFS	Feature Relevance based Unsupervised Feature Selection
GPU	Graphical processing unit
HGB	Histogram-Based Gradient Boosting
LOFO	Leave One Feature Out
MI	Mutual Information
mRMR	Minimum Redundancy and Maximum Relevance
RAM	Random Access Memory
RF	Random Forest
SVM	Support Vector Machine
XGBoost	Extreme Gradient Boosting

1. INTRODUCTION

With the development of information technologies, technology has become more accessible and a massive part of our lives. We do most of our daily tasks with it. This usage of technology has brought some serious vulnerabilities as well as advantages. Cybercriminals (called "hackers") who are experts in information technologies have recently increased usage of their immense knowledge and abilities to exploit those vulnerabilities. Malicious software (malware) is their most significant assistant in this way (Demirkıran et al. 2021).

Malware is any software crafted by cybercriminals to steal sensitive information, damage, or destroy computer systems. Given their powerful and devastating effects, it is hardly surprising that researchers have done much work in this area. Some of these studies are in their correct classification(Kalash et al. 2018).

Ransomware is one of the malware and is used by cybercriminals, often with fake emails with malicious links. According to Pandasecurity, ransomware attacks increased by more than 140% in the 3rd quarter of 2021 alone (Pandasecurity 2022). According to Cost of a Data Breach Report released by IBM in 2021 the total cost of a ransomware breach averaged $4.62 million excluding ransom in 2021(IBM 2021). These findings highlight the importance of malware classification.

The importance of malware classification is obvious, and it is also critical to classify it quickly. Security researchers and incident responders need to be able to react fast when classifying malware in order to take precautions and get immediate solutions. In this study, by reducing the number of features to the minimum possible number, we obtain results close to or even better than those obtained from training the entire dataset. The importance of this work is that our proposed model allows us to work with high accuracy classification results with significantly fewer features.

Our main contributions to this study can be outlined as follows:

- We have built Innovative and multi-stage model for malware classification.
- We have leveraged most used and up-to-date feature selection methods in malware classification.
- We have compared parameters sensitivity of feature selection methods by parameter adjustment.
- We have used a correlation matrix to eliminate ensemble models carrying close prediction probabilities.
- To the best of our knowledge, we have used the FRUFS feature selection method for the first time in the field of malware in this study.

This paper is structured as follows: Section 2 presents the related work. Section 3 presents the description of the datasets and our proposed model. In Section 4, experiments and results are presented, and lastly, the conclusion and future work are given in Section 5.

2. RELATED WORK

2.0.1 Feature Selection

When the number of data increases seriously, its quality decreases gradually and that number of data may include irrelevant features that researchers want to eliminate (Venkatesh and Anuradha 2019). Feature selection is an important method for dealing with irrelevant and redundant features (Li et al. 2017; Alelyani et al. 2018). Researchers have often used feature selection in the literature to reduce multidimensionality and eliminate redundant features in the field of malware (Komashinskiy and Kotenko 2010; Moonsamy et al. 2011; Fang et al. 2019; Lin et al. 2015).

In this study, we examined seven different feature selection methods.

AGRM stands for A General Framework for Auto-Weighted Feature Selection via Global Redundancy Minimization is a novel, auto-weighted feature selection method to find minimum global redundancy (Nie et al. 2018). We coded the algorithm proposed in the article by using the Python Cvxpy library. Cvxpy is an open-source Python library for convex optimization problems (Diamond and Boyd 2016; Agrawal et al. 2018).

LOFO (Leave One Feature Out) Importance is one feature selection method that calculates the importance of a given list of features by removing each feature from the list one by one and evaluating the model's performance according to the chosen metric mentioned in a Github link.[1] The researchers have become successful in the Microsoft Malware Prediction data science competition in Kaggle by using LOFO Importance on Microsoft Malware Prediction 2019 dataset (Çayır et al. 2019).

[1] `https://github.com/aerdem4/lofo-importance`

In the other study, using the LOFO Importance method on the DREBIN (Arp et al. 2014) dataset, the researchers obtained higher classification results than the whole dataset with fewer features (Roseline and Geetha 2021). Therefore we leveraged LOFO Importance algorithm in our tests to improve our results .

FRUFS (Feature Relevance based Unsupervised Feature Selection) is an unsupervised feature selection method that uses supervised models to evaluate feature importance mentioned in a Github link [2] . To the best of our knowledge, we have used the FRUFS feature selection algorithm for the first time in the field of malware in this study.

MI (Mutual Information) measures the amount of information that can be obtained from and the reduction in uncertainty of the other variable when the value of one variable is known (Witten and Frank 2002; MacKay et al. 2003). Researchers used mutual information to select the best features for malware's dynamic analysis and classification (Moshiri et al. 2017). In the other study, researchers used mutual information to select prominent features to build an Android malware detection system (John and Vinod 2018).

The Chi-square test is one of the feature selection methods used in many studies to select the best features. The researchers used this method to detect android malware (Dhalaria and Gandotra 2020; Lu et al. 2013).

mRMR (Minimum Redundancy and Maximum Relevance) is a feature selection method to find the minimum related features for a given dataset first introduced by (Ding and Peng 2005) since then used in many machine learning applications (Vinod et al. 2013, 2012).

BoostARoota is an XGBoost-based feature selection algorithm mentioned in a Github link [3] .

[2] `https://github.com/atif-hassan/FRUFS`
[3] `https://github.com/chasedehan/BoostARoota`

2.0.2 Stacking Ensemble

In this study, we used a stacking ensemble model (Wolpert 1992; Whalen and Pandey 2013; Ma et al. 2018; Zhou 2012) to combine our prediction probabilities in the best way. Many studies in the literature use the stacking ensemble in malware classification.

Researchers used the stacking ensemble method to classify malware families and improved their results (Yan et al. 2018). The other study by (Feng et al. 2018) also used stacking ensemble for malware detection and family classification for android dataset and achieved good results. Moreover, many other researchers used ensemble models (Wang et al. 2018; Gupta and Rani 2020; Idrees et al. 2017) to choose the best models and improve their results.

2.0.3 Correlation Matrix

The correlation matrix is very helpful in understanding the correlations between features. We used a correlation matrix in this study to see the correlations of prediction probabilities from the models in the ensemble and eliminate redundant ones. There are studies in the literature where the correlation matrix is used in this way. Researchers (Guizani and Ghafoor 2020) used the correlation matrix to see the highly correlated features and eliminated those features to obtain better results for detecting malware in IoT-based networks. In the other study, researchers (Moussas and Andreatos 2021) used the correlation matrix in the same way for the image dataset to detect and classify malware.

3. METHODOLOGY

We compared the seven feature selection algorithms mentioned in the related work section using RF, HGB, and Xgboost models according to accuracy, F1, and AUC-score metrics. We decided to continue with LOFO, FRUFS, and AGRM feature selection algorithms according to the results shown in 4.9,4.10,4.11.

3.0.1 Datasets

Dataset selection is important in seeing how effective our model is when classifying malware. This study aimed to establish an innovative architecture using feature selection methods, an ensemble model, and a correlation matrix. We also created a data preprocessing pipeline to apply to datasets in this study. It is important to use more than one dataset to see the effect of data preprocessing. For this reason, two different malware datasets were used in this study. In addition, it is important to use multidimensional malware datasets to see the effect of feature selection methods on malware datasets. For this reason, these two different malware datasets were selected from multidimensional ones.

Microsoft Malware Prediction Dataset . Microsoft held two significant malware prediction competitions on Kaggle in 2015 and 2019(Çayır et al. 2019). Microsoft's endpoint protection solution, Windows Defender, collected the Malware Prediction 2019 dataset used in this study[4] . In total, the dataset has 84 features and approximately 8.9 million samples. The target variable that means malware detected is HasDetections and MachineIdentifier is the unique identifier of each machine.

[4]`https://www.kaggle.com/competitions/microsoft-malware-prediction/data`

1.55M API IMPORT DATASET . This dataset consists of 1.55 million samples of 1000 API import features extracted from JSON format of the EMBER dataset 2017 v2 and 2018, of which 800,000 are malware and 750,000 benign (Trinh 2021).

API Call Sequences Dataset. The dataset consists of 42,797 malware API call sequences and 1,079 goodware API call sequences extracted from the 'calls' elements of Cuckoo Sandbox reports (Oliveira 2019). The dataset has 102 features, and 'malware' is the target variable.

3.0.2 Our Proposed Model

This study proposes a multi-stage, creative and innovative solution to malware classification. This model is an architecture where many methods that we think are effective in malware classification work in harmony. The first of these methods, feature selection, eliminates redundant features that are ineffective in classification. In addition, to increase the efficiency of the study, three different and up-to-date feature selection methods were chosen and aimed to be used for better classification results. The number of features that each feature selection method considers most important for the classification is different from each other. Each feature selection method gives a graph listing the effects of features on classification. We selected different amounts of features from each feature selection method, taking advantage of their obvious variation in graphs.

In the ensemble model design section, three different machine learning algorithms were trained with three different subsets from three different feature selection methods. As an ensemble method, the stacking ensemble method was chosen to obtain a more powerful model for synthesizing these models. For this reason, powerful and complex models XGBoost classifier, Random Forest Classifier, and Histogram Based Gradient Boost classifier were used. Eventually, we trained the final prediction probabilities we obtained with SVM (Support Vector Machine) due to its relatively simple structure as a meta classifier.

In our correlation matrix design part, we select the best models by passing the prediction probabilities obtained from different models in the stacking ensemble stage through the correlation matrix. Thanks to the correlation matrix, we also look at the correlations of the prediction probabilities from each model and eliminate the models that give very close prediction probabilities to each other. Because these models carry the same information, if we give each of them, we can cause the final model to make a biased choice. In this way, even if subsets from different feature selection models come close to each other, they cannot pass through the correlation matrix. The Figure 3.1 shows our proposed model.

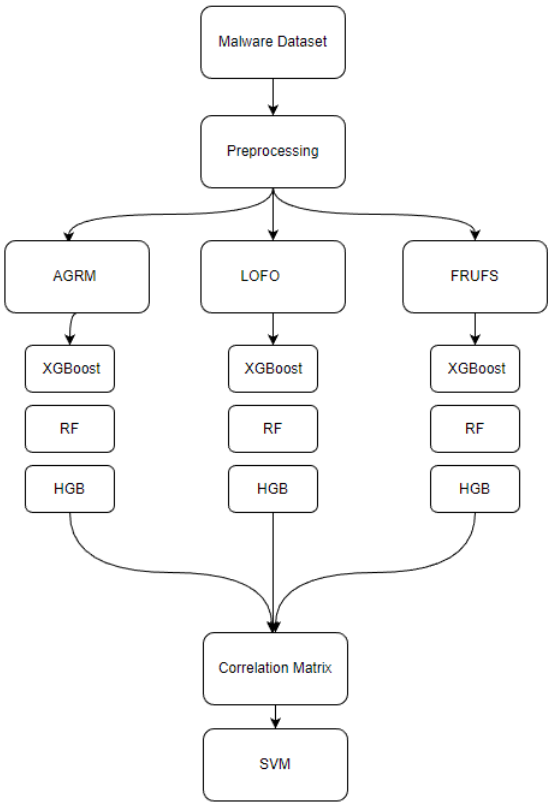

Figure 3.1 Our proposed model

4. Experiments and Results

4.0.1 Experimental Setup

We have conducted initial experiments in the Google Colab platform for the small chunk size of the train set. The computer we used in that platform had a Tesla K80 GPU with 12 GB RAM. We increased the chunk amount to ensure our results were similar for larger samples. For this reason, we continued our tests on the computer with 32 GB RAM and 2 GPU (GeForce GTX 1080 Ti) with 12 GB RAM per GPU.

4.0.2 Preprocessing

Data can sometimes be missing, noisy, and inconsistent. Data preprocessing is the process of optimizing data to overcome these difficulties.

In this study, we created a data preprocessing pipeline, and we applied this pipeline to both datasets. In this pipeline, we used the CategoricalImputer to replace the missing categorical values with the most frequent category and the MeanMedianImputer to replace the missing numeric values with the median value of the variable.

Moreover, we used RareLabelEncoder to group categories with less than 5% observations and the MeanEncoder to replace the categories with the mean value of the target variable for each category. Finally, we used StandardScaler to standardize features by removing the mean and scaling to unit variance.

We applied the same data preprocessing process to all datasets in a pipeline to avoid bias in the results.

4.0.3 Parameter Sensitivity

Feature selection is a nice way to reduce high dimensionality, and many feature selection algorithms are available today. What distinguishes these feature selection algorithms from each other is whether it has a built-in learner or what parameters it has, and how it handles the problem. In order to use them most efficiently, the optimum values of these parameters should be found.

FRUFS. Some parameters of the FRUFS feature selection algorithm are as follows:

model-c : The estimator model to be used in classification,
default=DecisionTreeClassifier().

We compared the results by assigning four machine learning algorithms to the model-c parameter: Random Forest, Hist Gradient Boosting, Xgboost, and Decision Trees. However, we observed that the parameter gives the same feature subsets for four different algorithms. Therefore, the parameter did not affect the results.

random-state : Parameter for reproducible output,
default=None.

We randomly selected three values, 0, 11, and 25, for the random state parameter and observed how the accuracy, f1-score, and AUC-score metric results would change.

Table 4.1 shows that changing the random state parameter values for the Microsoft dataset changes the feature subset chosen by the Frufs feature selection algorithm, and the results differ.

Table 4.1 Random State value change effect for Microsoft dataset.

Model	Accuracy	F1-score	AUC score
random-state = 0			
Random Forest	56.85%	0.607549	0.595030
Hist Gradient Boosting	57.98%	0.596011	0.613642
Xgboost	58.30%	0.605487	0.620324
random-state = 11			
Random Forest	57.30%	0.620613	0.591833
Hist Gradient Boosting	58.90%	0.602130	0.616203
Xgboost	58.67%	0.610601	0.618399
random-state = 25			
Random Forest	57.30%	0.618580	0.595313
Hist Gradient Boosting	57.83%	0.596894	0.614468
Xgboost	58.48%	0.610185	0.620436

Table 4.2 shows how changing the random state parameter value affects the classification results of machine learning models on the API call sequences dataset. We observed that the random state parameter value has little effect on the chosen feature subset and classification results.

Table 4.2 Random State value change effect for API call sequences dataset (Oliveira 2019).

Model	Accuracy	F1-score	AUC score
random-state = 0			
Random Forest	97.58%	0.987726	0.884893
Hist Gradient Boosting	98.45%	0.992108	0.960628
Xgboost	98.35%	0.991605	0.948662
random-state = 11			
Random Forest	97.58%	0.987726	0.883162
Hist Gradient Boosting	98.42%	0.991984	0.968259
Xgboost	98.42%	0.991984	0.946463
random-state = 25			
Random Forest	97.58%	0.987726	0.882333
Hist Gradient Boosting	98.47%	0.992232	0.968734
Xgboost	98.35%	0.991605	0.948662

BoostARoota. Some parameters of the BoostARoota feature selection algorithm are as follows:

clf : The estimator model to be used in feature selection process,
default = XGBClassifier().

For the clf parameter, we determined Decision Trees, Random Forest, and the default value Xgboost machine learning algorithms and compared the results. We observed that Hist Gradient Boosting could not be assigned directly as a classifier to the BoostARoota feature selection algorithm, so we used Decision Trees instead.

The table 4.3 shows the effect of the change in the classifier parameter on the feature selection and classification results in the Microsoft dataset.

The results showed that better accuracy and AUC score were obtained when the estimator was Xgboost, and a better f1-score was obtained when the estimator was Random Forest.

Table 4.3 Classifier change effect for Microsoft dataset .

Model	Accuracy	F1-score	AUC score
clf = Decision Tree			
Random Forest	61.12%	0.615194	0.655031
Hist Gradient Boosting	60.77%	0.592150	0.663720
Xgboost	61.58%	0.606604	0.669471
clf = Random Forest			
Random Forest	61.22%	**0.625091**	0.655708
Hist Gradient Boosting	60.72%	0.592054	0.658661
Xgboost	60.90%	0.600204	0.664985
clf = Xgboost			
Random Forest	60.85%	0.611607	0.655714
Hist Gradient Boosting	61.18%	0.592067	**0.670607**
Xgboost	**61.32%**	0.604853	0.669686

The table 4.4 shows the effect of the change in the classifier parameter on the feature selection and classification results in the API call sequences dataset (Oliveira 2019).

According to the results, the best results in all three metrics were obtained when the estimator was Xgboost, and the classifier was Hist Gradient Boosting.

Table 4.4 Classifier change effect for API call sequences dataset (Oliveira 2019).

Model	Accuracy	F1-score	AUC score
clf = Decision Tree			
Random Forest	97.95%	0.989604	0.909643
Hist Gradient Boosting	98.58%	0.992747	0.970097
Xgboost	98.55%	0.992619	0.958562
clf = Random Forest			
Random Forest	97.95%	0.989604	0.900956
Hist Gradient Boosting	98.47%	0.992236	0.971921
Xgboost	98.47%	0.992242	0.962106
clf = Xgboost			
Random Forest	97.95%	0.989604	0.905959
Hist Gradient Boosting	**98.75%**	**0.993631**	**0.973207**
Xgboost	98.52%	0.992495	0.962211

iters: The number of iterations to find best features, [default=10] - int (iters $>$ 0). The iters parameter of the BoostARoota feature selection algorithm allows us to specify how many iterations the algorithm makes to find the best features. We compared the results by giving the iter parameter values of 5, 10, and 15, respectively. The table 4.5 shows the effect of the iteration change on the classification results on the Microsoft dataset. According to the results, the classifiers got the best results at the value of iter 5.

Table 4.5 Iteration change effect for Microsoft dataset .

Model	Accuracy	F1-score	AUC score
iters = 5			
Random Forest	61.48%	**0.635094**	0.658002
Hist Gradient Boosting	61.82%	0.607758	0.672340
Xgboost	**62.18%**	0.616477	**0.675118**
iters = 10			
Random Forest	60.70%	0.612426	0.655413
Hist Gradient Boosting	61.10%	0.592670	0.664373
Xgboost	61.85%	0.608517	0.670882
iters = 15			
Random Forest	60.85%	0.617302	0.656996
Hist Gradient Boosting	61.42%	0.596601	0.662661
Xgboost	**62.18%**	0.610553	0.672213

The table 4.6 shows the effect of the iteration change on the classification results on the API call sequences dataset (Oliveira 2019). According to the results, the best results in all three metrics were obtained with the Hist Gradient Boosting model when the iters value was 5.

Table 4.6 Iteration change effect for API call sequences dataset (Oliveira 2019).

Model	Accuracy	F1-score	AUC score
iters = 5			
Random Forest	97.58%	0.987726	0.890639
Hist Gradient Boosting	**98.12%**	**0.990467**	**0.956042**
Xgboost	97.97%	0.989720	0.933760
iters = 10			
Random Forest	97.58%	0.987726	0.889130
Hist Gradient Boosting	98.05%	0.990094	0.942470
Xgboost	97.88%	0.989215	0.927225
iters = 15			
Random Forest	97.58%	0.987726	0.886947
Hist Gradient Boosting	97.92%	0.989452	0.950948
Xgboost	97.90%	0.989340	0.925599

LOFO. The LOFO feature selection algorithm has only one parameter called model, which can affect the results if adjusted.

model = The estimator model, default LightGBM().

We assigned the model parameter of the LOFO feature selection algorithm as the default parameter, which is LightGBM, Random Forest, Hist Gradient Boosting, and Xgboost, and compared the classification results. The table 4.7 shows the effect of changing the LOFO feature selection algorithm's model parameter on the Microsoft dataset classification results. According to the results, the best accuracy and AUC-score results were obtained with the LightGMB estimator model and Xgboost classifier. The best F1-score was obtained with the Xgboost estimator model and Random Forest classifier.

Table 4.7 Estimator model change effect for Microsoft dataset .

Model	Accuracy	F1-score	AUC score
model = LightGBM			
Random Forest	59.00%	0.661157	0.630664
Hist Gradient Boosting	59.40%	0.602740	0.633044
Xgboost	**60.10%**	0.630213	**0.638619**
model = Random Forest			
Random Forest	58.20%	0.587771	0.627283
Hist Gradient Boosting	54.90%	0.538383	0.585691
Xgboost	56.60%	0.537313	0.616889
model = Hist Gradient Boosting			
Random Forest	59.40%	0.661667	0.615337
Hist Gradient Boosting	55.80%	0.570039	0.587531
Xgboost	58.90%	0.628054	0.610240
model = Xgboost			
Random Forest	59.70%	**0.665560**	0.618779
Hist Gradient Boosting	58.40%	0.594542	0.602850
Xgboost	60.00%	0.638989	0.633474

The table 4.8 shows the effect of changing the LOFO feature selection algorithm's model parameter on the API call sequences dataset (Oliveira 2019) classification results. According to the results, while there was no change in the accuracy and F1-score metrics, there was a slight change in the Auc score, and the highest result was obtained when the estimator model was Random Forest.

Table 4.8 Estimator model change effect for API call sequences dataset (Oliveira 2019).

Model	Accuracy	F1-score	AUC score
model = LightGBM			
Random Forest	97.70%	0.988366	0.914779
Hist Gradient Boosting	**98.60%**	**0.992886**	0.966446
Xgboost	**98.60%**	0.992886	0.954119
model = Random Forest			
Random Forest	97.70%	0.988366	0.843309
Hist Gradient Boosting	**98.60%**	**0.992886**	**0.969872**
Xgboost	98.50%	0.992374	0.944462
model = Hist Gradient Boosting			
Random Forest	97.70%	0.988366	0.901829
Hist Gradient Boosting	**98.60%**	**0.992886**	0.927240
Xgboost	98.30%	0.991357	0.924124
model = Xgboost			
Random Forest	97.70%	0.988366	0.893952
Hist Gradient Boosting	**98.60%**	**0.992886**	0.957278
Xgboost	98.50%	0.992374	0.954519

4.0.4 Feature Selection Comparison

In that section, we compared the previously selected seven feature selection methods using the RF, HGB, and Xgboost models according to the accuracy, F1, and AUC-score metrics.

Table 4.9 shows the accuracy, F1-score, and AUC-score metric results obtained by training Xgboost model with the feature subsets from seven feature selection algorithms.According to the results, FRUFS feature selection algorithm obtained

higher scores than other algorithms in accuracy and AUC-score evaluation metrics.

Table 4.9 Feature selection comparison results with Xgboost model for Microsoft dataset.

Feature Selection	Accuracy	F1-score	AUC score
MRMR	61.65%	**0.609470**	0.669637
MI	61.82%	0.610360	0.669408
Chi	55.65%	0.566683	0.577591
LOFO	61.80%	0.609407	0.672903
FRUFS	**62.00%**	0.609455	**0.674751**
AGRM	61.45%	0.609225	0.673089
BoostARoota	60.45%	0.592058	0.663276

Table 4.10 shows the accuracy, F1-score, and AUC-score metric results obtained by training Random Forest model with the feature subsets from seven feature selection algorithms.According to the results, MI, AGRM, and FRUFS, feature selection methods, obtained higher accuracy, F1, and AUC-score metrics scores, respectively.

Table 4.10 Feature selection comparison results with Random Forest model for Microsoft dataset.

Feature Selection	Accuracy	F1-score	AUC score
MRMR	61.05%	0.627629	0.657565
MI	**61.20%**	0.631704	0.655036
Chi	55.20%	0.562927	0.563948
LOFO	60.98%	0.630882	0.654814
FRUFS	61.00%	0.623370	**0.658435**
AGRM	61.18%	**0.646000**	0.658104
BoostARoota	60.17%	0.599044	0.653551

Table 4.11 shows the accuracy, F1-score, and AUC-score metric results obtained by training Xgboost model with the feature subsets from seven feature selection algorithms.According to the results, LOFO, AGRM, and FRUFS, feature selection methods, obtained higher accuracy, F1, and AUC-score metrics scores, respectively.

Table 4.11 Feature selection comparison results with Hist Gradient Boosting model for Microsoft dataset.

Feature Selection	Accuracy	F1-score	AUC score
MRMR	60.95%	0.604757	0.663874
MI	61.25%	0.594241	0.663521
Chi	56.17%	0.586654	0.572309
LOFO	**62.20%**	0.606660	0.673703
FRUFS	61.72%	0.604495	**0.668783**
AGRM	61.50%	**0.614035**	0.668321
BoostARoota	60.65%	0.587742	0.660465

Table 4.12 shows the accuracy, F1-score, and AUC-score metric results obtained by training Xgboost model with the feature subsets from seven feature selection algorithms. According to the results, mRMR feature selection algorithm obtained higher scores than other algorithms in accuracy and F1-score evaluation metrics and AGRM feature selection algorithm obtained highest AUC-score.

Table 4.12 Feature selection comparison results with Xgboost model for API call sequences dataset (Oliveira 2019).

Feature Selection	Accuracy	F1-score	AUC score
MRMR	**98.67%**	**0.993253**	0.936882
MI	98.42%	0.991986	0.937804
Chi	98.15%	0.990604	0.930925
LOFO	98.30%	0.991353	0.931381
FRUFS	98.35%	0.991603	0.943601
AGRM	98.38%	0.991731	**0.949674**
BoostARoota	98.08%	0.990220	0.919080

Table 4.13 shows the accuracy, F1-score, and AUC-score metric results obtained by training Random Forest model with the feature subsets from seven feature selection algorithms. According to the results, AGRM feature selection algorithm obtained higher scores than other algorithms in accuracy and F1-score evaluation metrics and MI feature selection algorithm obtained highest AUC-score.

Table 4.13 Feature selection comparison results with Random Forest model for API call sequences dataset (Oliveira 2019).

Feature Selection	Accuracy	F1-score	AUC score
MRMR	97.95%	0.989604	0.879658
MI	97.95%	0.989604	**0.898228**
Chi	97.58%	0.987726	0.813777
LOFO	97.58%	0.987726	0.868362
FRUFS	97.58%	0.987726	0.886301
AGRM	**98.10%**	**0.990358**	0.866954
BoostARoota	97.58%	0.987726	0.861085

Table 4.14 shows the accuracy, F1-score, and AUC-score metric results obtained by training Hist Gradient Boosting model with the feature subsets from seven feature selection algorithms. According to the results, AGRM feature selection algorithm obtained highest scores than other algorithms in accuracy, F1-score and AUC-score evaluation metrics.

Table 4.14 Feature selection comparison results with Hist Gradient Boosting for API call sequences dataset (Oliveira 2019).

Feature Selection	Accuracy	F1-score	AUC score
MRMR	98.62%	0.993000	0.966180
MI	98.65%	0.993125	0.964656
Chi	98.28%	0.991229	0.955930
LOFO	98.47%	0.992238	0.955439
FRUFS	98.58%	0.992745	0.957516
AGRM	**98.78%**	**0.993756**	**0.967066**
BoostARoota	97.97%	0.989709	0.925956

Overall results show that AGRM, LOFO,FRUFS, mRMR and MI feature selection methods give better scores. When we compare the LOFO, FRUFS, and AGRM feature selection methods with MI and mRMR, we see that they are newer methods and have not been used much, especially in the field of malware. Especially since AGRM is based on convex optimization, it is interesting to see how successful this

new feature selection algorithm will be in this area.

For these reasons, we decided to continue our work with LOFO, FRUFS, and AGRM feature selection algorithms.

4.0.5 Model Comparison

After applying preprocessing to datasets (the Microsoft Malware Prediction dataset and the API Import dataset), we divided them as 75% train and the remaining 25% test set. We applied selected three feature selection methods for each dataset, and we obtained feature subsets.

The Figure 4.1 shows the best features selected by the LOFO feature selection. The features are sorted according to their effects in the classification. The LOFO feature selection algorithm has selected the best 50 features out of 81 for the Microsoft Malware Prediction dataset.

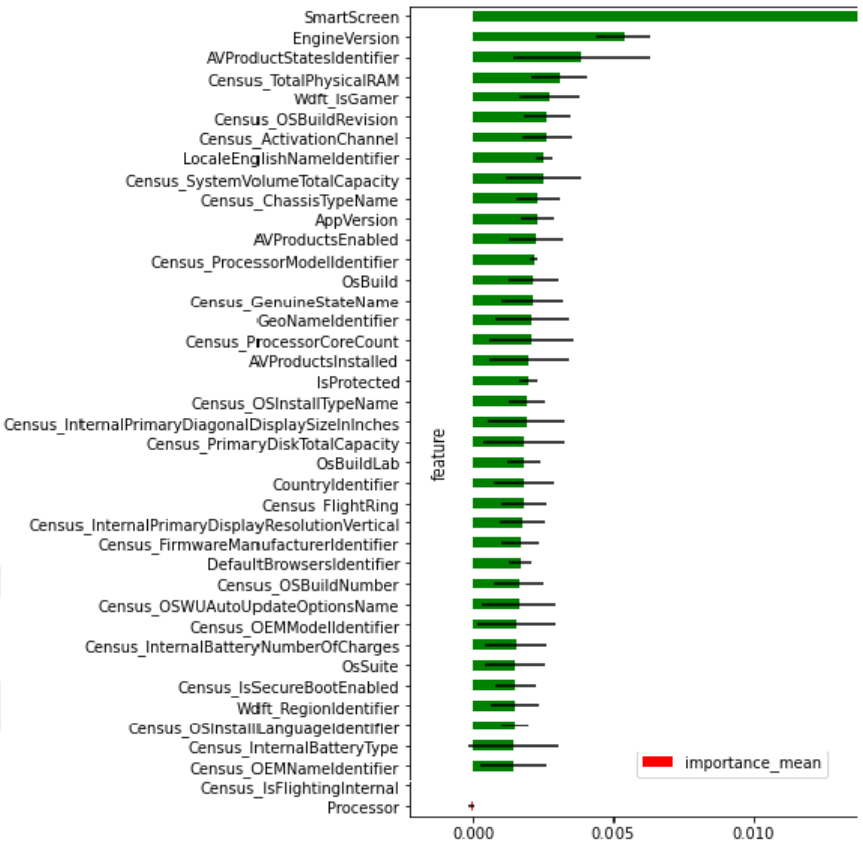

Figure 4.1 Lofo feature selection.

The Figure 4.2 shows the best features selected by the FRUFS feature selection. The features are sorted according to their effects in the classification. FRUFS feature selection algorithm draws a horizontal blue line on the graph, showing the most effective features to choose from. FRUFS has selected the best 67 features out of 81 for the Microsoft Malware Prediction dataset.

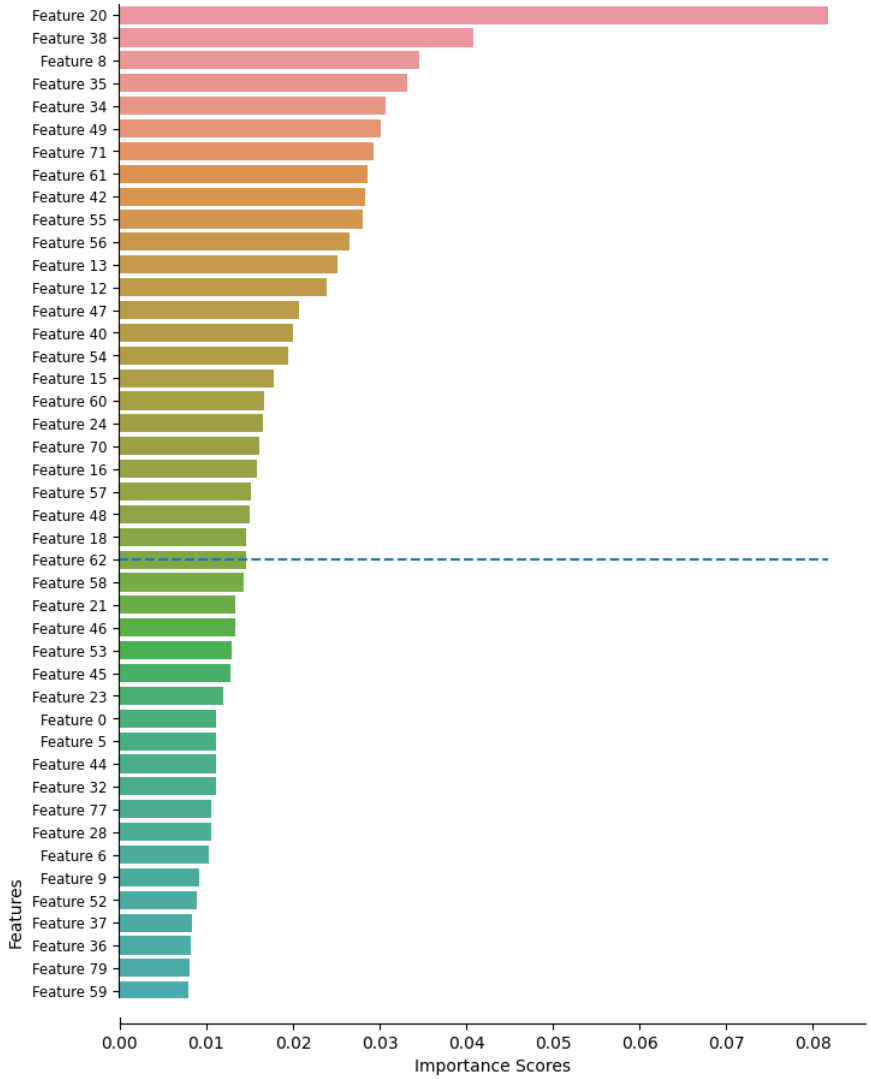

Figure 4.2 FRUFS feature selection.

The Figure 4.3 shows the graph created by the AGRM feature selection algorithm after the feature selection process. The horizontal part of the graph shows the number of features, and the vertical part shows the lambda value used in feature selection. AGRM feature selection algorithm is a method based on convex optimization. The algorithm finds the best features when the lambda value approaches its optimum value. The point on the graph where the lambda value stabilizes gives the number of best features. The algorithm also presents the best features in a list. We have selected the top 10 features by looking at the graph.

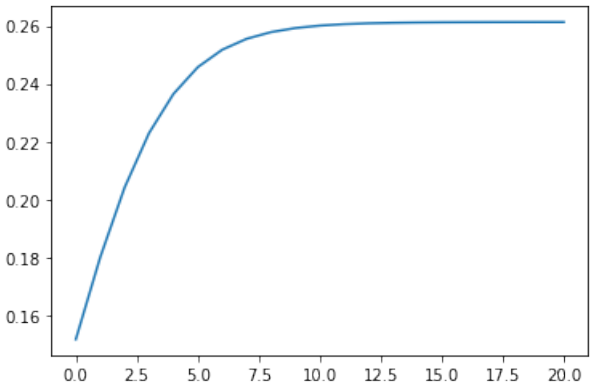

Figure 4.3 AGRM feature selection.

The Figure 4.4 shows the correlations of the predictions made by XGboost, Random Forest, and Hist Gradient Boosting models by training them with feature subsets from the LOFO, FRUFS, and AGRM feature selection algorithms.Predictions with high correlation, especially above 0.9, show that models carry information close to each other. For this reason, we determined the threshold 0.9 and gave the predictions between 0.8 and 0.9, which will make the main difference to the meta classifier.We used SVM (Support Vector Machine) as a meta classifier as a simple model.

	cvx_xgb	lofo_xgb	frufs_xgb	cvx_rf	lofo_rf	frufs_rf	cvx_hgb	lofo_hgb	frufs_hgb
cvx_xgb	1.000000	0.913931	0.914980	0.920579	0.858135	0.841916	0.929006	0.827238	0.820579
lofo_xgb	0.913931	1.000000	0.985758	0.882329	0.892565	0.865676	0.847466	0.911579	0.907621
frufs_xgb	0.914980	0.985758	1.000000	0.880255	0.889010	0.862196	0.847475	0.911387	0.913392
cvx_rf	0.920579	0.882329	0.880255	1.000000	0.919678	0.905040	0.835155	0.787217	0.775236
lofo_rf	0.858135	0.892565	0.889010	0.919678	1.000000	0.980219	0.788095	0.803709	0.794490
frufs_rf	0.841916	0.865676	0.862196	0.905040	0.980219	1.000000	0.774521	0.779463	0.771259
cvx_hgb	0.929006	0.847466	0.847475	0.835155	0.788095	0.774521	1.000000	0.784228	0.779730
lofo_hgb	0.827238	0.911579	0.911387	0.787217	0.803709	0.779463	0.784228	1.000000	0.926613
frufs_hgb	0.820579	0.907621	0.913392	0.775236	0.794490	0.771259	0.779730	0.926613	1.000000

Figure 4.4 Correlation matrix for Microsoft dataset.

The Figure 4.5 shows the correlations of the predictions made by XGboost, Random Forest, and Hist Gradient Boosting models by training them with feature subsets from the LOFO, FRUFS feature selection algorithms. Highly correlated prediction probabilities above 0.9 were eliminated as other correlation matrix.

	lofo_xgb	frufs_xgb	lofo_rf	frufs_rf	lofo_hgb	frufs_hgb
lofo_xgb	1.000000	0.623543	0.789368	0.555467	0.953421	0.654612
frufs_xgb	0.623543	1.000000	0.726159	0.803446	0.610272	0.933124
lofo_rf	0.789368	0.726159	1.000000	0.873178	0.708984	0.661150
frufs_rf	0.555467	0.803446	0.873178	1.000000	0.497925	0.697780
lofo_hgb	0.953421	0.610272	0.708984	0.497925	1.000000	0.679788
frufs_hgb	0.654612	0.933124	0.661150	0.697780	0.679788	1.000000

Figure 4.5 Correlation matrix for API Import (Trinh 2021) dataset.

4.0.6 Results

In this section, we will present the results of our tests on two different data sets. Tables 4.15,4.16,4.17,4.18 represent the test results of the Microsoft Malware prediction dataset.

Table 4.15 shows the accuracy, F1-score, and AUC-score metric results from training the Microsoft malware prediction dataset with three different models and our proposed model without using any feature selection method. According to the results, for Microsoft Prediction dataset, If we evaluate the other three models among themselves, Hist Gradient boosting in accuracy, Random Forest in F1-score, and Xgboost in AUC score got the best results. Our proposed model obtained higher values than other models in all three different evaluation metrics.

Table 4.15 Model comparison results for Microsoft dataset.

Model	Accuracy	F1-score	AUC score
Xgboost	62.00%	0.609455	0.674736
Random Forest	61.38%	0.616149	0.659116
Hist Gradient Boosting	62.15%	0.612986	0.674070
Our Model	**62.83%**	**0.617639**	**0.684048**

Table 4.16 shows the accuracy, F1-score, and AUC-score metric results obtained by training three different models with the feature subset from the LOFO feature selection algorithm. According to the results, our proposed model has higher accuracy and AUC score results than the other three models, while it has a slightly lower result close to them in the F1-score metric.

Table 4.16 Model comparison results with LOFO feature selection for Microsoft dataset.

Model	Accuracy	F1-score	AUC score
Xgboost	61.80%	0.609407	0.672903
Random Forest	60.98%	**0.630882**	0.654814
Hist Gradient Boosting	62.20%	0.606660	0.673703
Our Model	**62.83%**	0.617639	**0.684048**

Table 4.17 shows the accuracy, F1-score, and AUC-score metric results obtained by training three different models with the feature subset from the FRUFS feature selection algorithm. According to the results, our proposed model has higher accuracy and AUC score results than the other three models, while it has a slightly lower result close to them in the F1-score metric.

Table 4.17 Model comparison results with FRUFS feature selection for Microsoft dataset.

Model	Accuracy	F1-score	AUC score
Xgboost	62.00%	0.609455	0.674751
Random Forest	61.00%	**0.623370**	0.658435
Hist Gradient Boosting	61.72%	0.604495	0.668783
Our Model	**62.83%**	0.617639	**0.684048**

Table 4.18 shows the accuracy, F1-score, and AUC-score metric results obtained by training three different models with the feature subset from the AGRM feature selection algorithm. According to the results, our proposed model has higher accuracy and AUC score results than the other three models, while it has a slightly lower result close to them in the F1-score metric.

Table 4.18 Model comparison results with AGRM feature selection for Microsoft dataset.

Model	Accuracy	F1-score	AUC score
Xgboost	61.45%	0.609225	0.673089
Random Forest	61.18%	**0.646000**	0.658104
Hist Gradient Boosting	61.50%	0.614035	0.668321
Our Model	**62.83%**	0.617639	**0.684048**

Tables 4.19,4.20,4.21, represent the test results of the API Import dataset. Since the dataset is very large, we carried out the studies in this dataset on Google Colab Pro plus platform. Although we have 51 GB of RAM on this platform, we could not make the feature selection process more than 50000 chunk-size due to insufficient ram. Also, due to insufficient ram, we had to remove the AGRM feature selection method for this dataset. The results were obtained by training the models with the best 100 features selected from 1000 features of 50000 chunk-size.

Table 4.19 shows the accuracy, F1-score, and AUC-score metric results from training the API Import dataset with three different models and our proposed model without using any feature selection method. According to the results, HGB algorithm did the best on all three metrics. Considering that we have reduced the feature number to one in 10 and cannot increase the chunk size due to insufficient ram, our model results are successful.

Table 4.19 Model comparison results for API Import (Trinh 2021) dataset.

Model	Accuracy	F1-score	AUC score
Xgboost	83.47%	0.854885	0.926443
Random Forest	71.80%	0.807377	0.817660
Hist Gradient Boosting	**86.60%**	**0.886296**	**0.941192**
Our Model	85.60%	0.875163	0.926504

Table 4.20 shows the accuracy, F1-score, and AUC-score metric results obtained by training three different models with the feature subset from the LOFO feature selection algorithm. According to the results, our proposed model obtained higher

values than other models in all three different evaluation metrics.

Table 4.20 Model comparison results with LOFO feature selection for API Import (Trinh 2021) dataset.

Model	Accuracy	F1-score	AUC score
Xgboost	77.48%	0.842510	0.876663
Random Forest	69.27%	0.794310	0.780962
Hist Gradient Boosting	79.05%	0.852465	0.894891
Our Model	**85.60%**	**0.875163**	**0.926504**

Table 4.21 shows the accuracy, F1-score, and AUC-score metric results obtained by training three different models with the feature subset from the FRUFS feature selection algorithm. According to the results, our proposed model obtained higher values than other models in all three different evaluation metrics.

Table 4.21 Model comparison results with FRUFS feature selection for API Import (Trinh 2021) dataset.

Model	Accuracy	F1-score	AUC score
Xgboost	74.45%	0.822877	0.780374
Random Forest	71.15%	0.801990	0.698672
Hist Gradient Boosting	76.72%	0.835367	0.792859
Our Model	**85.60%**	**0.875163**	**0.926504**

4.0.7 Limitations

We encountered some limitations while conducting this study. In this section, we will address these limitations.

Our first limitation is AGRM feature selection algorithm requires a lot of ram due to its structure. Since the API Import dataset contains 1000 features, we carried out our work on the Google Colab pro plus environment. Finally, we decided not to use AGRM feature selection method with this dataset because it stopped working after a certain period.

Another limitation we encountered is that although we were able to obtain results from other feature selection methods for the API Import dataset, we could not reach the 50000 chunk-size amount for AGRM due to the insufficient amount of ram.

5. CONCLUSIONS AND FUTURE WORK

In this study, we selected seven feature selection methods considering their popularity, effectiveness, and complexity. We measured the parameter sensitivities of these feature selection methods having adjustable parameters. We observed that only the 'model' parameter of the LOFO algorithm significantly affects the accuracy and F1-score evaluation metric results among the adjustable parameters. We then compared these seven feature selection algorithms and observed that AGRM obtained better metric results than other feature selection methods and behind AGRM, FRUFS, LOFO, MI, and mRMR achieved the best results in different metrics. Considering AGRM performs better, and FRUFS and LOFO are newer than other algorithms, we decided to continue our work with these three feature selection methods.

Then we proposed a creative and state-of-the-art ensemble model for malware datasets. We evaluated the performance of our model with the accuracy, F1-score and AUC score metrics for two high-dimensional malware datasets. Our model achieved better results in all three metrics, Accuracy, F1-score, and AUC score (Area under the ROC Curve) obtained from models without using feature selection for the Microsoft Malware Prediction dataset. Our model got better accuracy and AUC score metrics than Feature Selection methods.

For the API Import dataset, the results of the models without using feature selection are better than the feature selection methods. However, our model results are higher than the results obtained by the feature selection methods. That shows us the strength of our proposed model. Feature selection methods are ineffective on this dataset. Based on our assumption, the reason for this inefficiency is that the dataset has lots of features and samples, a large amount of ram is required for large chunk sizes, and our experimental setup is insufficient.

Our proposed model's significant success is based on the ensemble's strength, dimensionality reduction power of feature selection, and increasing prediction diversity due to the correlation matrix.

In the future, we are planning to increase RAM size and hence chunk size to obtain more reliable and robust results and deploy our model to an antivirus scanner.

REFERENCES

Agrawal, Akshay, Robin Verschueren, Steven Diamond, and Stephen Boyd. 2018. "A Rewriting System for Convex Optimization Problems." *Journal of Control and Decision* 5 (1): 42–60. https://doi.org/10.1080/23307706.2017.1397554.

Alelyani, Salem, Jiliang Tang, and Huan Liu. 2018. "Feature Selection for Clustering: A Review." In *Data Clustering*, 29–60. Chapman and Hall/CRC.

Arp, Daniel, Michael Spreitzenbarth, Malte Hübner, Hugo Gascon, and Konrad Rieck. 2014. "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket." In *Proceedings 2014 Network and Distributed System Security Symposium*, 23–26. Reston, VA: Internet Society.

Unal, Ugur and Yenidoğan, Işıl and Dag, Hasan and Cayir, Aykut. 2020. "Use Case Study: Data Science Application for Microsoft Malware Prediction Competition on Kaggle." In *International Conference on Data Science, Machine Learning and Statistics.*

Demirkıran, Ferhat, Aykut Çayır, Uğur Ünal, and Hasan Dağ. 2022. "An Ensemble of Pre-Trained Transformer Models for Imbalanced Multiclass Malware Classification." *Computers & Security* 121 (October): 102846. https://doi.org/10.1016/j.cose.2022.102846.

Dhalaria, Meghna, and Ekta Gandotra. 2020. "Android Malware Detection Using Chi-Square Feature Selection and Ensemble Learning Method." In *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 36–41. IEEE.

Diamond, Steven, and Stephen Boyd. 2016. "CVXPY: A Python-Embedded Modeling Language for Convex Optimization." *Journal of Machine Learning Research: JMLR* 17 (1): 2909–13.

Ding, Chris, and Hanchuan Peng. 2005. “Minimum Redundancy Feature Selection from Microarray Gene Expression Data.” *Journal of Bioinformatics and Computational Biology* 03 (02): 185–205. https://doi.org/10.1142/s0219720005001004.

Fang, Zhiyang, Junfeng Wang, Jiaxuan Geng, and Xuan Kan. 2019. “Feature Selection for Malware Detection Based on Reinforcement Learning.” *IEEE Access: Practical Innovations, Open Solutions* 7: 176177–87. https://doi.org/10.1109/access.2019.2957429.

Feng, Pengbin, Jianfeng Ma, Cong Sun, Xinpeng Xu, and Yuwan Ma. 2018. “A Novel Dynamic Android Malware Detection System with Ensemble Learning.” *IEEE Access: Practical Innovations, Open Solutions* 6: 30996–11. https://doi.org/10.1109/access.2018.2844349.

Guizani, Nadra, and Arif Ghafoor. 2020. “A Network Function Virtualization System for Detecting Malware in Large IoT Based Networks.” *IEEE Journal on Selected Areas in Communications* 38 (6): 1218–28. https://doi.org/10.1109/jsac.2020.2986618.

Gupta, Deepak, and Rinkle Rani. 2020. “Improving Malware Detection Using Big Data and Ensemble Learning.” *Computers & Electrical Engineering: An International Journal* 86 (106729): 106729. https://doi.org/10.1016/j.compeleceng.2020.106729.

IBM. 2021. “Cost of a Data Breach Report 2021.”

Idrees, Fauzia, Muttukrishnan Rajarajan, Mauro Conti, Thomas M. Chen, and Yogachandran Rahulamathavan. 2017. “PIndroid: A Novel Android Malware Detection System Using Ensemble Learning Methods.” *Computers & Security* 68: 36–46. https://doi.org/10.1016/j.cose.2017.03.011

John, Meenu Mary, and P. Vinod. 2018. “Statistical Approach Using Meta Features for Android Malware Detection System.” In *Advances in Intelligent Systems and Computing*, 269–79. Singapore: Springer Singapore.

Kalash, Mahmoud, Mrigank Rochan, Noman Mohammed, Neil D. B. Bruce, Yang Wang, and Farkhund Iqbal. 2018. "Malware Classification with Deep Convolutional Neural Networks." In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 1–5. IEEE.

Komashinskiy, Dmitriy, and Igor Kotenko. 2010. "Malware Detection by Data Mining Techniques Based on Positionally Dependent Features." In *2010 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, 617–23. IEEE.

Li, Yun, Tao Li, and Huan Liu. 2017. "Recent Advances in Feature Selection and Its Applications." *Knowledge and Information Systems* 53 (3): 551–77. https://doi.org/10.1007/s10115-017-1059-8.

Lin, Chih-Ta and Wang, Nai-Jian and Xiao, Han and Eckert, Claudia. 2015. "Feature Selection and Extraction for Malware Classification." *Journal of Information Science and Engineering* 31 (3): 965–92.

Lu, Yu, Pan Zulie, Liu Jingju, and Shen Yi. 2013. "Android Malware Detection Technology Based on Improved Bayesian Classification." In *2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control*, 1338–41. IEEE.

Ma, Zhiyuan, Ping Wang, Zehui Gao, Ruobing Wang, and Koroush Khalighi. 2018. "Ensemble of Machine Learning Algorithms Using the Stacked Generalization Approach to Estimate the Warfarin Dose." *PloS One* 13 (10): e0205872. https://doi.org/10.1371/journal.pone.0205872.

MacKay, David J. C. 2003. *Information Theory, Inference and Learning Algorithms*. Cambridge, England: Cambridge University Press.

Moonsamy, Veelasha and Tian, Ronghua and Batten, Lynn. 2011. "Feature Reduction to Speed up Malware Classification." In *Nordic Conference on Secure IT Systems*, 176–88. Springer.

Moshiri, Ehsan, Azizol Bin Abdullah, Raja Azlina Binti Binti, Zaiton Muda. 2017. “Malware Classification Framework for Dynamic Analysis Using Information Theory.” *Indian Journal of Science and Technology* 10 (21): 1–10. https://doi.org/10.17485/ijst/2017/v10i21/100023.

Moussas, Vassilios, and Antonios Andreatos. 2021. “Malware Detection Based on Code Visualization and Two-Level Classification.” *Information (Basel)* 12 (3): 118. https://doi.org/10.3390/info12030118.

Nie, Feiping, Sheng Yang, Rui Zhang, and Xuelong Li. 2018. “A General Framework for Auto-Weighted Feature Selection via Global Redundancy Minimization.” *IEEE Transactions on Image Processing: A Publication of the IEEE Signal Processing Society* 28 (5): 2428–38. https://doi.org/10.1109/TIP.2018.2886761.

Oliveira, Angelo. 2019. “Malware Analysis Datasets: API Call Sequences.” IEEE Dataport. https://doi.org/10.21227/tqqm-aq14.

Pandasecurity. 2022. “73 Ransomware Statistics Vital for Security in 2022.” https://www.pandasecurity.com/en/mediacenter/security/ransomware-statistics/.

Roseline, S. Abijah, and S. Geetha. 2021. “Android Malware Detection and Classification Using LOFO Feature Selection and Tree-Based Models.” *Journal of Physics. Conference Series* 1911 (1): 012031. https://doi.org/10.1088/1742-6596/1911/1/012031.

Trinh, Quynh. 2021. “1.55m Api Import Dataset for Malware Analysis.” IEEE Dataport. https://doi.org/10.21227/98jc-y909.

Venkatesh, B., and J. Anuradha. 2019. “A Review of Feature Selection and Its Methods.” *Cybernetics and Information Technologies* 19 (1): 3–26. https://doi.org/10.2478/cait-2019-0001

P., Vinod, V. Laxmi, and M. S. Gaur. 2012. “REFORM: Relevant Features for Malware Analysis.” In *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, 738–44. IEEE.

Vinod, P., V. Laxmi, M. S. Gaur, S. Naval, and P. Faruki. 2013. "MCF: MultiComponent Features for Malware Analysis." In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, 1076–81. IEEE.

Wang, Wei, Yuanyuan Li, Xing Wang, Jiqiang Liu, and Xiangliang Zhang. 2018. "Detecting Android Malicious Apps and Categorizing Benign Apps with Ensemble of Classifiers." *Future Generations Computer Systems: FGCS* 78: 987–94. https://doi.org/10.1016/j.future.2017.01.019.

Whalen, Sean, and Gaurav Pandey. 2013. "A Comparative Analysis of Ensemble Classifiers: Case Studies in Genomics." In *2013 IEEE 13th International Conference on Data Mining*, 807–16. IEEE.

Witten, Ian H., and Eibe Frank. 2002. "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations." *SIGMOD Record* 31 (1): 76–77. https://doi.org/10.1145/507338.507355.

Wolpert, David H. 1992. "Stacked Generalization." *Neural Networks: The Official Journal of the International Neural Network Society* 5 (2): 241–59. https://doi.org/10.1016/s0893-6080(05)80023-1.

Yan, Jinpei, Yong Qi, and Qifan Rao. 2018. "Detecting Malware with an Ensemble Method Based on Deep Neural Network." *Security and Communication Networks* 2018: 1–16. https://doi.org/10.1155/2018/7247095.

Zhou, Zhi-Hua. 2012. *Ensemble Methods: Foundations and Algorithms*. Caithness, UK: Whittles Publishing.

CURRICULUM VITAE

Personal Information

Name Surname	: FARUK CÜREBAL

Education

Undergraduate Education	: Electrical and Electronics Engineering, Anadolu University Istanbul, Turkey
Foreign Language Skills	: English

Work Experience

Companies and Dates	: KPMG Turkey October 2020 – Present