ANALYSIS OF MACHINE LEARNING AND DEEP LEARNING TECHNIQUES FOR RANSOMWARE DETECTION

2023
MASTER THESIS
COMPUTER ENGINEERING

Elaf Talib Abduljabbar ABDULJABBAR

Thesis Advisor
Assist. Prof. Dr. İsa AVCI

ANALYSIS OF MACHINE LEARNING AND DEEP LEARNING TECHNIQUES FOR RANSOMWARE DETECTION

Elaf Talib Abduljabbar ABDULJABBAR

Thesis Advisor
Assist. Prof. Dr. İsa AVCI

T.C.
Karabuk University
Institute of Graduate Programs
Department of Computer Engineering
Prepared as
Master Thesis

KARABUK
October 2023

I certify that in my opinion the thesis submitted by Elaf Talib Abduljabbar ABDULJABBAR titled “FEATURE SELECTION-BASED RANSOMWARE DETECTION USING MACHINE LEARNING AND DEEP LEARNING TECHNIQUES ” is fully adequate in scope and quality as a thesis for the degree of Master of Science.

Assist. Prof. Dr. İsa AVCI .....

Thesis Advisor, Department of Computer Engineering

This thesis is accepted by the examining committee with a unanimous vote in the Department of Computer Engineering as a Master of Science thesis October 16, 2023.

Examining Committee Members (Institutions)	Signature
Chairman : Assist.Prof.Dr. Nehad T.A. RAMAHA (KBU)	.....
Member : Assist. Prof. Dr. İsa AVCI (KBU)	.....
Member : Assoc. Prof. Dr. Zafer ALBAYRAK (SUBU)	.....

The degree of Master of Science by the thesis submitted is approved by the Administrative Board of the Institute of Graduate Programs, Karabuk University.

Assoc. Prof. Dr. Zeynep ÖZCAN .....

Director of the Institute of Graduate Programs

"I declare that all the information within this thesis has been gathered and presented in accordance with academic regulations and ethical principles and I have according to the requirements of these regulations and principles cited all those which do not originate in this work as well."

Elaf Talib Abduljabbar ABDULJABBAR

ABSTRACT

M. Sc. Thesis

ANALYSIS OF MACHINE LEARNING AND DEEP LEARNING TECHNIQUES FOR RANSOMWARE DETECTION

Elaf Talib Abduljabbar ABDULJABBAR

Karabuk University
Institute of Graduate Programs
The Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. İsa AVCI
October 2023, 72 pages

Ransomware is a special type of malware by which the attacker targets the victim's device using a link attached by an email, and once the victim opens the attachment, all his files are encrypted. The victim cannot retrieve his data without paying the attacker for the decryption key. Ransomware becomes very dangerous and affects all human facilities, including medical centers, military organizations, security platforms, financial facilities, etc. Ransomware detection and classification-based artificial intelligence applications are essential to limit the attacker's ability and prevent it from harming devices. The current study proposes a new ransomware detection and classification study. Besides, a novel feature selection algorithm is proposed to involve the essential information of network tasks and drop the redundant data that can slow the detection process. The study uses a challenging dataset of 392034 records, 84 features, and 11 different types of ransomware. In the first step, the dataset is preprocessed by cleaning it, encoding all textual (categorical) features, and

normalizing them to ensure it fits all machine learning and deep learning models. In the second step, the dataset is split into the train (80%) and test (20%) for the machine learning models. Besides, another validation set is created with a percentage of (20%) of the training set for the deep learning models. The third step is feature selection, in which the redundant features are dropped using a novel hybrid feature selection method depending on both ANOVA and Random Forests to select the best subset of features. In the fourth step, many machine learning and deep learning models are trained using the training set. The experiment part includes applying the fusion of the individual models (for both machine learning and deep learning models) besides the ensemble learning of these individual models. In the evaluation step, the precision, recall, F1-score, and accuracy are used to assess the performance of the individual, the fusion, and the ensemble models. Besides this, three different feature selection scenarios are conducted to seek the best combination of features. The training time of all models is also computed to see the effect of the feature reduction on the computational costs. Results showed that the best models are the XGB, LGBM, and RF models. Besides that, the ML ensemble model achieves a good performance. The feature selection method minimized the training time significantly, especially for the high-computational models like XGB and LGBM, without any remarkable degradation in performance. The best-obtained accuracy is related to the XGB model with 99.87%. The study is also compared with the current state-of-art methodologies. The comparison proves that the current study outperforms all previous ones. Future work can focus on the idea of hyperparameter optimization to improve the performance.

Key Words : Machine Learning, Deep Learning, Ransomware Detection, Multi-Class Classification, Feature Selection, Security.

Science Code : 92432

ÖZET

Yüksek Lisans Tezi

MAKİNE ÖĞRENMESİ VE DERİN ÖĞRENME TEKNİKLERİ KULLANARAK ÖZELLİK SEÇİMİ TABANLI FİDYE YAZILIMI TESPİTİ

Elaf Talib Abduljabbar ABDULJABBAR

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğr. Üyesi İsa AVCI
Ekim 2023, 72 sayfa

Fidye yazılımı, bilgisayar korsanının(dolandırıcı) kurbanın cihazını bir e-posta ekindeki bağlantı aracılığıyla hedef aldığı özel bir o kadar da kötü amaçlı yazılım türüdür. Kurban seçilen şahıs ek dosyayı açtığında, tüm dosyaları şifrelenir. Tespit edilen kişi, verilerini korsana deşifre anahtarı için ödeme yapmadan geri alamaz. Günümüzde Fidye yazılımı son derece tehlikeli hale gelmiştir ve tıbbi merkezler, askeri organizasyonlar, güvenlik platformları,ve finans kuruluşları dahil olmak üzere tüm insan tesislerini etkilemektedir. Dolandırıcının yeteneğini sınırlamak ve cihazlara zarar vermesini engellemek için fidye yazılımının tespiti ve sınıflandırılmasına dayalı yapay zeka uygulamaları hayati önem taşımaktadır Bu çalışma, yeni bir fidye yazılımı tespiti ve sınıflandırma çalışması önermektedir. Ayrıca, ağ görevlerinin temel bilgisini içerecek ve tespit sürecini yavaşlatabilecek gereksiz verileri düşürecek yeni bir özellik seçim algoritması önermektedir. Çalışma, 392034 kayıt, 84 özellik ve 11 farklı fidye yazılım türünden oluşan zorlayıcı bir veri kümesini kullanmaktadır. İlk etapta veri

kümesi temizlenir, tüm metinsel (kategorik) özellikler kodlanır ve tüm makine öğrenimi ve derin öğrenme modellerine uygun olması için normalize edilir.

İkinci adımda ise, veri kümesi makine öğrenimi modelleri için eğitim (%80) ve test (%20) olarak bölünür. Ayrıca, derin öğrenme modelleri için eğitim kümesinin %20'si bir doğrulama kümesi olarak oluşturulur. Üçüncü adım ise özellik seçimidir. Bu adımda, en iyi özellik alt kümesini seçmek için hem ANOVA hem de Rastgele Ormanlara dayanan hibrit bir özellik seçim yöntemi kullanılarak gereksiz özellikler düşürülür. Dördüncü adımda, birçok makine öğrenimi ve derin öğrenme modeli eğitim kümesi kullanılarak eğitilir. Deney kısmında, bireysel modellerin (hem makine öğrenimi hem de derin öğrenme modelleri için) birleşiminin yanı sıra bu bireysel modellerin topluluğunun öğrenilmesini içerir. Değerlendirme aşamasında, bireysel, birleşim ve topluluk modellerinin performansını değerlendirmek için kesinlik, hatırlama, F1 puanı ve doğruluk kullanılır. Bunun yanı sıra en iyi özellik kombinasyonunu bulmak için üç ayrı özellik seçimi senaryosu gerçekleştirilir. Tüm modellerin eğitim süresi de, özellik azaltmanın hesaplama maliyetleri üzerindeki etkisini görmek için hesaplanır. Sonuçlar, en iyi modellerin XGB, LGBM ve RF modelleri olduğunu göstermektedir. Bunun yanı sıra, ML topluluk modeli iyi bir performans göstermektedir. Özellik seçim yöntemi, özellikle XGB ve LGBM gibi yüksek hesaplama modelleri için eğitim süresini önemli ölçüde azaltırken, performansta ciddi bir düşüş olmaksızın gerçekleştirmiştir. En iyi elde edilen doğruluk, %99,87 ile XGB modeline aittir. Çalışma ayrıca güncel en iyi metodolojilerle kıyaslanmıştır. Kıyaslama, mevcut çalışmanın tüm önceki çalışmalardan daha üstün olduğunu kanıtlamaktadır. Gelecekteki çalışmalar, performansı iyileştirmek için hiperparametre optimizasyonu düşüncesine odaklanabilir.

Anahtar Kelimeler: Makine Öğrenmesi, Derin Öğrenme, Fidye Yazılımı Tespiti, Çok Sınıflı Sınıflandırma, Öznitelik Seçimi, Güvenlik.

Bilim Kodu **:** 92432

ACKNOWLEDGMENT

First and foremost, I wish to express my heartfelt gratitude to Allah Almighty for his divine guidance and blessings throughout my educational journey. Additionally, I extend my thanks to Karbuk University for affording me this opportunity to undertake my graduate studies. A special acknowledgment goes to my supervisor, Assist. Prof. Dr. İsa AVCI, and the dedicated professionals at this renowned institution. I sincerely appreciate my husband, a steadfast pillar of support whose consistent encouragement and faith have been instrumental in my academic pursuits. My thanks are also extended to my colleague, Mr. Abbadullah H. SALEH, for his valuable advice and generous knowledge sharing.

Lastly, I wish to convey my thanks and my enduring love for Turkey and my homeland, Iraq.

CONTENTS

Page

Page

Page

LIST OF FIGURES

Page

Page

LIST OF TABLES

Page

ABBREVIATIONS

ML : Machine Learning
DL : Deep Learning
EL : Ensemble Learning
DT : Decision Trees
CNN : Convolutional Neural Networks
RF : Random Forests
SVM : Support Vector Machines
IoT : Internet of Things
K-NN : K-Nearest Neighbor
LR : Logistic Regression
CSPE-R : Cost-Sensitive Pareto Ensemble strategy
CAE : Contractive Auto Encoder
ELM : Extreme Learning Machine
SDN : Software Defined Network
LSTM : Long-Short Term Memory
TP : True Positives
TN : True Negatives
FP : False Positives
FN : False Negatives
XGBoost : Extreme Gradient Boosting
AdaBoost : Adaptive Boosting
LGBM : Light Gradient Boosting Machine
HFSBAR : Hybrid Feature Selection Based ANOVA and RF
ROC : Receiver Operating Characteristic
AUC : Area Under Curve

PART 1

INTRODUCTION

Ransomware is a type of malware that can cause damage to information by encrypting it or preventing users from getting their files [1], [2]. Ransomware types are developing and increasing which makes the need to detect and classify them an essential network security topic [3], [4]. Feature selection along with the ransomware detection process can provide much more accurate models [5], [6]. This study will try to detect and classify the network ransomware types based on a new method combining feature selection with the efficiency of fused Machine Learning (ML) and Deep Learning (DL) best models.

1.1. SYSTEM AIMS

1- Detect ransomware through the network before it encrypts files and valuable used information.
2- Classify many types of ransomware depending on the feature selection of the best features.
3- Select the most appropriate features (predictors) that can precisely identify the ransomware and avoid redundant features, leading to a faster and more efficient detection rate.
4- Utilize the efficiency of machine learning and deep learning algorithms for ransomware detection.
5- Using large datasets (with a large number of features and records) to evaluate the proposed methods effectively.

1.2. SYSTEM IMPORTANCE

Most ML and DL studies in the field of network security are guided to malware and intrusion detection [7]. Few studies were introduced in the field of ransomware detection and fewer ones were developed to deal with ransomware classification. The current study will deal with the problem of ransomware detection and type classification. The development in ransomware types makes the importance of such studies very critical. Using new technologies in ML and DL, especially fusion and ensemble learning along with the feature selection for redundant data elimination, will help improve the performance of ransomware detection and classification (our study's main importance).

1.3. SYSTEM OBJECTIVES

1- Improve the traditional methods of ransomware detection by using more efficient models and applying enhancement techniques like fusion and ensemble learning.
2- Analyze a ransomware dataset consisting of 203,556 rows, 85 features, and 10 types of ransomware.
3- Devolve and compare multiple machine learning (decision trees DT, random forests RF, support vector machines SVM, XGBoost, etc.) and deep learning algorithms (1D convolutional neural networks 1D-CNN, Dense, long-short term memory LSTM, etc.) for ransomware detection.
4- Evaluate and compare performance using different feature selection scenarios.
5- Apply many experiments to analyze the effect of different hyperparameters (learning rate, preprocessing, layers architectures, etc.) on the performance of the models.
6- Compare the current proposed methods with the state-of-the-art related works in the field of ransomware detection to define the efficiency and limitations of the current study.

1.4. SCOPE OF THE STUDY

This study focuses on the development of many ML and DL models for the aim of ransomware detection and classification. The study will build a hybrid feature selection algorithm based on both filter and wrapper feature selection methods to make a powerful feature selection method (which was not introduced before) that will select the best subset for ML and DL training steps. The study will evaluate the trained models using an open-source big dataset with high dimensionality complexity. The study will use the fusion techniques to ensure the best performance.

1.5. STUDY CONTRIBUTION

The main contribution of the current study can be concluded as follows: Build a comprehensive ransomware detection system based on the efficient ransomware feature selection, fusion, and ensemble learning of the best ML and DL models. The system will be based on a big dataset (a large number of records and high dimensionality). The system will improve the security of network processes by detecting potential ransomware before encrypting files or damaging data.

1.6. STUDY PROBLEM AND PROPOSED SOLUTION

The main problem of the current research is that ransomware attackers can encrypt valuable data, leading to data loss. This problem could be essential, especially for financial and military applications. The existing ransomware detection methods are good in detecting some types of ransomware, but to build a comprehensive ransomware detection system, the traditional methods failed. In the current research, a hybrid feature selection algorithm and fusion and ensemble deep learning models will be used to detect and classify ransomware types.

The current study will solve many problems of the existing systems:

1. Many existing systems didn't take into account different types of ransomware, rather, they classified actions into ransomware of benign. In the current study,

the multi-class classification of 10 types of ransomware will be used to build a more efficient ransomware detection system.

2. Most existing systems used small or moderate datasets, leading to a lack in many cases of ransomware actions. In the current study, a big dataset will be used.
3. Most existing systems used a dataset with a small number of predictors (low dimensionality complexity). In the current study, 85 features (predictors) will be used (which is considered a high-level dimensionality complexity). The efficient feature selection algorithms will be used (fusion of two different fusions can be used to get the best combination of features).
4. Many existing systems used specific types of deep learning models, which were good for detecting but not classifying all potential types of ransomware. In the current study, fusion, and ensemble learning are proposed to build the most effective ransomware detection and classification system.

1.7. SYSTEM HYPOTHESES

The current system introduces the following hypotheses:

1- The hybrid feature selection algorithm will enhance the performance by reducing the training time for all trained models and preserving high performance.
2- Machine learning and deep learning models can be used for the aim of ransomware detection and classification.
3- The fusion of the best-trained models will enhance the performance of the ransomware detection and classification process.

1.8. SYSTEM BLOCK DIAGRAM

The proposed ransomware detection model is described in Figure 1.1. In the first step, the ransomware dataset is acquired. In this study, the "Android Ransomware Detection" dataset is suggested. This dataset is available for free on Kaggle [8]. Then, the preprocessing steps are applied to transform the dataset into a better form. The

preprocessing steps include cleaning, encoding, normalization, etc. In the third step, the dataset is split into train, validation, and test sets. The training and validation sets will be used in the training process, while the test set will be used in the evaluation step. The fourth step is the feature selection process in which the most significant features of the pre-processed dataset will be chosen, and the redundant features will be dropped. The training of the machine learning and deep learning models is the next step in which many models will be used. The best models will be used in the fusion step to enhance the performance. The final step is the evaluation step by which the individual and fused models will be evaluated using many performance evaluation metrics.

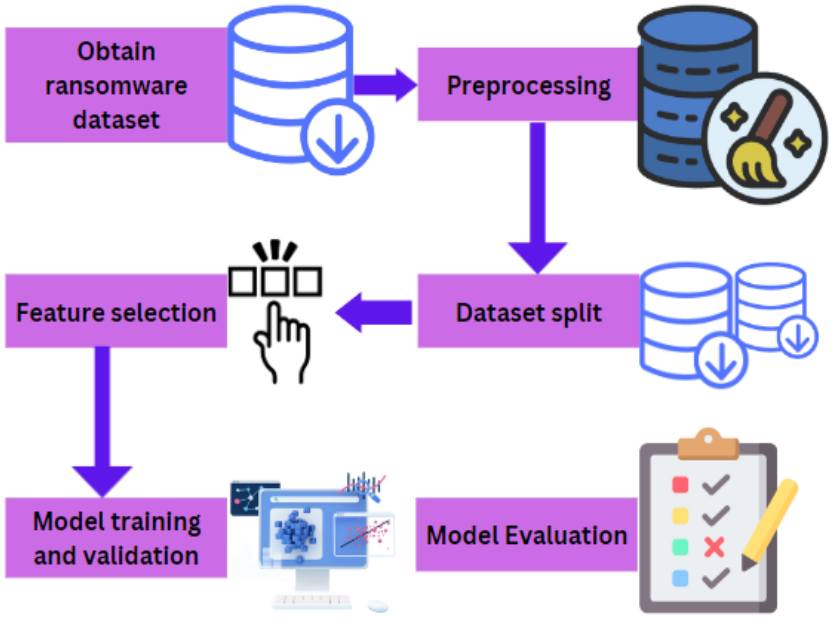


Figure 1.1. Block diagram of the ransomware detection model.

The rest of the thesis will be organized as follows: The literature review and related work will be introduced in Chapter 2, while Chapter 3 will include a description of the proposed methodologies and tools. Chapter 4, on the other hand, will show the experiments and results, while the discussion and conclusion will be listed in Chapter 5.

PART 2

LITERATURE REVIEW

Malware, known as ransomware, can harm data by encrypting it or prohibiting users from accessing their files [1, 2, 9]. Locker ransomware and cryptography ransomware are the two basic categories under which ransomware is typically divided. The user's files are encrypted in the second case, whereas in the first, just the critical computer files are [10]. According to [11], the cost of ransomware attacks is increasing significantly and will be very big by 2030. Figure (2-1) shows the expected damage costs caused by ransomware attacks from 2015 to 2030 [11].

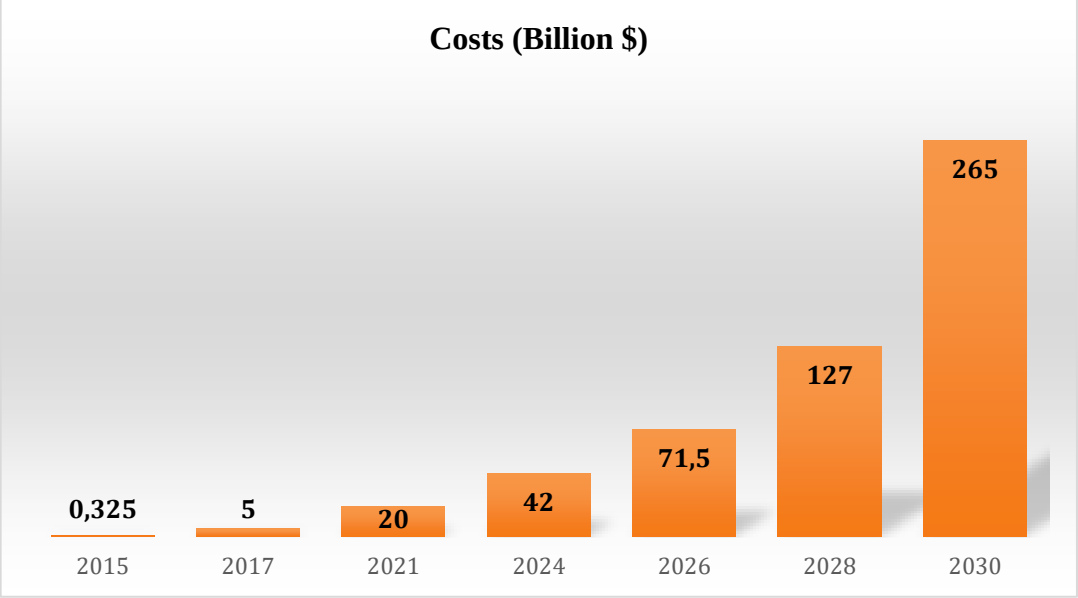


Figure 2.1. Costs of damage caused by ransomware attacks [11].

Most of the world's countries are affected by ransomware attacks every day. However, the USA is considered the most affected country by ransomware attacks. Table 2. shows the number of ransomware attacks related to each country according to statistics for 2022 [12].

Table 2.1. Number of ransomware attacks corresponding to each country according to statistics of 2022 [12].

Country	USA	UK	Spain	Brazil	Germany	Colombia	Netherlands	Italy	Norway	Australia
Num. Attacks	217.49	71.35	52.68	21.81	20.17	15.52	13.64	12.47	8.36	7.62

After attacking devices, the attacker requests ransomware from the victim to send the decryption key of the encrypted files. According to statistics, 58% of victims agree with the attacker's will and pay the payment. However, only 10% of those who decide to pay can't restore their files even after paying [13]. In some cases, victims pay double payments to decrypt their files.

The bad issue about ransomware attacks is that they are the most frequent malware attacks since they can attack all life fields, including financial companies, banks, insurance systems, government facilities, healthcare facilities, etc. [14], [15], [16].

2.1. RANSOMWARE TAXONOMY

Ransomware is commonly classified into two main categories: crypto and locker [17]. Each one of those two types includes several sub-types. Both varieties of ransomware target a victim's device, exploiting operating system weaknesses as soon as the victim opens the attachment or clicks the link in the email. In the crypto kind Figure 2.2, the ransomware targets and encrypts just particular user files based on their extensions. The adoption of a 24-bit encryption method, which prevents users from decrypting the encrypted data without obtaining the key, is the fundamental issue that this ransomware-type creates [18]. Infected websites and other exploit kits can be used to disseminate ransomware. To avoid being apprehended, the assailant in this instance, demands a Bitcoin ransom. Occasionally, ransomware uses network propagation weaknesses in the operating system to its advantage [18].

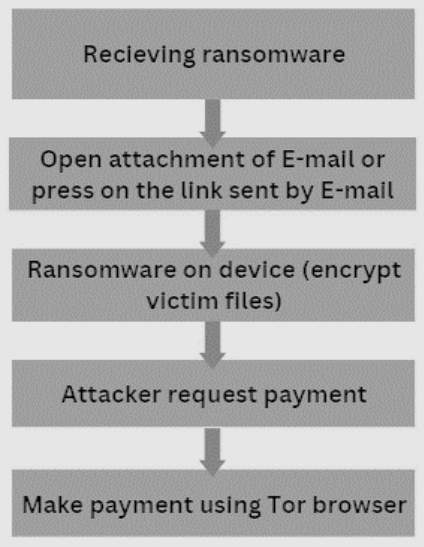


Figure 2.2 Crypto-type ransomware attack steps.

Locker ransomware, on the other hand, works in different ways. As soon as the victim receives an email with an attachment or link and opens these materials, his device is locked, and gives him instructions to follow to unlock it. In some cases, after paying the ransom, user devices are still locked [19]. Figure 2.3 illustrates the main processes of the locker ransomware type.

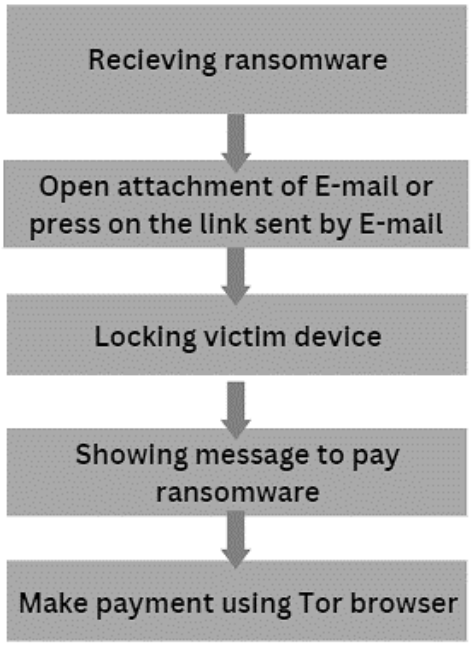


Figure 2.3. Locker -type ransomware attack steps.

2.2. RANSOMWARE ANALYSIS

There are three types of ransomware analysis: static, dynamic, and hybrid. Figure 2.4 shows these types [20, 21]. Static analysis: With this type, it is possible to determine whether a file is infected by looking at features like hashes, opcodes, byte sequences, etc. Without having to execute it, the binary file is checked. With this method, a lot of information may be extracted, including malware commands, targets, control infrastructure, etc. The most popular static analysis programs are regarded as PeStudio, ExeInfo PE, HxD, CyberChef, and IDA Pro. This method is quick, but it can't find ransomware that uses code obfuscation techniques. As part of the dynamic analysis, a sample of the ransomware is run in a controlled environment, such as a virtual environment (sandbox), which allows the researcher to observe the ransomware behavior without endangering the system.

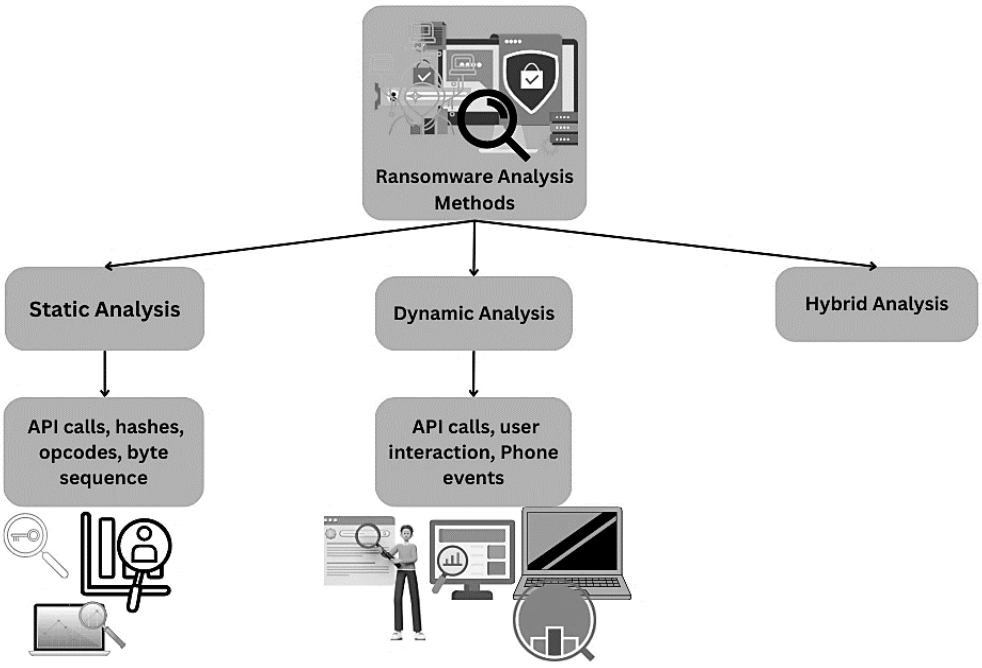


Figure 2.4. Ransomware analysis types.

2.3. ARTIFICIAL INTELLIGENCE RULE IN RANSOMWARE DETECTION

Artificial Intelligence AI plays an essential role in detecting malware and ransomware in mobile devices, IoT infrastructures, and network devices. Machine learning ML and

deep learning DL-based methods are the most used AI techniques to detect ransomware. However, ML is limited to the scanning process only. Not only detection but also prevention techniques are also essential to protect users' files and devices. ML and DL can both be used to do these missions with a transcendence of the DL techniques against the ML models, especially when training models with vast data [22, 23]. Figure 2.5 shows the general steps used in ML/DL models to detect ransomware. First, the dataset is collected and pre-processed to remove undesired data (data cleaning) and transform data into a suitable form (encoding). Then, the feature selection and extraction methods are used to define the most appropriate features to be included in the training process. Later, the dataset is split into train, validation, and testing. The training and validation set is used in the training process, while the test set is used in the evaluation process [24-26].

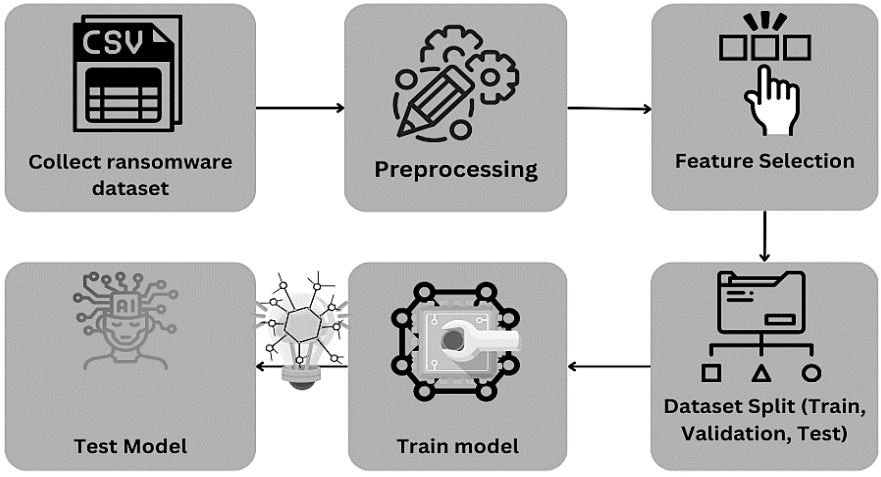


Figure 2.5. ML/DL ransomware detection steps.

2.4. RELATED WORK

In this section, the related work will be classified into two subcategories: the ransomware detection studies and the ransomware prevention studies.

2.4.1. Ransomware Detection Studies

Sgandurra et al. [27] introduced the EldeRan-based ML model to detect ransomware on a dataset consisting of 582 ransomware and 942 normal cases (too small dataset). They used the feature selection methods and obtained an accuracy of 96.34%. Their study utilized a small dataset. In 2017, Vinayakumar et al. [28] implemented shallow and deep models for ransomware detection purposes. They applied the dynamic analysis of seven different ransomware types, and Application Programming Interface (API) calls were collected as features for their AI models (support vector machines SVM and multi-layer perceptron MLP).

Cohen and Nissim [29] introduced an ML-based ransomware detection model based on random forests. Their method detected anomalous virtual machine states besides the other ransomware. They achieved an accuracy of 96.6% using Virus Total, open malware, VXVault, and Zelster datasets. They detected five types of ransomware.

Shaukat et al. [30] proposed a ransomware detection system based on ML algorithms and gradient boosting algorithms. They utilized a dataset consisting of 475 ransomware and 442 benign records. They achieved an accuracy of 98.25%. They used a small dataset and the binary classification method.

Maniath et al. [31] used the LSTM deep model for the aim of ransomware detection. They applied no feature selection algorithm. They utilized a dataset of 157 ransomware only. The results showed that the test accuracy was 96.67%. Their used dataset was too small besides the fact that they performed the binary classification problem (ransomware or normal).

In a study by Takeuchi et al. [32], researchers used the SVM model for ransomware detection. There were no feature selection methods. Their study used a dataset of 276 ransomware and 312 benign. The results indicated an accuracy of 97.18%, recall of 97.13%, and precision of 98.34%, respectively. The dataset used was too small, and they used only 4 types of ransomware.

Hwang et al. [33] introduced a two-stage ML model for ransomware detection based on RF and Markov models. Their experiments were conducted on a dataset of 2507 ransomware and 3886 normal cases. They applied no feature selection methods and achieved an accuracy of 97.3%. Although their dataset was not too small it had only 6063 records. They used the binary classification of only two types (ransomware and benign).

Zuhair et al. [34] proposed a hybrid ML model consisting of Naïve Bayes (NB) and DT classifiers for the aim of multi-class classification ransomware detection. Their dataset consisted of 35000 ransomware and 500 benign files. No feature selection methods were applied. The experiments showed that the ransomware classification accuracy was 97%.

Manavi and Hamzeh [35] proposed a ransomware detection approach using features extracted from the PE headers of the executable files. They used the convolutional neural network CNN as a classifier and achieved an accuracy of 93.33% on a subset of the Virus Share dataset consisting of 1000 ransomware and 1000 benign records. They detected only four types of ransomware. Almousa et al. [36] introduced a new approach to ransomware detection using API calls and machine learning algorithms. They used the dynamic analysis method and sandbox platform. They trained many ML models, including RF, K-NN, and SVM, on 58 ransomware and 66 benign files. They achieved an accuracy of 99.18%. Their method detected 12 ransomware types.

A study by Masum et al. [37] proposed a feature-selection-related method by adopting various ML methods to classify the security level for the prevention and detection of ransomware. Their study used many machine learning algorithms, including DT, RF, Logistic Regression (LR), and Neural Networks (NN). The researchers showed that their method outperformed the existing ones. However, their used dataset contained only three types of ransomware. The results indicated an accuracy of 96.78%.

Zahoora et al. [1] introduced a Cost-Sensitive Pareto Ensemble strategy (CSPE-R) to detect Ransomware attacks. The proposed framework exploited the unsupervised deep Contractive Auto Encoder (CAE) to transform the underlying varying feature space to

a more uniform and core semantic feature space. The experimental results showed that the proposed CSPE-R framework performed well against zero-day ransomware attacks. Their dataset consisted of 582 ransomware and 942 (very small dataset). Their approach achieved a recall of 99%.

Alissa et al. [38] suggested a ransomware detection model based on dwarf mongoose optimization and ML algorithms. The enhanced krill herd optimization algorithms were used for the feature selection step. An Extreme Learning Machine (ELM) classifier was used as an ML model. The used dataset consisted of only 840 samples (420 normal and 420 ransomware). Their results indicated accuracies between 98.69% and 99.4%. They applied the binary classification (ransomware or normal) without any classification of ransomware types.

Fernando et al. [6] proposed a new feature selection algorithm for a ransomware detection system called FeSA. The functional technique was compared to other systems, such as evolutionary search and harmony search, to measure the degree of ransomware exposure, recall, false negative rate, and accuracy. However, the proposed mechanism did not adequately account for the impact on the victim node's boot record. Additionally, the study did not investigate the ransomware identification and prevention paradigm at the gateway, which is a critical location for effective anomaly detection and response. Their study used many ML algorithms, including SVM, DT, K-NN, and RF. They achieved an accuracy of 98.4% on a CIC-IDS2017 dataset (78 columns).

Bold et al. [39] introduced a ransomware detection system based on machine learning algorithms, including SVM, K-NN, LR, RF, and neural networks. They achieved accuracies between 95.56% to 97.61% on a dataset of 730 ransomware and 735 benign. Their dataset was small, they didn't use any feature selection methods, and they applied binary classification (ransomware or normal).

Rani and Dhavale [40] used many machine-learning algorithms for ransomware detection. They used DT, RF, K-NN, SVM, XGBoost, and Logistic Regression LR. The best and worst models were XGBoost and K-NN, with accuracies of 98.21% and

93.64%, respectively. Their models were trained using a dataset consisting of 582 ransomware and 942 benign files with 12 different ransomware categories.
A weighted minimum redundancy maximum relevance WmRmR technique was proposed by Ahmed et al. [41]. Their approach decreased the number of selected features in order to minimize the computational time. Their methodology detected the most essential system calls. They collected a dataset of 1,500 ransomware and benign samples. They used different ML classifiers in the classification phase, including K-NN, SVM, DT, LR, and K-NN. The best-obtained accuracy was 99.3%.

Yassen [42] studied the effect of dataset ages on the performance of ransomware detection systems. He collected a dataset of 6545 ransomware from 2008 to 2020 and investigated many ML models, including SVM, DT, LR, K-NN, and RF. The study extracted 161 features of the old ransomware and 274 features of the new ransomware. The results indicated the best accuracy of 97.5% of the RF model. His study utilized a moderate dataset without any fusion of the ML models. His study focused on the binary classification.

Herrera-Silva et al. [43] used machine learning algorithms for ransomware detection on a very small dataset consisting of 20 ransomware and 20 benign files. They used DT, RF, and neural network models. Their approach achieved 99% accuracy. However, their dataset was too small, and the binary classification was used. Besides that, no feature selection algorithms were used.

A new deep learning model titled "Spline Interpolation envisioned neural networks for ransomware detection (SINN-RD) model was proposed by a study by Singh et al. [4]. They applied the normalization step to produce new features of the log records. The experiments showed that the obtained accuracy was 99.83%.

Silva and Alvarez [43] introduced a dynamic analysis and machine learning-based ransomware detection study. They performed the feature selection mechanism to identify the best subset of features for the aim of the ransomware detection process. They utilized a public common dataset consisting of only 2000 records of both ransomware and benign records. They used well-known ML models, including Naïve

Bayes, boosted trees, RF, and neural networks. They used a small dataset.

The Minerva ransomware detection method was suggested by a study by Hitaj et al. [44]. Their method was file-based in which they used a time window to build the behavioral profiles of the benign and ransomware files. They utilized two ransomware datasets: the ShieldFs and Cerberus, and the RF algorithm. Their results indicated that their method achieved an accuracy of 99.45%. They applied the binary classification without defining any ransomware type.

2.4.2. Ransomware Prevention Studies

Prevention is the process of protecting resources from ransomware attacks by studying many resource properties. However, few researchers have studied the prevention idea [45].

Song et al. [46] proposed a ransomware prevention method in Android systems based on a statistical analysis of the I/O rates, memory consumption, and CPU usage. Their method depended on resource monitoring to define any possible processes with undesired performance.

Lee et al. [56] designed a ransomware prevention system to monitor networks, files, logs, and servers in real time to define any possible undesired processes. Their method was based on abnormal behavior analysis in a cloud environment called CloudRPS. Results showed that the detection process could be enhanced, and the damage might be reduced using the proposed CoudRPS system.

In their study, AlSabeh et al.[47] checked specific actions that ransomware performs to identify its execution environment. Their proposed methodology includes intercepting Windows API calls to detect if a process attempts to identify its surroundings. The approach terminates the process when such activity is detected, preventing a potential attack. They achieved an accuracy of 91%.

Research by Wani and Revathi [48] focused on the threat of ransomware to IoT devices. They introduced a detection model for detecting ransomware attacks on IoT systems. Their proposed solution involves using a Software Defined Network (SDN) gateway to monitor incoming traffic to IoT networks actively. Additionally, ransomware attacks are identified and mitigated through policies created in the SDN controller. Their method achieved an accuracy of 97.9%.

Ibrahim et al. [49] proposed a ransomware prevention system in the cloud –Enabled PureOS. They developed a parameterized categorization strategy for functional classes and proposed features based on real-world ransomware samples. They also introduced a set of criteria that specified the most frequently observed attributes and analyzed both behavior and insights. They utilized a distinct operating system and specific cloud platform to enable remote access and file collaboration throughout the experimental infrastructure. They successfully detected and prevented both state-of-the-art and modified ransomware attacks. The combined logs revealed a consistent and satisfactory detection rate of 89%.

A hidden decoy file was used by Lin and Lee [50] to prevent ransomware from destroying users' files. The technique was based on placing a decoy file in the computer and letting a monitoring process track its status. Once the decoy file is changed or damaged by ransomware, the computer is shut down immediately as a prevention act. The proposed method achieved a protection rate of 98.82%.

2.4.3. Ransomware Datasets

There are many ransomware datasets. Some of them contain samples of ransomware and normal tasks without any classification of the ransomware types, while a few of them contain different types of ransomware and can be used for classification purposes.

Some of these datasets are small, while others contain a large number of records that are more dependable and universal. Figure 2.6 contains a comparison between the most commonly used ransomware datasets in the literature [51].

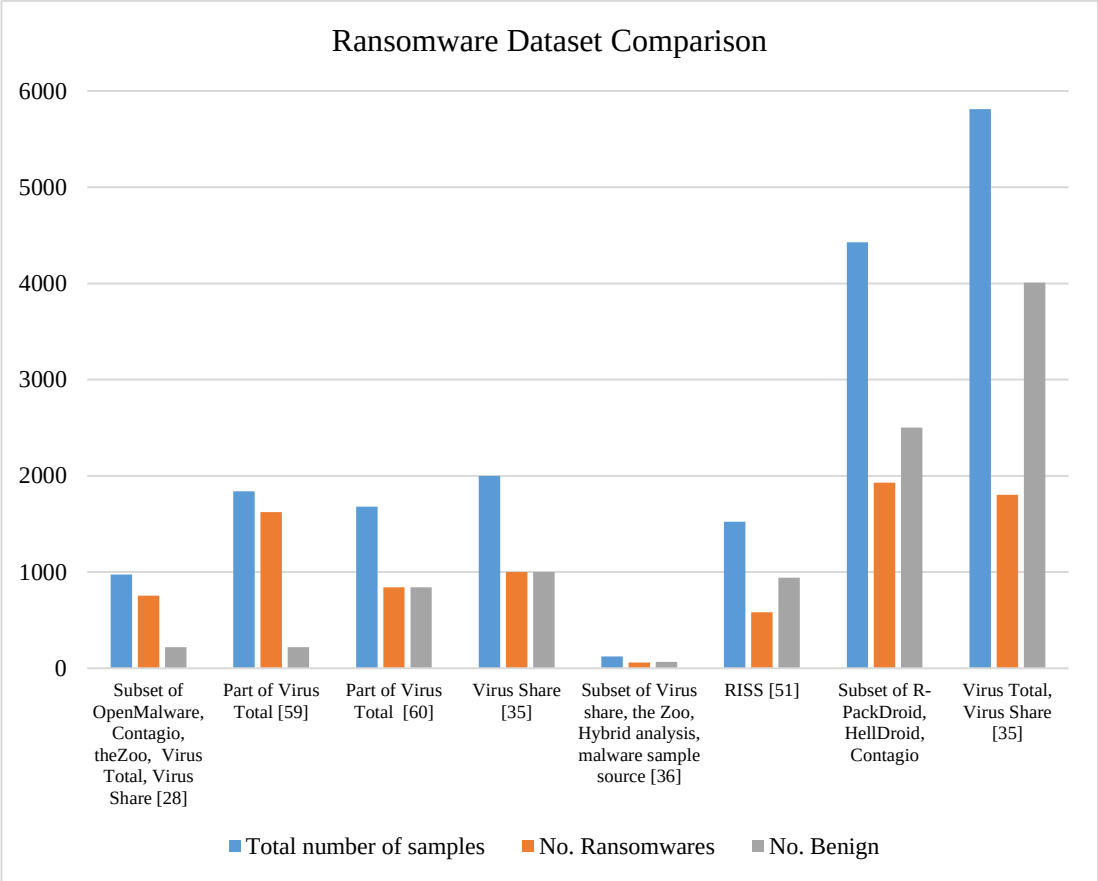


Figure 2.6. Ransomware datasets comparison.

The main limitations of the previous studies can be concluded in the following:

1. Most of the previous studies used binary classification (ransomware or normal) without any type of classification.
2. Most of the previous studies used small datasets which reduced the universality of these systems.
3. Some of the studies didn't use the feature selection technique (this led to high computational training time).
4. Only too few studies used some form of fusion or ensemble learning.
5. Although small datasets, some studies recorded low or moderate accuracies.

PART 3

MATERIALS AND METHODS

3.1. THE PROPOSED METHODS

The proposed methods of the current study consist of the following steps that are illustrated in Figure 3.1:

1- Obtain dataset: The main part of the study is the dataset since every next step is based on the selection of the dataset. For this study, the "Android Ransomware Detection" dataset is suggested. This dataset is available for free on Kaggle [8]. It includes 10 different ransomware types and the normal condition so it's a very good choice for the current mession.
2- Preprocessing: In this step, many tasks will be applied to the dataset to transform it into the most suitable case. The preprocessing steps include data cleaning, label encoding, feature encoding, and balancing.
3- Dataset split: Into training, validation, and test sets. Training and validation sections are used through the training process, while the test set is used for model evaluation after training is ended.
4- Feature selection: This step is essential in this system since it will select the most promised features to serve as predictors for the next ML and DL training process. In this study, a hybrid method of wrapper and filter methods to get a robust, efficient feature selection algorithm. The output of this step will be the selected subset of features that will be used for the next training steps.
5- Model training and validation: After getting the best subset of features, many ML and DL models will be trained using this subset of features. Further, the same ML and DL models will be trained using the original entire features to compare the two scenarios in terms of time and accuracy to identify the efficiency of the proposed feature selection method.

For this step, we suggest using the following models:

A- RF, DT, SVM, XGBoost, Adaboost
B- Ensemble of ML models
C- Fusion of ML models using score-level fusion by which the matching score of the individual ML models will be computed then the final fused score will be conducted using equation (3.1):

$$Score_F = \frac{1}{\sum w}\left(\sum_{i=1}^{m} w_i * score_i\right) \quad (3.1)$$

Where $Score_F$ is the final fused score, m is the number of individual models, w_i is the weight of individual model i, and the score is the score of the individual model score. Weight is a scalar value that is used to give the most attention to the best individual model so that the model with a high score has a better performance than the model with a low score.

D- 1D-CNN model
E- LSTM model
F- Ensemble of DL models
G- Fusion of DL models using score-level fusion.

6- Model evaluation: in this step, the evaluation metrics are used to assess the trained model using the test set. This step is essential to ensure that the best model is built.

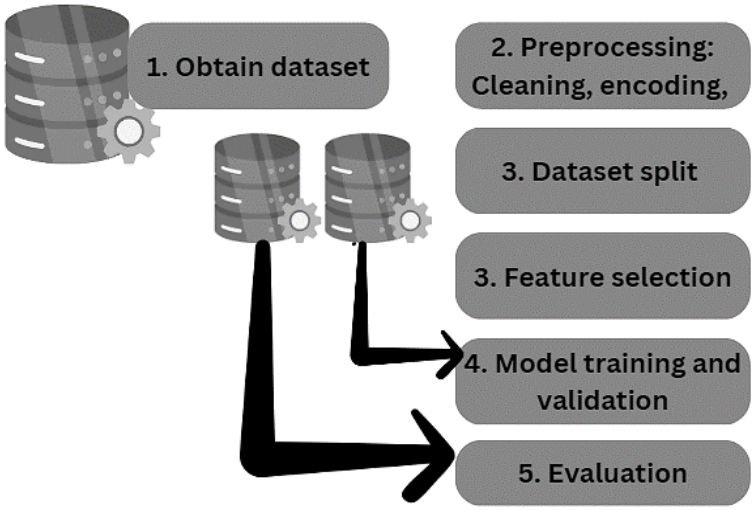


Figure 3.1. The proposed ransomware detection method.

3.2. MATERIALS

3.2.1. Dataset

In the current study, the "Android Ransomware Detection" dataset [8]. This dataset contains information about Android devices, including the monitoring records of the network. This information is used to define the type of network processes (normal or ransomware). The dataset consists of 392034 records and 84 features (predictors). The target of the dataset is the "label" column, which includes 11 different ransomware types and the normal condition. This dataset is the best choice for the current study since it includes a large number of samples and 11 various types of ransomware.

3.2.2 Software

The current study will be implemented in Python programming language. Many libraries will be used, including the following:

1- Numpy: matrix processing library.
2- Sklearn: sickit learn library which is used to build, train, and evaluate machine learning models.
3- Opendatsets: To load CSV files.

4- ipython-autotime: used to compute training time.
5- Tensorflow and Keras: to build and train deep learning models.
6- Matplotlib: for plotting options.

3.3. PERFORMANCE EVALUATION

Performance evaluation is the last step of any ML model since it contains the evaluation of the trained models using many performance metrics to assess the performance and suggest modifications.
In the current study, the following evaluation metrics are used [52]:

1- Accuracy: this metric calculates the general accuracy of the model and is given as Equation (3.2) shows.

$$ACC = (TP+TN)/(TP+TN+FP+FN) \quad (3.2)$$

2- Precision: precision computes the percentage (ratio) of the true predictions that are made by the ML model.

$$Precision = TP/(TP+FP) \quad (3.3)$$

3- Recall: recall calculates the percentage of samples that are correctly classified by the ML model (among all true and false samples) as shown in Equation (3.4).

$$Recall = TP/(TP+FN) \quad (3.4)$$

4- F1-score: is a mixed measure of precision and recall and is given as Equation (3.5) illustrates.

$$F1\text{-}score = (2*Precision*Recall)/(precision+recall) \quad (3.5)$$

Where, TP: true positives; the number of correctly accepted samples of a specific class of the dataset.

TN: True Negatives; the number of correctly rejected samples of a specific class of the dataset.
FP: False Positives; the number of incorrectly accepted samples of a specific class of the dataset.

FN: False Negatives; the number of incorrectly rejected samples of a specific class of the dataset.

Training Time: The detailed confusion matrix of the partial classes will also be drawn to define the individual evaluation metrics for each class of the dataset.

The accuracy, precision, and F1 score will also be computed for all individual classes using the classification report. In the end, we will define the best training scenario which has the best performance.

3.4. PROPOSED ML METHODOLOGIES

In this study, many individual and fused ML and DL models are proposed. In this section, a detailed explanation of the used algorithms and methodologies will be introduced.

3.4.1. Decision Trees

Decision trees are one of the most commonly used machine learning algorithms that divide the feature space into regions (splits) based on the feature values (gain). DT makes predictions by following a path from the root node (initially chosen feature) to a leaf node, making a tree-like shape, where each internal node represents a decision based on a specific feature. Two main concepts are used in the branching operation of a decision tree (Gini Impurity and information gain). The former calculates the degree of uncertainty among a set of samples [53]. The Gini impurity is measured using the

distribution of class labels in a specific node of the DT. It's calculated as Equation (3.6) shows.

Gini Index = 1 - (sum of squared proportions of each class label in the node) (3.6)

Information gain, on the other hand, can also be used to select the best feature for branching decisions. It computes the entropy reduction (degree of loss in information) which happens due to splitting data based on a selected feature.

Information Gain = Entropy(parent) - [weighted average of entropies of child nodes] (3.7)

Figure 3.2 shows the DT model [54].

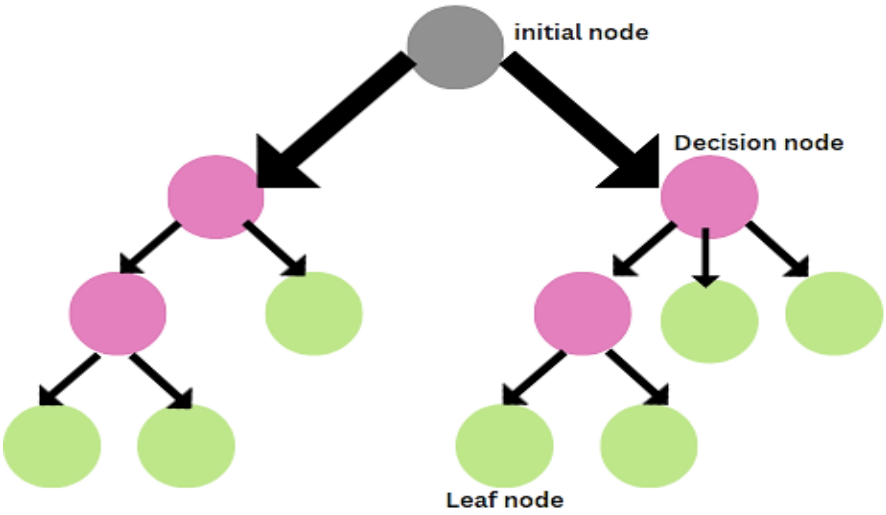


Figure 3.2. Part of DT model architecture.

The main advantages of DT are the ease of use, low training time, handling of both numerical and categorical values, the ability to capture the nonlinear relationships among features, and the ability to handle missing values and outliers of datasets.

The main limitations of the DT model are the prone to overfitting when going too deep in trees, the sensitivity to small variations in data, the lack of smoothness, and the tendency to bias when dealing with imbalanced datasets.

3.4.2. Random Forests

Random Forests are an ensemble learning model combining many decision trees in a unified model called an ensemble. Each tree is built on a random subset of features; besides, it uses bootstrapping to produce different training sets. The final prediction is made by fusing the predictions of the individual trees. Figure 3.3 shows the architecture of the random forest model [54].

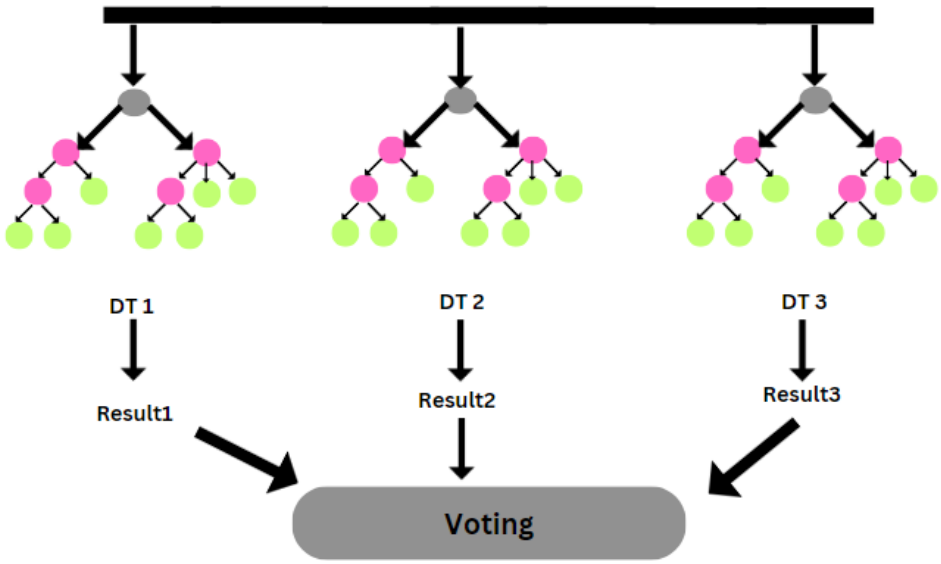


Figure 3.3. RF model architecture.

The main advantages of an RF classifier are that it reduces overfitting compared to the single DT, it can handle high-dimensional datasets, it's also robust to outliers, it can be used to select the best combination of features, and can be used for both classification and regression problems. However, it requires a higher computational time than DT, it also requires tuning of hyperparameters, and it may not perform well on datasets with very correlated features.

3.4.3. k-Nearest Neighbors (k-NN)

K-Nearest Neighbors is a non-parametric model classifying samples based on the majority class of their k-nearest neighbors in the feature space of a given dataset [55]. The main advantages of the K-NN model are easy implementation and training, it can also handle multi-class classification problems, it has good robustness against noise in

data, it has no training phase, and fits only small datasets. However, K-NN doesn't fit the large dataset, it also requires a good choice of K parameter, and it lacks interpretability and requires scaling of features. Figure 3.4 explains how the K-NN model works.

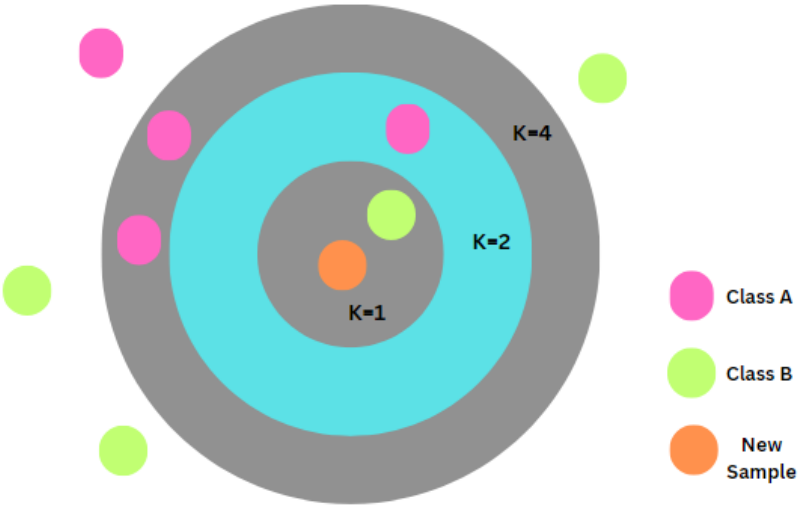


Figure 3.4. K-NN concept.

3.4.4. Extreme Gradient Boosting (XGBoost)

XGBoost is an enhanced implementation of the gradient boosting (GB) model by which an optimization process is applied. XGBoost consists of an ensemble of weak prediction models sequentially, optimizing a differentiable loss function. The main advantage of this model is that it has high accuracy, it can handle complex and non-linear relationships of data, it supports different objective functions and evaluation metrics, it's scalable and fits large datasets, and it provides a feature selection and estimation method. Although these are good attributes, XGBoost is computationally expensive, requires hyperparameter tuning, and lacks interpretability compared to other simple models.

Figure 3.5 illustrates the boosting ensemble concept that the XGBoost applies. In the boosting algorithm, many weak models are used in sequential order so that each model learns from the previous model's error to minimize the entire error, and in the end, the last model produces the least error rate and the best performance.

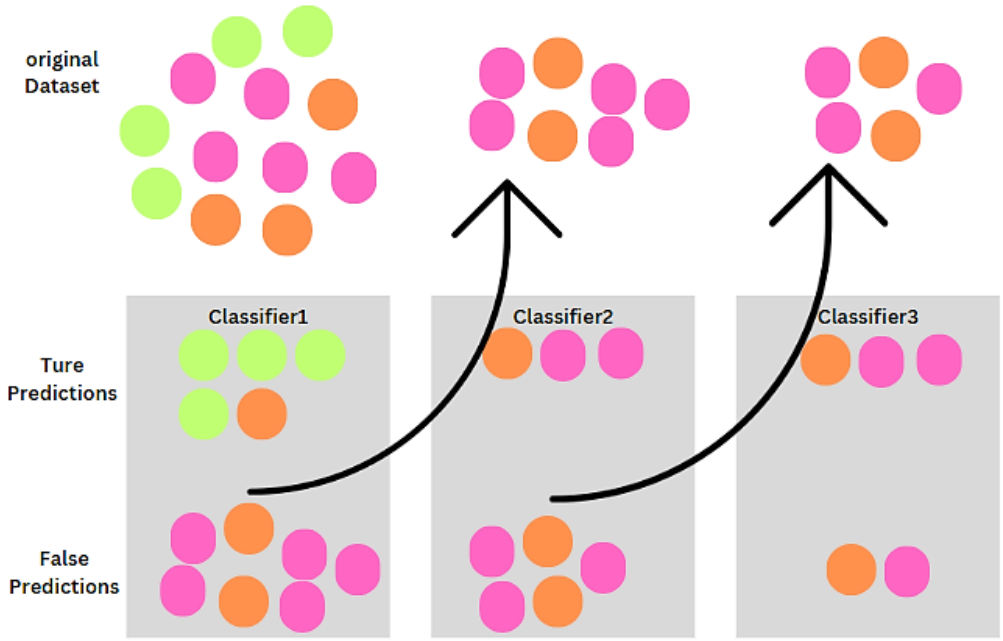


Figure 3.5. Boosting ensemble concept.

3.4.5. Adaptive Boosting (AdaBoost)

AdaBoost is another ensemble boosting model that combines multiple weak classifiers to create a powerful classifier [56]. It specifies weights for each instance in the training set, focusing more on misclassified samples in the next iterations. The main advantages of this classifier are its ability to handle both numerical and categorical data, good performance, less prone to overfitting than individual models, and ability to handle imbalanced datasets. However, this model is sensitive to noise and outliers, it is also computationally expensive, it requires hyperparameters tuning, and it suffers from overfitting in case of choosing bad, weak classifiers.

3.4.6. Light Gradient Boosting Machine (LGBM)

LightGBM (Light Gradient Boosting Machine) is one of the best ML models used for classification and regression tasks [57]. It is mainly based on the gradient boosting algorithm but it is more efficient and scalable. LGBM is very suitable for large-scale datasets. As in the AdaBoost model, LGBM creates an ensemble of weak models (decision trees as default) sequentially. LGBM uses gradient-based learning to optimize the ensemble of individual models. First, the model is initialized with the

mean value of the target column as an initial result. Then, the gradient and Hessian of the loss function are computed concerning the prediction of the current individual model in the ensemble. After that, a decision tree is built to fit the negative gradient. This tree structure is grown using a leaf-wise methodology by which the split with the biggest reduction in loss value is chosen as the candidate split in each step.

For the next step, the model is updated and a new tree is added with a decreased learning rate. Next, the operation of creating new trees is repeated either for a specific number of epochs or when we reach a stopping criterion (minimize the loss function to the desired value). Finally, the ensemble of all created trees is constituted and weighted by the learning rate.

The main advantages of this model are high performance, the ability to handle large-scale datasets with a high number of features, low computational time, and flexibility since it supports different objective functions and performance evaluation metrics. The main limitations of LGBM are: it requires hyperparameters tuning, it's prone to overfitting in case of choosing bad values for hyperparameters, difficult to interpret compared to individual decision tree model, and the feature importance estimator generated by LGBM is based on the number of times a feature is used by the decision tree model.

Figure 3.6 shows the LGBM model growth step by step.

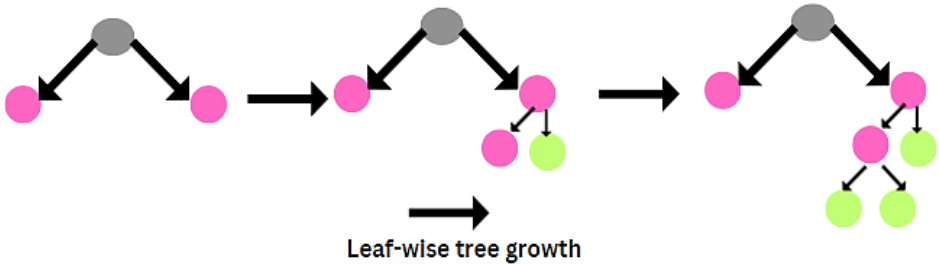


Figure 3.6. LGBM growth.

3.5. FEATURE SELECTION

Feature selection is a part of any pattern recognition task. It performs a selection on the entire features (columns or predictors) of the dataset to get the most beneficial features and drop the redundant ones.

Feature selection is optional but essential in the case of a huge dimensional dataset (dataset with large number of features), since it minimizes the training time, may improve the performance, and even improves the interpretability of the model by simplifying it.

Some ML models get a little benefit in performance by removing the redundant features, while other models' performance can be increased exponentially. However, at least, the feature selection methods minimize the training time of the ML model, especially for those with high computational time like XGBoost and AdaBoost.

In this study, a new hybrid feature selection algorithm is proposed by combining both wrapper and filter-based feature selection methodologies. The algorithm is titled "Hybrid Feature Selection Based ANOVA and RF.

Algorithm 1: Hybrid Feature Selection Based ANOVA and RF (HFSBAR)

Input:

- X: a feature matrix of shape (n_samples, n_features)
- y: a target variable of length n_samples
- k: the initial number of top features to select, default = 10
- n_estimators: the number of trees for the random forest classifier, default = 100

Output:

- Selected_features: the selected features
- Selected_indices: the indices of the selected features in the original feature matrix

Steps:

1: Initialize k, n_estimators
2: Apply SelectKBest(f_classif, k=k) on X, y -> get X_sel, selected_indices
3: Initialize RandomForestClassifier with n_estimators=n_estimators
4: Calculate cross-validation scores of RandomForestClassifier on X_sel, y -> get scores
5: while scores.mean() < 0.9 AND k < X.shape[1] do
6: Set k = min(k + 10, X.shape[1])
7: Apply SelectKBest(f_classif, k=k) on X, y -> get X_sel, selected_indices
8: Calculate cross-validation scores of RandomForestClassifier on X_sel, y -> get scores
9: end while
10: Set Selected_features = X[:, selected_indices]
11: return Selected_features, Selected_indices

This algorithm is a hybrid approach to feature selection, combining filter methods (ANOVA F-value) and wrapper methods (Random Forest Classifier performance) to choose the best features from a given dataset.

1. Function Definition: The function, hybrid_feature_selection, takes four parameters:
   - X: the feature matrix
   - y: the target variable
   - k: the initial number of top features to select based on ANOVA F-value, default is 10
   - n_estimators: the number of trees in the random forest classifier, default is 100
2. Initial Feature Selection: It first selects k top features based on ANOVA F-value. This is a filter-based method, where features are selected based on their relationship with the dependent variable. This is done using the SelectKBest class with f_classif method from sklearn, which computes the ANOVA F-value

between each feature and the target variable. The top k features with the highest F-values are selected.

3. Performance Evaluation: It then evaluates the performance of the selected features using a Random Forest Classifier (a machine learning model) with n_estimators trees. The performance is evaluated using 5-fold cross-validation.
4. Iterative Selection: If the mean cross-validation score is less than 0.9 (indicating that the model performance is not satisfactory) and k is less than the total number of features, it increments k by 10 (or up to the total number of features, whichever is smaller), selects the top k features again, and re-evaluates the performance. This process is repeated until the performance is satisfactory (i.e., the mean cross-validation score is at least 0.9) or all features are selected.
5. Return Values: Finally, it returns the selected features and their indices in the original feature matrix.

PART 4

RESULTS AND DISCUSSIONS

This chapter will contain the experimental part of the current study, including the training scenarios and their corresponding results. The experiments will be applied to the dataset without and with feature selection. For the feature selection, three different scenarios will be applied (5 selected features, 15 selected features, and 20 selected features).

4.1. DATASET PREPROCESSING RESULTS

The used ransomware dataset consists of 84 predictors (features) so it's considered a high-dimensional dataset and needs a feature selection step to get the best features and drop the redundant ones. Table 4.1 includes detailed information on the used dataset's features.

Table 4.1. Description of the used ransomware dataset.

Column	Non-Null Count	Dtype
Flow ID	392034	object
Source IP	392034	object
Source Port	392034	int64
Destination IP	392034	object
Destination Port	392034	int64
Protocol	392034	int64
Timestamp	392034	object
Flow Duration	392034	int64
Total Fwd Packets	392034	int64
Total Backward Packets	392034	int64
Total Length of Fwd Packets	392034	float64
Total Length of Bwd Packets	392034	float64
Fwd Packet Length Max	392034	float64
Fwd Packet Length Min	392034	float64

Column	Non-Null Count	Dtype
Fwd Packet Length Mean	392034	float64
Fwd Packet Length Std	392034	float64
Bwd Packet Length Max	392034	float64
Bwd Packet Length Min	392034	float64
Bwd Packet Length Mean	392034	float64
Bwd Packet Length Std	392034	float64
Flow Bytes/s	392034	float64
Flow Packets/s	392034	float64
Flow IAT Mean	392034	float64
Flow IAT Std	392034	float64
Flow IAT Max	392034	float64
Flow IAT Min	392034	float64
Fwd IAT Total	392034	float64
Fwd IAT Mean	392034	float64
Fwd IAT Std	392034	float64
Fwd IAT Max	392034	float64
Fwd IAT Min	392034	float64
Bwd IAT Total	392034	float64
Bwd IAT Mean	392034	float64
Bwd IAT Std	392034	float64
Bwd IAT Max	392034	float64
Bwd IAT Min	392034	float64
Fwd PSH Flags	392034	int64
Bwd PSH Flags	392034	int64
Fwd URG Flags	392034	int64
Bwd URG Flags	392034	int64
Fwd Header Length	392034	int64
Bwd Header Length	392034	int64
Fwd Packets/s	392034	float64
Bwd Packets/s	392034	float64
Min Packet Length	392034	float64
Max Packet Length	392034	float64
Packet Length Mean	392034	float64
Packet Length Std	392034	float64
Packet Length Variance	392034	float64
FIN Flag Count	392034	int64
SYN Flag Count	392034	int64
RST Flag Count	392034	int64
PSH Flag Count	392034	int64
ACK Flag Count	392034	int64

Column	Non-Null Count	Dtype
URG Flag Count	392034	int64
CWE Flag Count	392034	int64
ECE Flag Count	392034	int64
Down/Up Ratio	392034	float64
Average Packet Size	392034	float64
Avg Fwd Segment Size	392034	float64
Avg Bwd Segment Size	392034	float64
Fwd Header Length.1	392034	int64
Fwd Avg Bytes/Bulk	392034	int64
Fwd Avg Packets/Bulk	392034	int64
Fwd Avg Bulk Rate	392034	int64
Bwd Avg Bytes/Bulk	392034	int64
Bwd Avg Packets/Bulk	392034	int64
Bwd Avg Bulk Rate	392034	int64
Subflow Fwd Packets	392034	int64
Subflow Fwd Bytes	392034	int64
Subflow Bwd Packets	392034	int64
Subflow Bwd Bytes	392034	int64
Init_Win_bytes_forward	392034	int64
Init_Win_bytes_backward	392034	int64
act_data_pkt_fwd	392034	int64
min_seg_size_forward	392034	int64
Active Mean	392034	float64
Active Std	392034	float64
Active Max	392034	float64
Active Min	392034	float64
Idle Mean	392034	float64
Idle Std	392034	float64
Idle Max	392034	float64
Idle Min	392034	float64

After applying the encoding, the textual columns will be transformed into a numerical form as Figure 4.1.

	Flow ID	Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration	Total Fwd Packets	Total Backward Packets	...	min_seg_size_forward	Active Mean	Active Std	...	Label
0	172.217.2.174-10.42.0.211-443-51023-6	10.42.0.211	51023	172.217.2.174	443	6	16/06/2017 03:55:47	151054	6	8	...	32	0.0	0.0	...	Benign
1	172.217.2.174-10.42.0.211-443-51023-6	10.42.0.211	51023	172.217.2.174	443	6	16/06/2017 03:55:47	349	2	0	...	32	0.0	0.0	...	Benign
2	172.217.12.174-10.42.0.211-443-34259-6	10.42.0.211	34259	172.217.12.174	443	6	16/06/2017 03:55:52	119	2	0	...	32	0.0	0.0	...	Benign
3	172.217.10.74-10.42.0.211-443-55509-6	10.42.0.211	55509	172.217.10.74	443	6	16/06/2017 03:55:53	37055	1	1	...	32	0.0	0.0	...	Benign
4	172.217.2.174-10.42.0.211-443-44852-6	10.42.0.211	44852	172.217.2.174	443	6	16/06/2017 03:55:58	178727	6	7	...	32	0.0	0.0	...	Benign

(a)

	Flow ID	Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration	Total Fwd Packets	Total Backward Packets	...	min_seg_size_forward	Active Mean	Active Std	...	Label
0	151298	14	51023	1328	443	6	4	151054	6	8	...	32	0.0	0.0	...	0
1	151298	14	51023	1328	443	6	4	349	2	0	...	32	0.0	0.0	...	0
2	148227	14	34259	1294	443	6	7	119	2	0	...	32	0.0	0.0	...	0
3	142970	14	55509	1247	443	6	8	37055	1	1	...	32	0.0	0.0	...	0
4	151270	14	44852	1328	443	6	9	178727	6	7	...	32	0.0	0.0	...	0

(b)

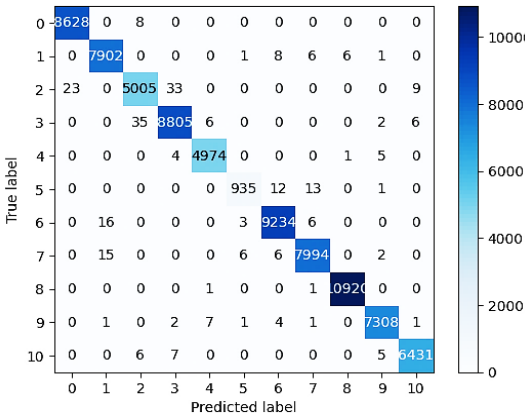
Figure 4.1. Part of the dataset (a) before and (b) after encoding.

After that, the normalization process is applied to ensure that all column values have standard values and no one can bias the learning process.

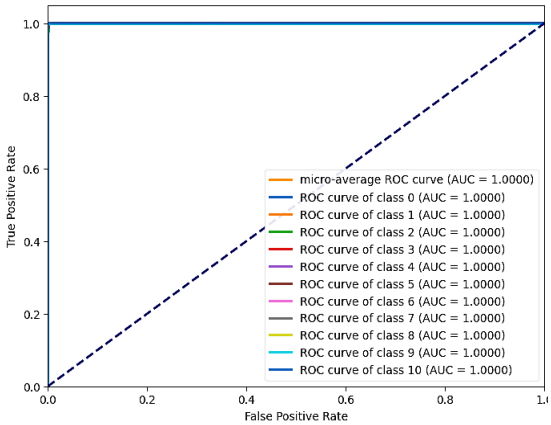
In the next step, the dataset is split into train and test sets so that the training size is 313627 samples, while the test size is 78407 samples. Now the training and test sets are ready for the training step.

4.3. EVALUATION RESULTS OF ML MODELS WITHOUT FEATURE SELECTION

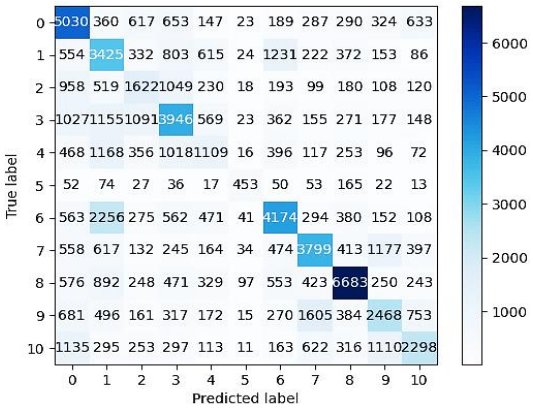
In this scenario, the ML models will be trained using the entire dataset without any feature selection. Figure 4.2 shows the confusion matrixes and ROC plots with AUC values of the individual ML and DL models achieved by evaluating trained models using the test set. In the ROC plot, the AUC of each class of the target column (ransomware type) is also provided.



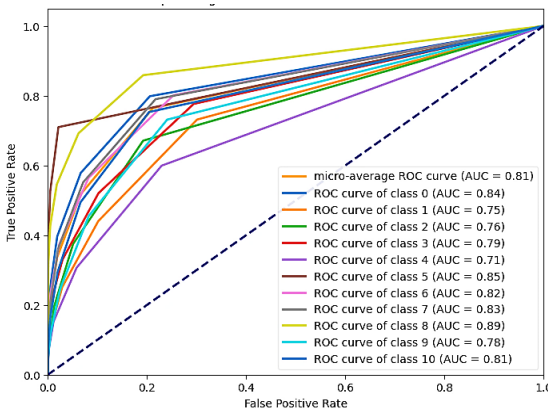
LGBM CM



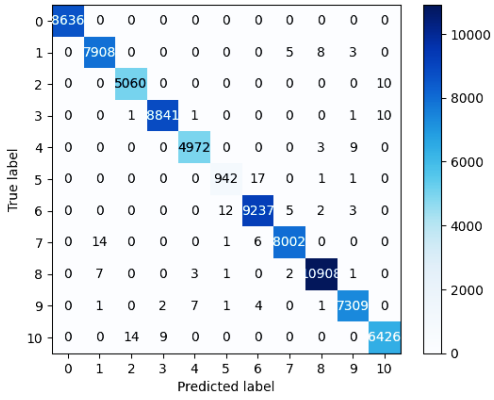
LGBM ROC



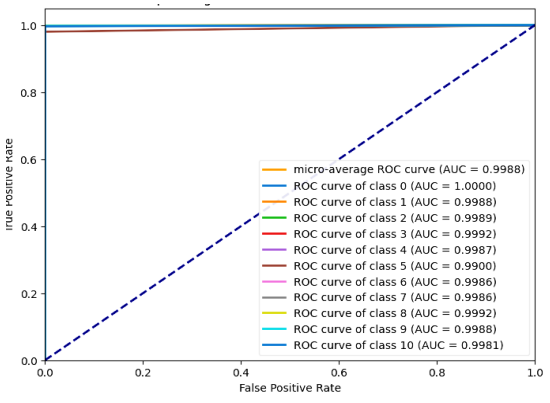
K-NN CM



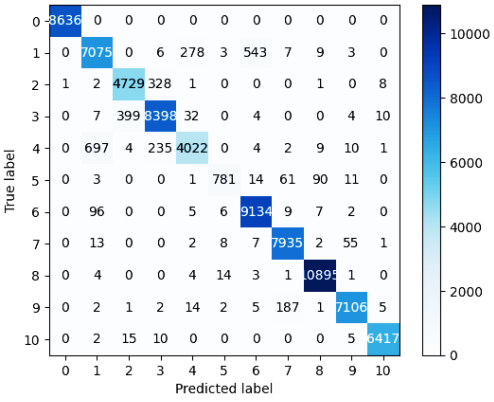
K-NN ROC



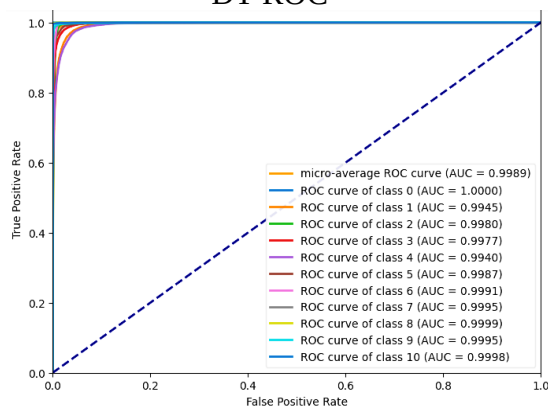
DT CM



DT ROC



RF CM



RF ROC

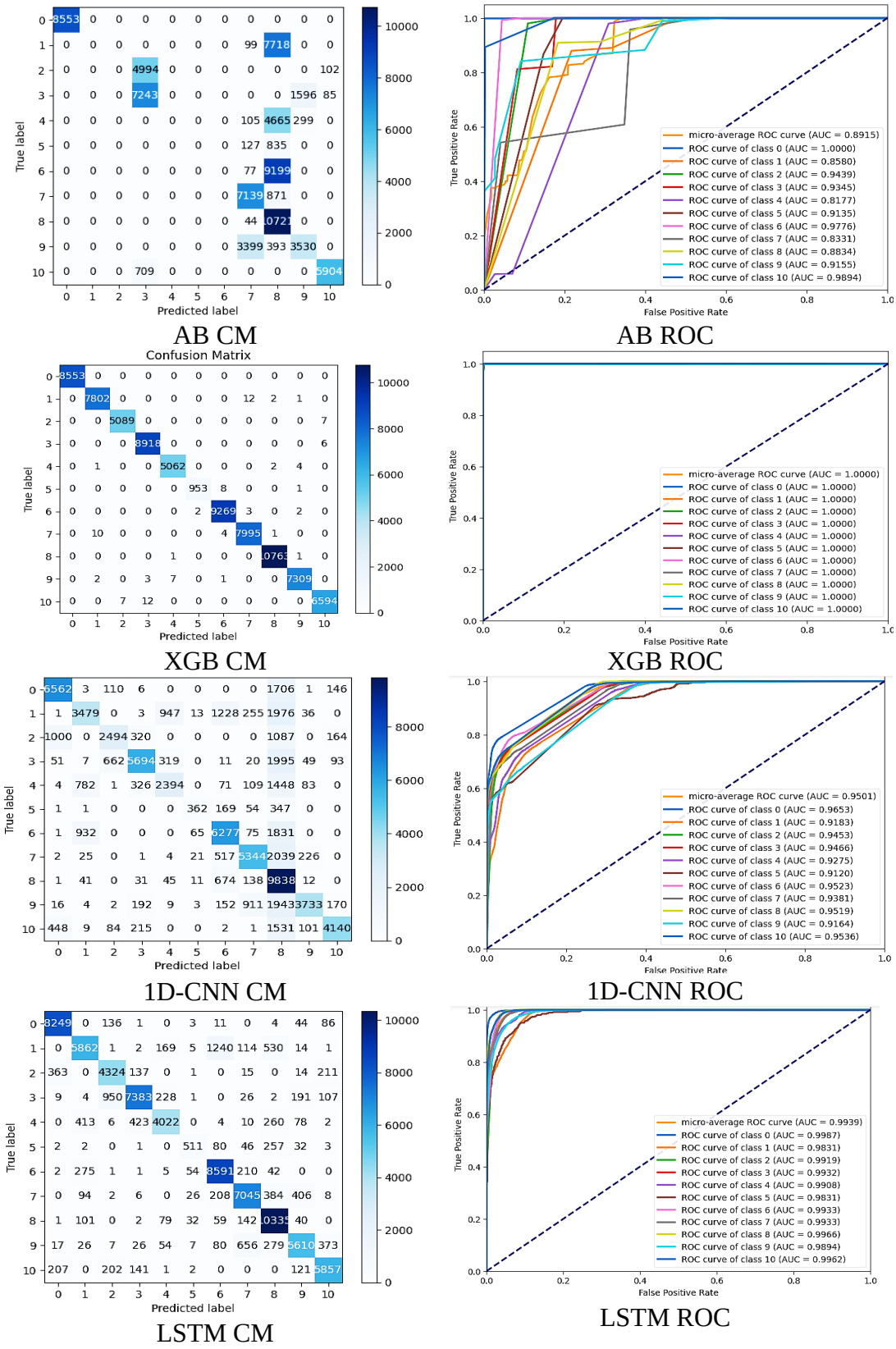
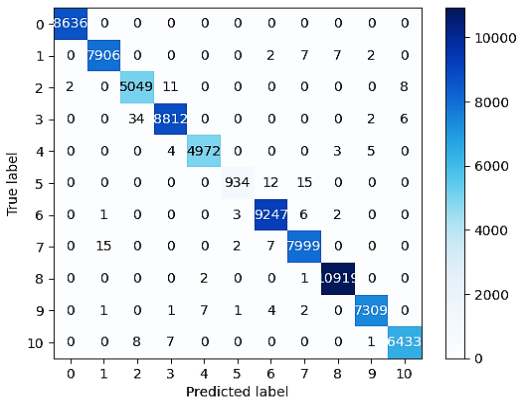


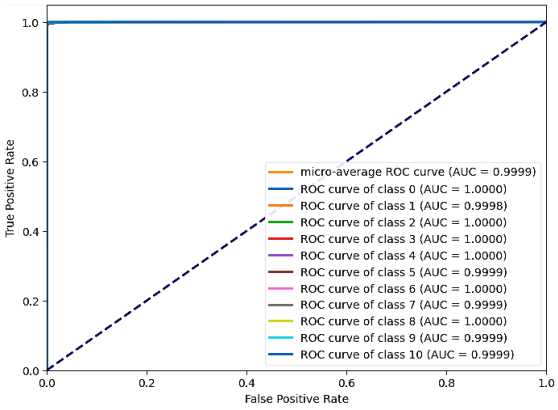
Figure 4.2. Confusion matrixes and ROC plots with AUC values of the individual ML and DL models achieved by evaluating trained models

Figure 4.2. shows that the best individual model is the XGB model, and the second best model is the LGBM model in terms of confusion matrix and ROC. The registered AUC of the XGB model is almost 1 (100%) in all ransomware types. The worst models are the AdaBoost and K-NN models, which is normal since the K-NN model doesn't fit the large datasets, and the AdaBoost model needs a hyperparameters tuning step to get a good accuracy. Figure (4-2) shows that DL models achieved a lower performance than the ML models. However, the LSTM model outperforms the 1D-CNN, K-NN, and AdaBoost models. For the DL models, a percentage of 20% of the training set is considered a validation set for the training process.

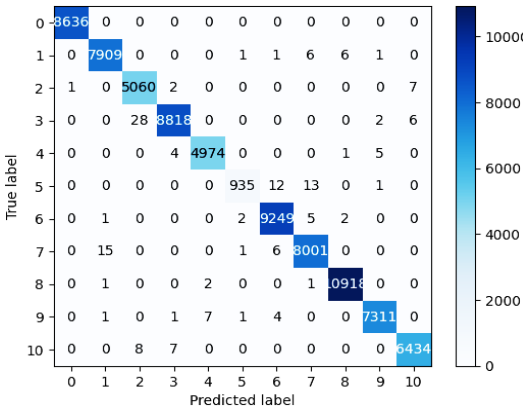
Figure 4.3. includes the confusion matrixes and ROC plots for the fusion models and ensemble models.



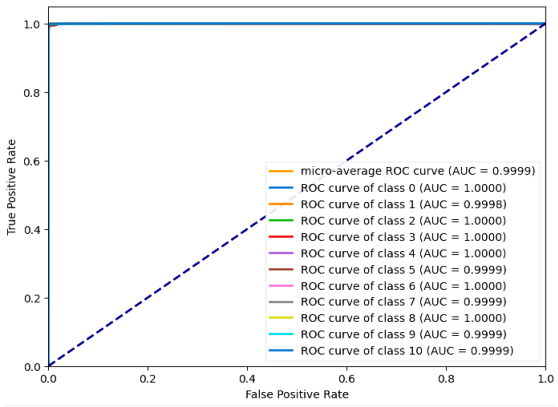
ML-based Fusion CM



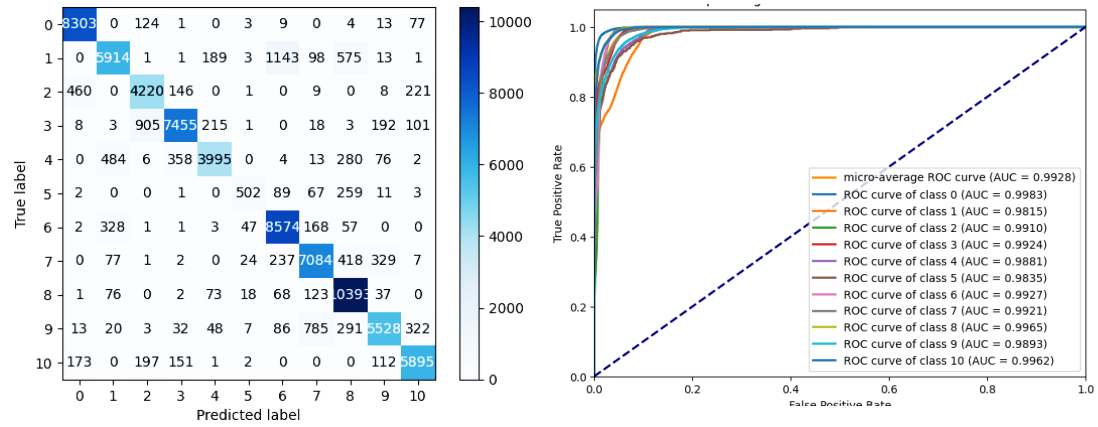
ML-based Fusion ROC



ML-based Ensemble CM

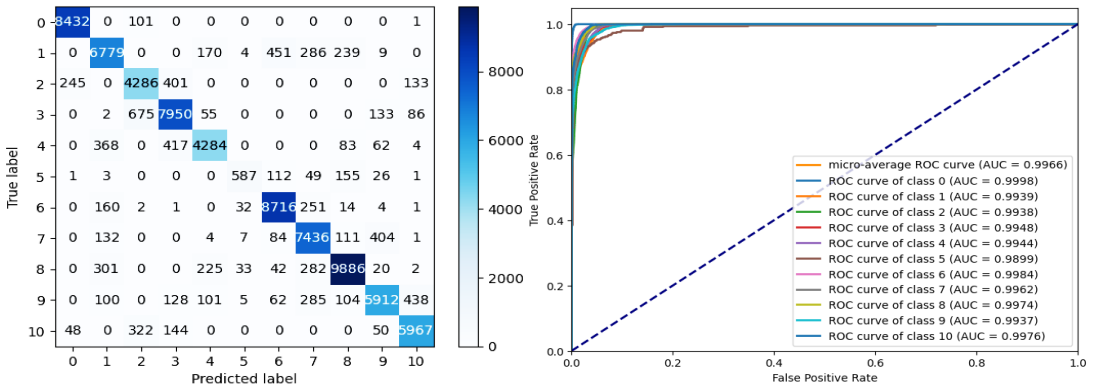


ML-based Ensemble ROC



DL-based Fusion CM

DL-based Fusion ROC

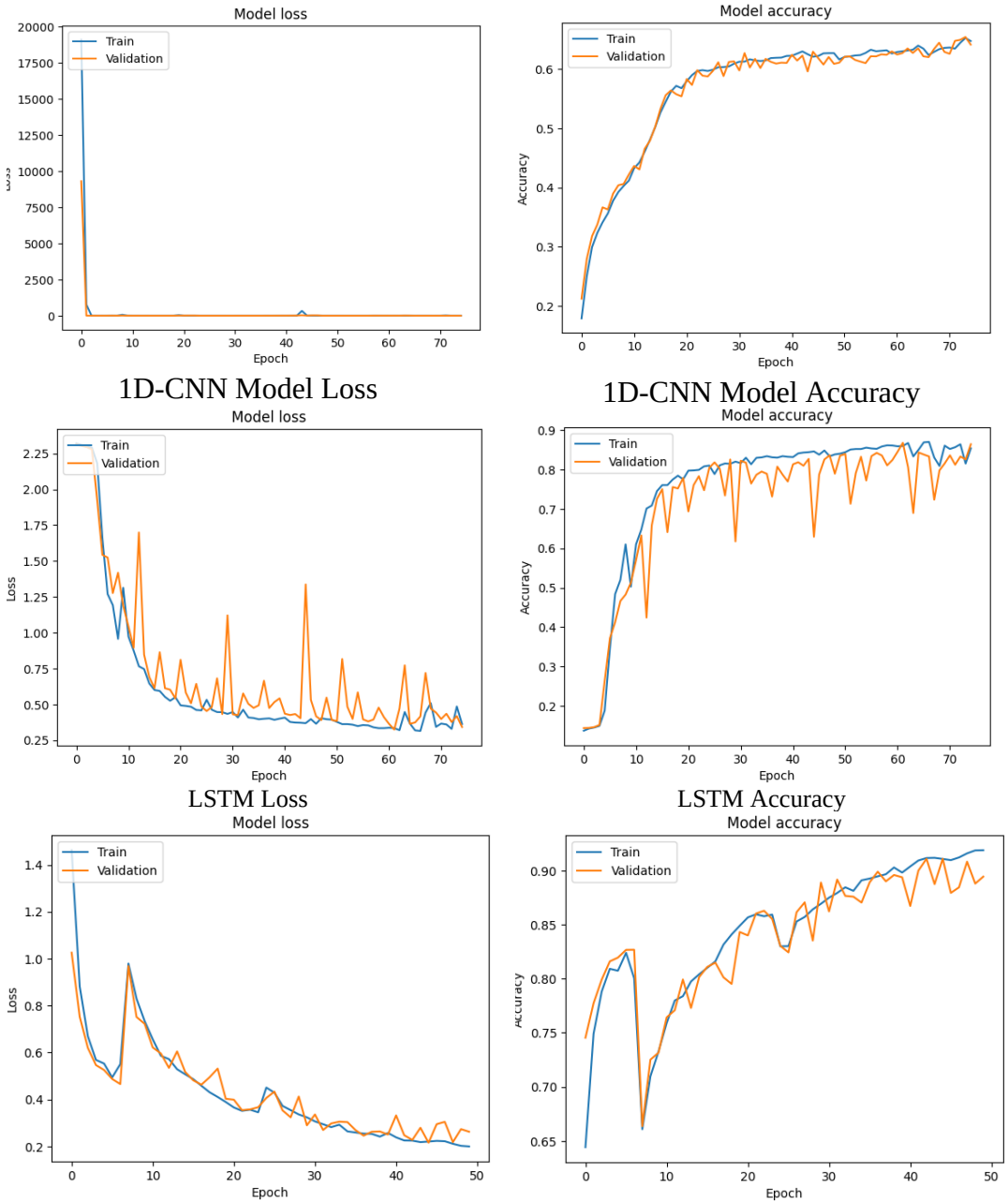


DL-based Ensemble CM

DL-based Ensemble ROC

Figure 4.3. Confusion matrixes and ROC plots with AUC values of the fused/ensemble ML and DL models achieved by evaluating trained models.

Figure 4.3 shows that the ML-based fusion model has a better performance than the DL-based model. The best ROC score corresponds to the ML-Ensemble model with almost 1 (100%). Figure 4.4. includes the training and validation accuracy and loss of the individual and fused DL models.



1D-CNN Model Loss

1D-CNN Model Accuracy

LSTM Loss

LSTM Accuracy

DL Ensemble Loss

DL Ensemble Accuracy

Figure 4.4. Loss and accuracy of the individual and fused DL models.

The precision, recall, F1-score, and accuracy scores are also computed for all individual, fused, and ensemble models to define the best model. Table (4-2) illustrates these metrics for individual models.

Table 4.2. Individual models accuracy, precision, recall, and F1-score metrics.

Model	**LGBM**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	100	100	100	8636
Charger	100	100	100	7924
Jisut	99	99	99	5070
Koler	99	99	99	8854
Lockerpin	100	100	100	4984
Pletor	99	97	98	961
PornDroid	100	100	100	9259
RansomBO	100	100	100	8023
SVpeng	100	100	100	10922
Simplocker	100	100	100	7325
WannaLocker	100	100	100	6449
Average Value	100	99	99	78407
Accuracy %	99.65			78407
Model	**K-NN**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	43	59	50	8636
Charger	30	44	36	7924
Jisut	32	32	32	5070
Koler	42	44	43	8854
Lockerpin	28	22	25	4984
Pletor	60	47	53	961
PornDroid	52	45	48	9259
RansomBO	49	47	48	8023
SVpeng	69	62	65	10922
Simplocker	41	34	37	7325
WannaLocker	47	35	40	6449
Average Value	45	43	43	78407
Accuracy %	45			78407
Model	**DT**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	100	100	100	8636
Charger	99.72	99.98	99.76	7924
Jisut	99.7	99.8	99.75	5070
Koler	99.88	99.85	99.86	8854
Lockerpin	99.78	99.76	99.77	4984
Pletor	99.43	98.02	98.23	961
PornDroid	99.71	99.76	99.74	9259
RansomBO	99.85	99.74	99.79	8023
SVpeng	99.86	99.87	99.87	10922
Simplocker	99.75	99.78	99.77	7325
WannaLocker	99.69	99.64	99.67	6449
Average Value	99.67	99.64	99.66	78407
Accuracy %	99.79			78407
Model	**RF**			

Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	99.99	100	99.99	8636
Charger	89.55	89.29	89.42	7924
Jisut	91.86	93.27	92.56	5070
Koler	93.53	94.85	94.18	8854
Lockerpin	92.27	80.7	86.1	4984
Pletor	95.95	81.27	88	961
PornDroid	94.03	98.65	96.28	9259
RansomBO	96.74	98.9	97.8	8023
SVpeng	98.92	99.75	99.33	10922
Simplocker	98.74	97.01	97.87	7325
WannaLocker	99.61	99.5	99.56	6449
Average Value	95.56	93.93	94.65	78407
Accuracy %		95.82		78407

Model	**AdaBoost**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	100	100	100	8636
Charger	0	0	0	7924
Jisut	0	0	0	5070
Koler	55.95	81.16	66.24	8854
Lockerpin	0	0	0	4984
Pletor	0	0	0	961
PornDroid	0	0	0	9259
RansomBO	64.96	89.13	75.15	8023
SVpeng	31.16	99.59	47.47	10922
Simplocker	65.07	48.21	55.39	7325
WannaLocker	96.93	89.28	92.95	6449
Average Value	37.64	46.12	39.74	78407
Accuracy %		54.96		78407
Model	**XGB**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	100	100	100	8636
Charger	99.83	99.81	99.82	7924
Jisut	99.86	99.86	99.86	5070
Koler	99.83	99.93	99.88	8854
Lockerpin	99.84	99.86	99.85	4984
Pletor	99.79	99.06	99.43	961
PornDroid	99.86	99.92	99.89	9259
RansomBO	99.81	99.81	99.81	8023
SVpeng	99.95	99.98	99.97	10922
Simplocker	99.88	99.82	99.85	7325
WannaLocker	99.8	99.81	99.76	6449
Average Value	99.86	99.8	99.83	78407
Accuracy %		99.87		78407
Model	**1D-CNN**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***

Benign	81.14	76.89	78.96	8636
Charger	65.85	43.83	52.63	7924
Jisut	74.38	49.24	59.25	5070
Koler	83.88	63.97	72.59	8854
Lockerpin	64.39	45.88	53.58	4984
Pletor	76.21	38.76	51.38	961
PornDroid	68.97	68.37	68.67	9259
RansomBO	77.37	65.34	70.85	8023
SVpeng	38.22	91.17	53.86	10922
Simplocker	88.02	52.32	65.63	7325
WannaLocker	87.84	63.39	73.64	6449
Average Value	73.3	59.92	63.73	78407
Accuracy %		64.17		78407
Model	**LSTM**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	93.21	96.66	94.9	8636
Charger	86.5	73.85	79.67	7924
Jisut	76.82	85.37	80.87	5070
Koler	90.89	82.95	86.74	8854
Lockerpin	88.24	77.08	82.28	4984
Pletor	79.6	54.71	64.85	961
PornDroid	83.63	93.57	88.32	9259
RansomBO	85.25	86.14	85.69	8023
SVpeng	85.46	95.77	90.33	10922
Simplocker	85.65	78.63	81.99	7325
WannaLocker	88.1	89.68	88.88	6449
Average Value	85.76	83.13	84.05	78407
Accuracy %		86.46		78407

Table 4.2 proves the same conclusion as Figure (4-3) since the precision, recall, and F1-score of the XGB and LGBM models are the best values. Precision, recall, F1-score, and accuracy of the LGBM model are 100%, 99%, 99%, and 99.65%, respectively. The same metrics of the XGB model are 99.86%, 99.8%, 99.83%, and 99.87%, respectively.

Table 4.3 shows the results of the fused/ensemble ML and DL models, and it's obvious that the ensemble ML model is the best one with precision, recall, F1-score, and accuracy values of 99.7%, 99.59%, 99.67%, and 99.79%, respectively. As seen in Table 4.3, the ML-based fusion or ML-based Ensemble has no enhancement compared to the individual models. In contrast, the DL-based fusion and ensemble models improved the performance of the individual DL model significantly. The DL ensemble model accuracy increased by 3.12% compared to the best individual DL model (LSTM).

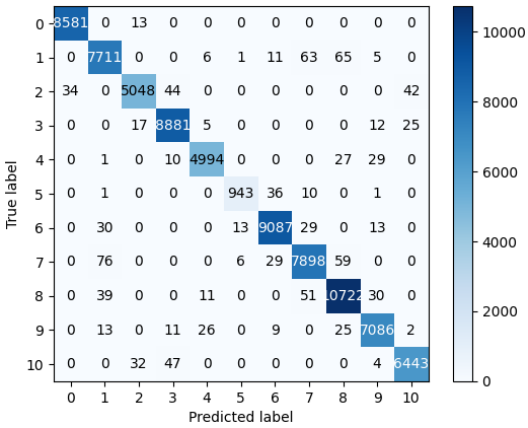
Table 4.3. Fused/Ensemble models accuracy, precision, recall, and F1-score metrics.

Model			**ML-Based Fusion**	
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	99.98	100	99.99	8636
Charger	99.79	99.77	99.78	7924
Jisut	99.18	99.59	99.38	5070
Koler	99.74	99.53	99.63	8854
Lockerpin	99.82	99.76	99.79	4984
Pletor	99.36	97.19	98.26	961
PornDroid	99.73	99.87	99.8	9259
RansomBO	99.61	99.7	999.66	8023
SVpeng	99.89	99.97	99.93	10922
Simplocker	99.86	99.78	99.82	7325
WannaLocker	99.78	99.75	99.77	6449
Average Value	99.7	99.54	99.62	78407
Accuracy %		99.76		78407
Model			**ML-Based Ensemble**	
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	99.99	100	99.99	8636
Charger	99.77	99.81	99.79	7924
Jisut	99.29	99.8	99.55	5070
Koler	99.84	99.59	99.72	8854
Lockerpin	99.82	99.8	99.81	4984
Pletor	99.47	97.29	98.37	961
PornDroid	99.75	99.89	99.82	9259
RansomBO	99.69	99.73	99.71	8023
SVpeng	99.92	99.96	99.94	10922
Simplocker	99.88	99.81	99.84	7325
WannaLocker	99.8	99.77	99.79	6449
Average Value	99.7	99.59	99.67	78407
Accuracy %		99.79		78407
Model			**DL-Based Fusion**	
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	92.65	97.29	94.91	8636
Charger	85.69	74.5	79.7	7924
Jisut	77.32	83.32	80.21	5070
Koler	91.47	83.75	87.44	8854
Lockerpin	88.31	76.56	82.02	4984
Pletor	82.57	53.75	65.11	961
PornDroid	83.98	93.39	88.43	9259
RansomBO	84.69	86.61	85.64	8023
SVpeng	84.63	96.31	90.1	10922
Simplocker	87.48	77.48	82.18	7325
WannaLocker	88.93	90.26	89.59	6449
Average Value	86.15	83.02	84.12	78407
Accuracy %		86.55		78407
Model			**DL-Based Ensemble**	

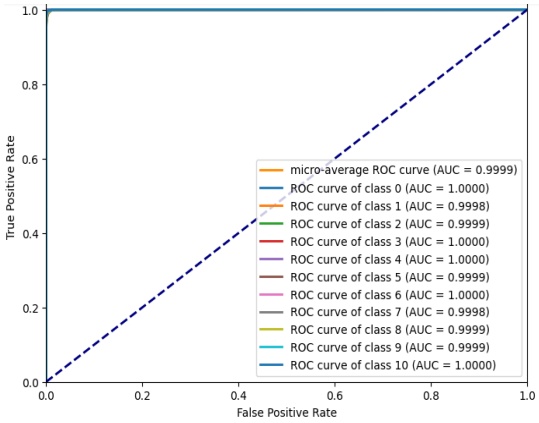
Metrics	*Precision %*	*Recall %*	*F1-score %*	*No. Test samples*
Benign	96.63	98.8	97.71	8636
Charger	86.41	85.4	85.9	7924
Jisut	79.58	84.62	82.02	5070
Koler	87.93	89.32	88.62	8854
Lockerpin	88.53	82.1	85.19	4984
Pletor	87.87	62.85	73.28	961
PornDroid	92.07	94.94	88.69	9259
RansomBO	86.58	90.92	88.69	8023
SVpeng	93.33	91.61	92.47	10922
Simplocker	89.31	82.86	85.69	7325
WannaLocker	89.95	91.36	90.65	6449
Average Value	88.93	86.8	87.63	78407
Accuracy %		89.58		78407

4.4. EVALUATION RESULTS OF ML MODELS WITH FEATURE SELECTION

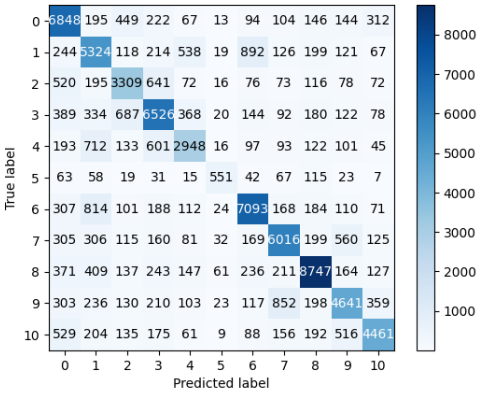
In this scenario, the ML models will be trained using the selected features of the original ransomware dataset. For this case, three different scenarios are performed (20 selected features, 10 selected features, and 5 selected features). Figure 4.5 shows the confusion matrixes and ROC plots with AUC values of the individual ML and DL models achieved by evaluating trained models using the test set (in the case of feature selection with only 20 features out of 84 features).



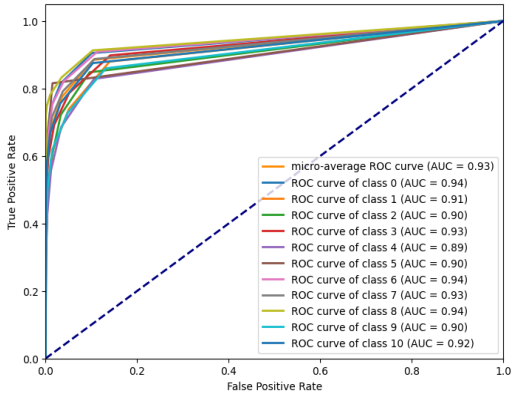
LGBM CM



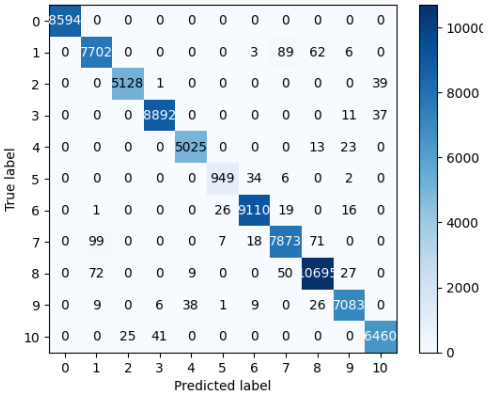
LGBM ROC



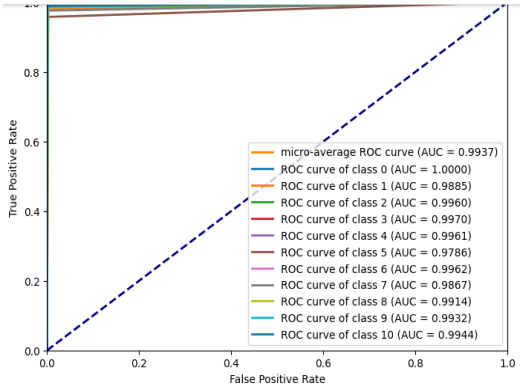
K-NN CM



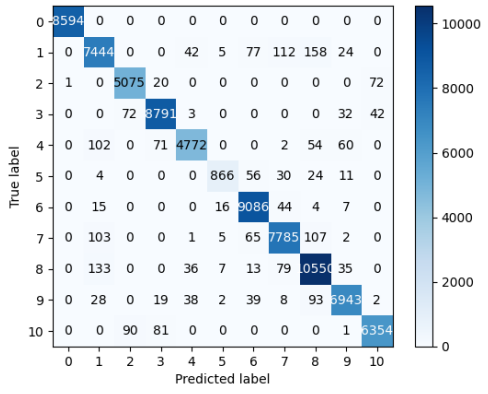
K-NN ROC



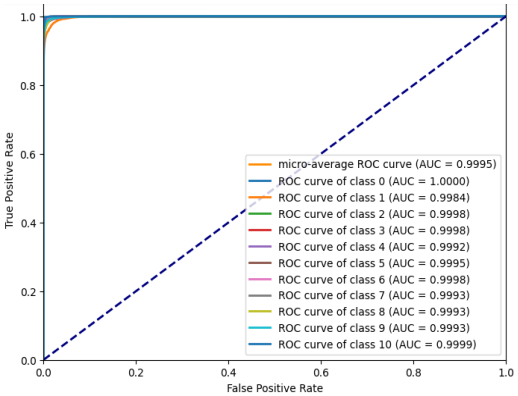
DT CM



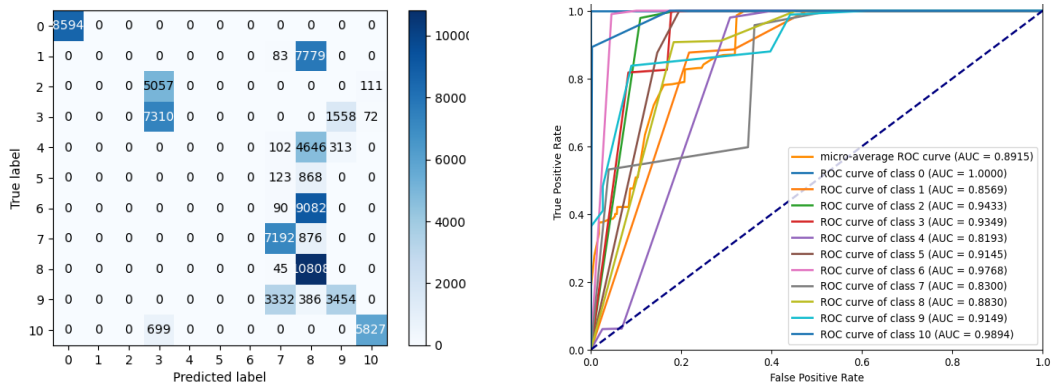
DT ROC



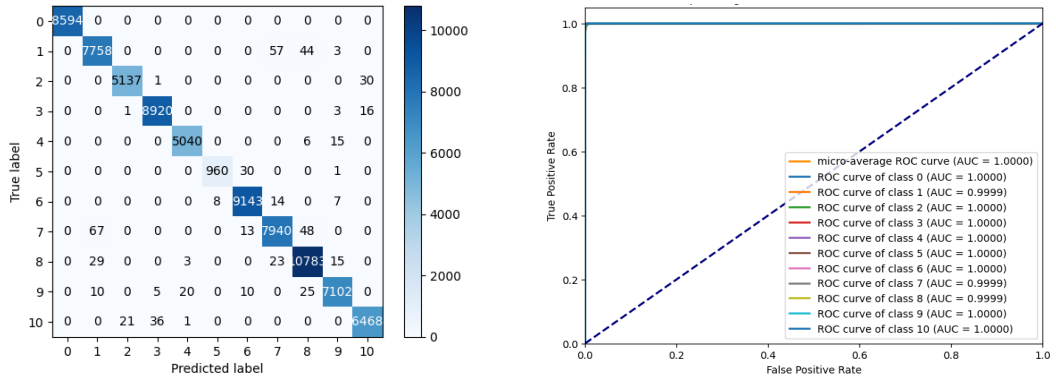
RF CM



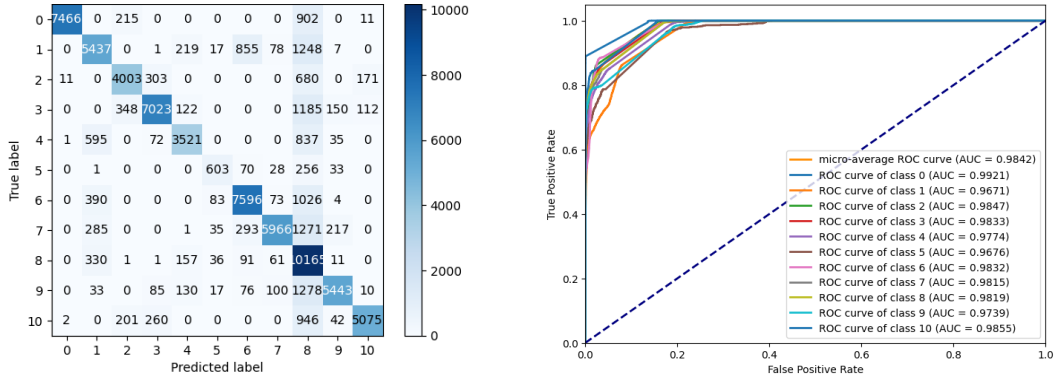
RF ROC



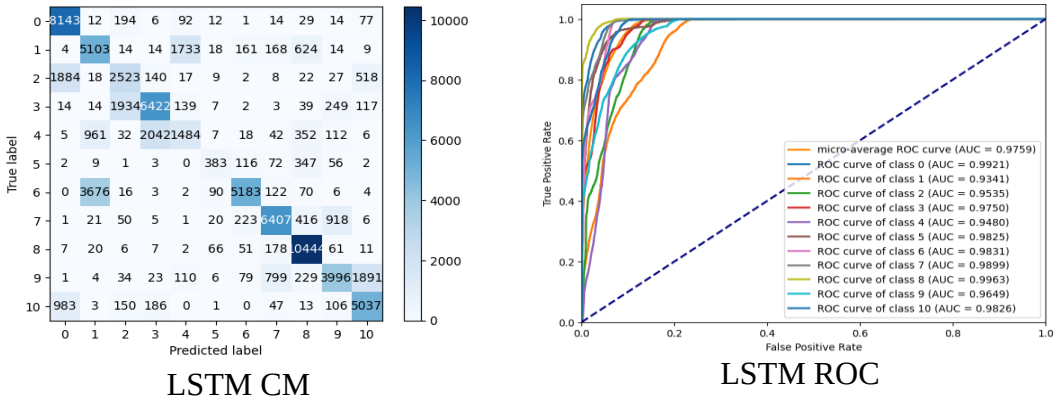
AB CM AB ROC



XGB CM XGB ROC



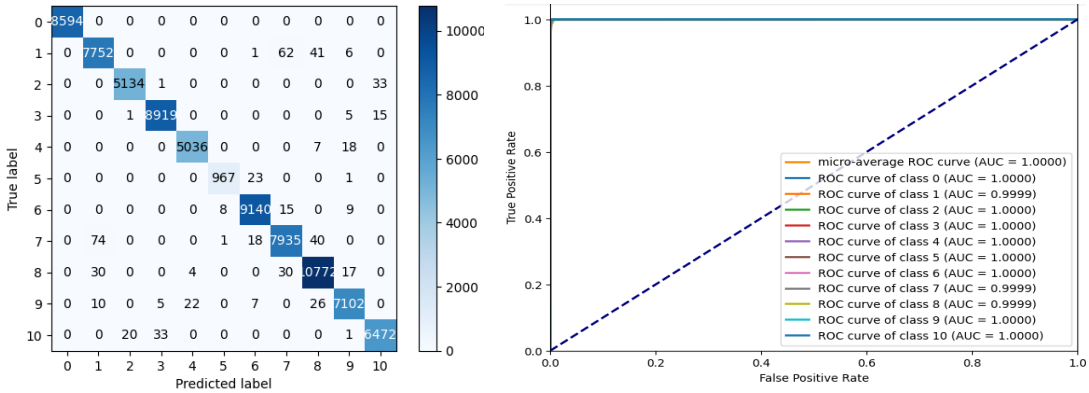
1D-CNN CM 1D-CNN ROC



LSTM CM LSTM ROC

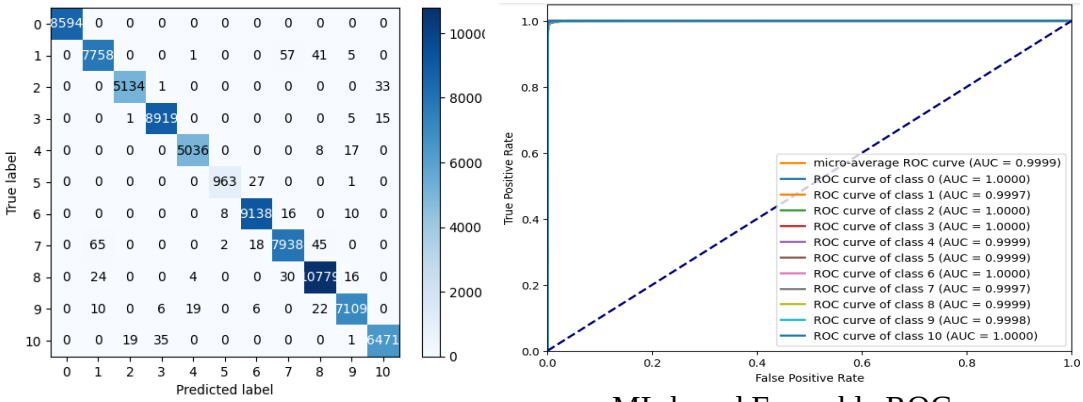
Figure 4.5. Confusion matrixes and ROC plots with AUC values of the individual ML and DL models achieved by evaluating trained models (with feature selection, Number of features=20).

Figure 4.5 shows that the best individual models are the XGB and the LGBM models in terms of confusion matrix and ROC. The registered AUCs of the XGB and LGBM models are almost 1 (99.99%-100%) in all ransomware types. The worst models are the AdaBoost and K-NN models, which is normal since the K-NN model doesn't fit the large datasets, and the AdaBoost model needs a hyperparameters tuning step to get a good accuracy. Figure 4.5 shows that DL models achieved a lower performance than the ML models. However, the LSTM model outperforms the 1D-CNN, K-NN, and AdaBoost models. Figure 4.6 includes the confusion matrixes and ROC plots for the fusion models and ensemble models.



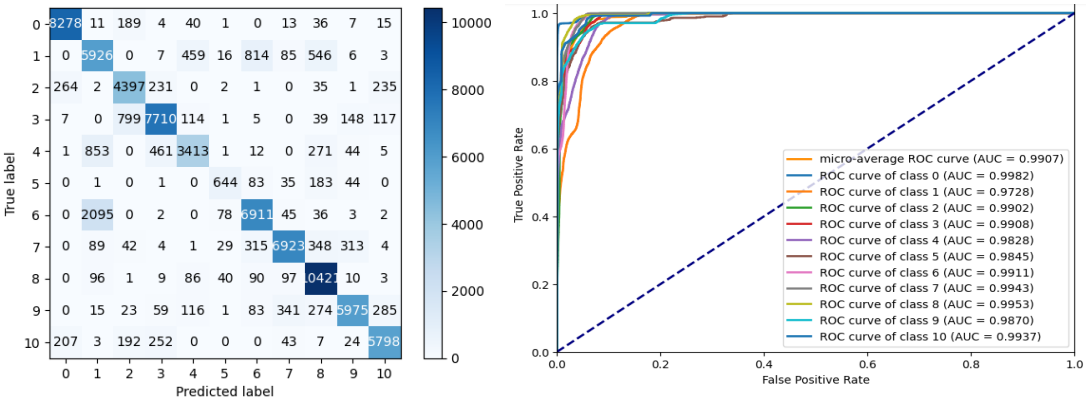
ML-based Fusion CM

ML-based Fusion ROC



ML-based Ensemble CM

ML-based Ensemble ROC



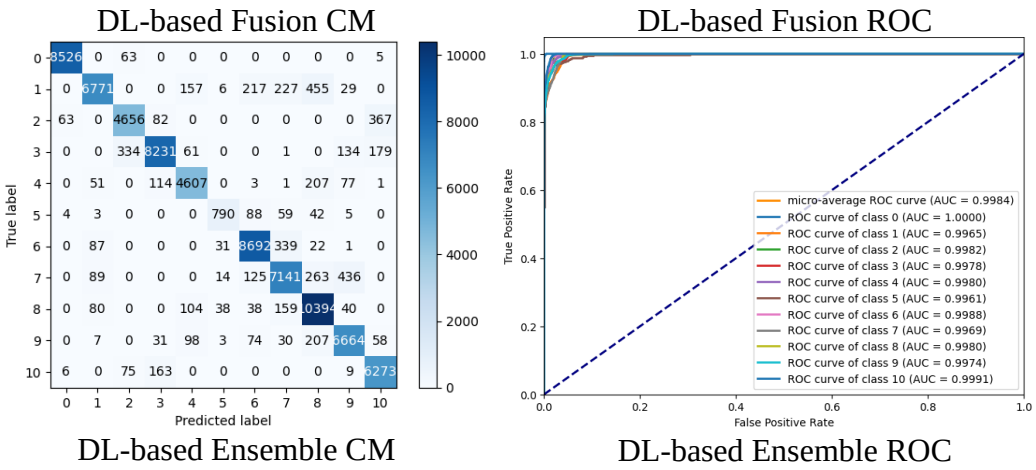


Figure 4.6. Confusion matrixes and ROC plots with AUC values of the fused/ensemble ML and DL models achieved by evaluating trained models (with feature selection, Number of features=20).

Figure 4.6 shows that the ML-based fusion model has a better performance than the DL-Based model. The best ROC score corresponds to the ML-Ensemble model with almost 1 (100%). In this scenario, the ensemble and fusion models improved the performance of the individual ML and DL models.

Figure 4.7. includes the training and validation accuracy and loss of the individual and fused DL models.

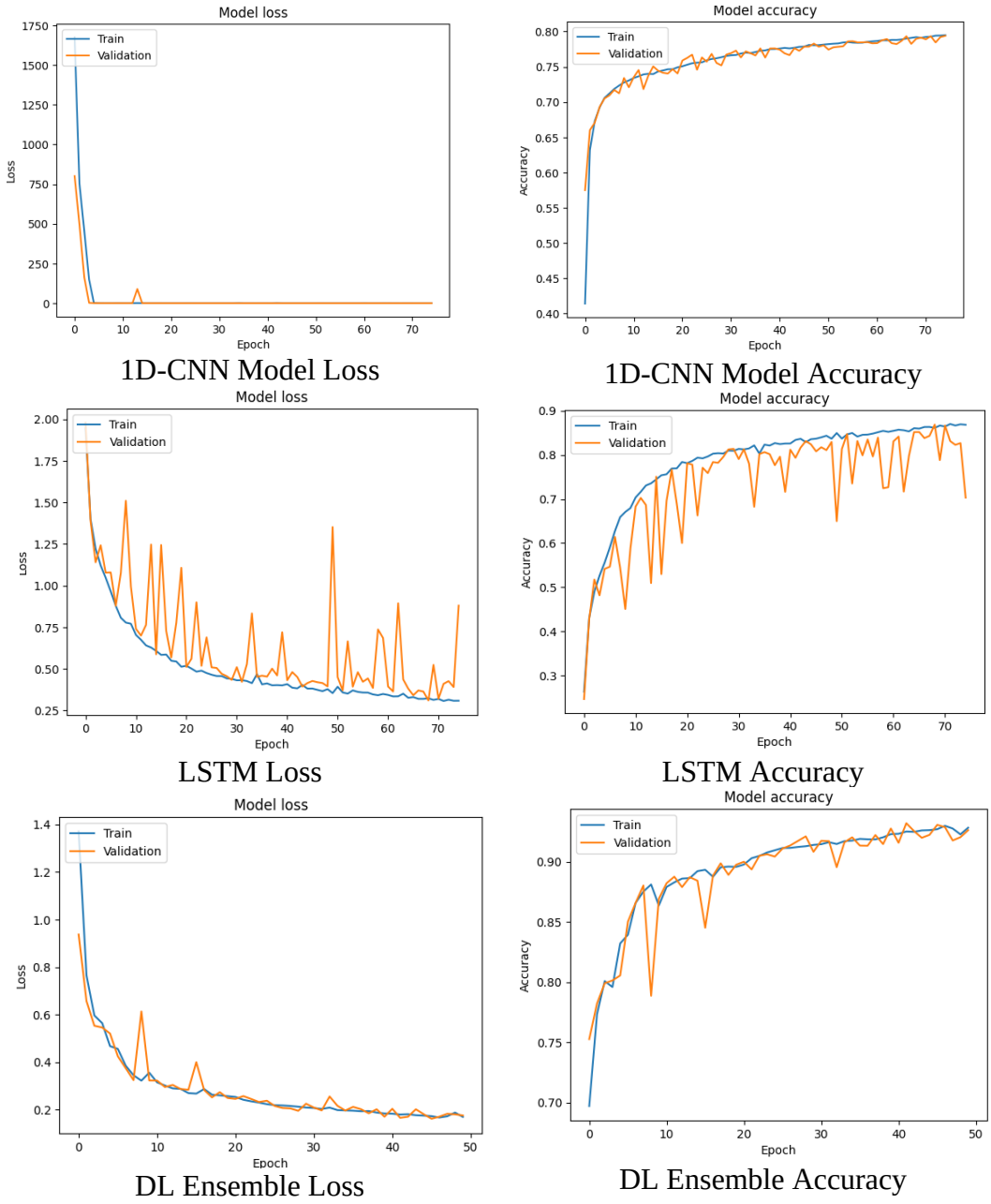


Figure 4.7. Loss and accuracy of the individual and fused DL models (with feature selection, number of features=20).

The precision, recall, F1-score, and accuracy scores are also computed for all individual, fused, and ensemble models to define the best model. Table 4.4 illustrates these metrics for individual models in the case of feature selection (20 selected features).

Table 4.4. Individual models accuracy, precision, recall, and F1-score metrics (with feature selection, number of features=20).

Model			**LGBM**	
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	100	100	100	8636
Charger	98	98	98	7924
Jisut	99	98	98	5070
Koler	99	99	99	8854
Lockerpin	99	99	99	4984
Pletor	98	95	97	961
PornDroid	99	99	99	9259
RansomBO	98	98	98	8023
SVpeng	98	99	99	10922
Simplocker	99	99	99	7325
WannaLocker	99	99	99	6449
Average Value	99	98	99	78407
Accuracy %		98.7		78407
Model			**K-NN**	
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	68	80	73	8636
Charger	61	68	64	7924
Jisut	62	64	63	5070
Koler	71	73	72	8854
Lockerpin	65	58	62	4984
Pletor	70	56	62	961
PornDroid	78	77	78	9259
RansomBO	76	75	75	8023
SVpeng	84	81	82	10922
Simplocker	71	65	67	7325
WannaLocker	78	68	73	6449
Average Value	71	69	70	78407
Accuracy %		72.01%		78407
Model			**DT**	
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	100	100	100	8636
Charger	97.7	97.6	97.83	7924
Jisut	99.51	99.23	99.37	5070
Koler	99.46	99.46	99.46	8854
Lockerpin	99.07	99.29	99.18	4984
Pletor	96.54	95.76	96.5	961
PornDroid	99.3	99.32	99.31	9259
RansomBO	97.96	97.58	97.77	8023
SVpeng	98.42	98.54	98.48	10922
Simplocker	98.81	98.76	98.79	7325
WannaLocker	98.84	98.99	98.91	6449
Average Value	98.69	98.63	98.66	78407
Accuracy %		98.58		78407
Model			**RF**	
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	99.99	100	99.99	8636
Charger	95.08	94.68	94.88	7924

Jisut	96.91	98.2	97.55	5070
Koler	97.87	98.33	98.1	8854
Lockerpin	97.55	94.29	95.89	4984
Pletor	96.12	87.39	91.54	961
PornDroid	97.32	99.06	98.18	9259
RansomBO	96.59	96.49	96.54	8023
SVpeng	96	97.21	96.6	10922
Simplocker	97.58	96.81	97.19	7325
WannaLocker	98.21	97.36	97.78	6449
Average Value	97.2	96.35	96.75	78407
Accuracy %		97.26		78407
Model		**AdaBoost**		
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	100	100	100	8636
Charger	0	0	0	7924
Jisut	0	0	0	5070
Koler	55.95	81.77	66.44	8854
Lockerpin	0	0	0	4984
Pletor	0	0	0	961
PornDroid	0	0	0	9259
RansomBO	65.58	89.14	75.57	8023
SVpeng	31.38	99.59	47.72	10922
Simplocker	64.86	48.16	55.28	7325
WannaLocker	96.96	89.29	92.96	6449
Average Value	37.7	46.18	39.81	78407
Accuracy %		55.077		78407
Model		**XGB**		
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	100	100	100	8636
Charger	98.65	98.68	98.66	7924
Jisut	99.57	99.4	99.49	5070
Koler	99.53	99.78	99.65	8854
Lockerpin	99.53	99.59	99.56	4984
Pletor	99.17	96.87	98.01	961
PornDroid	99.42	99.68	99.55	9259
RansomBO	98.83	98.41	9.62	8023
SVpeng	98.87	99.36	99.11	10922
Simplocker	99.38	99.02	99.2	7325
WannaLocker	99.29	99.11	99.2	6449
Average Value	99.3	99.08	99.19	78407
Accuracy %		99.28		78407

Model	**1D-CNN**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	99.81	86.87	92.9	8636
Charger	76.89	69.16	72.82	7924
Jisut	83.96	77.46	80.58	5070
Koler	90.68	78.56	84.18	8854
Lockerpin	84.84	69.57	76.45	4984

Pletor	76.23	60.85	67.68	961
PornDroid	84.58	82.82	83.6	9259
RansomBO	94.61	73.95	83.01	8023
SVpeng	51.35	93.66	66.34	10922
Simplocker	91.6	75.89	83.01	7325
WannaLocker	94.35	77.77	85.26	6449
Average Value	84.45	76.96	79.63	78407
Accuracy %		79.45		78407
Model		**LSTM**		
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	73.73	94.75	82.93	8636
Charger	51.85	64.91	57.65	7924
Jisut	50.93	48.82	49.85	5070
Koler	72.56	71.83	72.19	8854
Lockerpin	41.45	29.32	34.35	4984
Pletor	61.87	38.65	47.58	961
PornDroid	88.81	56.51	69.07	9259
RansomBO	81.51	79.41	80.45	8023
SVpeng	82.99	96.23	89.12	10922
Simplocker	71.88	55.72	62.78	7325
WannaLocker	65.6	77.18	70.92	6449
Average Value	67.56	64.85	65.17	78407
Accuracy %		70.306		78407

Table 4.4 proves the same conclusion of Figure 4.5 since the precision, recall, and F1-score of the XGB and LGBM models are the best values. Precision, recall, F1-score, and accuracy of the LGBM model are 99%, 98%, 99%, and 98.7%, respectively. The same metrics of the XGB model are 99.3%, 99.08%, 99.19%, and 99.28%, respectively.

Table 4.5 shows the results of the fused/ensemble ML and DL models, and it's obvious that the ensemble ML model is the best one with precision, recall, F1-score, and accuracy values of 99.27%, 99.1%, 99.18%, and 99.28%, respectively. Results of Table 4.5 and the confusion matrixes along with the ROC plot prove that the best model is the ML-based ensemble model with 99.28% accuracy. In contrast, the DL-based fusion and ensemble models improved the performance of the individual DL model significantly. The DL ensemble model accuracy increased by 13.33% compared to the best individual DL model.

Table 4.5. Fused/Ensemble models accuracy, precision, recall, and F1-score metrics (with feature selection, Number of features=20).

Model	**ML-Based Fusion**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	100	100	100	8636
Charger	98.55	98.6	98.58	7924
Jisut	99.59	99.34	99.47	5070
Koler	99.56	99.77	99.66	8854
Lockerpin	99.49	99.51	99.5	4984
Pletor	99.0	97.58	98.32	961
PornDroid	99.47	99.65	99.56	9259
RansomBO	98.67	98.35	98.51	8023
SVpeng	98.95	99.25	99.1	10922
Simplocker	99.2	99.02	99.11	7325
WannaLocker	99.26	99.17	99.22	6449
Average Value	99.26	99.11	99.18	78407
Accuracy %	99.255			78407
Model	**ML-Based Ensemble**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	100	100	100	8636
Charger	98.74	98.68	98.71	7924
Jisut	99.61	99.34	99.48	5070
Koler	99.53	99.77	99.65	8854
Lockerpin	99.53	99.51	99.52	4984
Pletor	98.97	97.17	98.07	961
PornDroid	99.44	99.63	99.54	9259
RansomBO	98.72	98.39	98.55	8023
SVpeng	98.94	99.32	99.13	10922
Simplocker	99.23	99.12	99.18	7325
WannaLocker	99.26	99.16	99.21	6449
Average Value	99.27	99.1	99.18	78407
Accuracy %	99.28			78407

Model	**DL-Based Fusion**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	94.53	96.32	95.42	8636
Charger	65.19	75.38	69.91	7924
Jisut	77.92	85.08	81.34	5070
Koler	88.22	86.24	87.22	8854
Lockerpin	80.7	67.44	73.48	4984
Pletor	79.21	64.98	71.4	961
PornDroid	83.12	75.35	79.05	9259

RansomBO	91.31	85.81	88.47	8023
SVpeng	85.45	96.02	90.42	10922
Simplocker	90.87	83.31	86.93	7325
WannaLocker	89.66	88.84	89.25	6449
Average Value	84.2	82.25	82.99	78407
Accuracy %	84.68			78407
Model	DL-Based Ensemble			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***No. Test samples***
Benign	99.15	99.21	99.18	8636
Charger	95.53	86.12	90.58	7924
Jisut	90.8	90.09	90.44	5070
Koler	95.48	92.07	93.74	8854
Lockerpin	91.65	91.03	91.34	4984
Pletor	89.57	79.72	84.36	961
PornDroid	94.1	94.77	94.43	9259
RansomBO	89.74	88.51	89.12	8023
SVpeng	89.68	95.77	92.63	10922
Simplocker	90.11	92.92	91.49	7325
WannaLocker	91.14	96.12	93.56	6449
Average Value	92.45	91.48	91.9	78407
Accuracy %	92.78			78407

4.4.1. Different Feature Selection Scenarios Comparison

As mentioned in previous sections, three different feature selection scenarios are proposed with 20, 15, and 5 selected features. Table 4.6 includes the precision, recall, F1-score, and accuracy of all three scenarios.

Table 4.6. Precision, recall, F1-score, and accuracy of all scenarios (original and feature selection) for all individual ML and DL models.

Model	**LGBM**				
Metrics	***Precision %***	***Recall %***	***F1-score %***	***Accuracy %***	***Training time***
Original	**100**	**99**	**99**	**99.65**	23min 17s
20 selected features	99	98	99	98.7	6min 53s
15 selected features	99	98	98	98.67	6min 9s
5 selected features	94	94	94	93.31	5min 18s
Model	**K-NN**				
Metrics	***Precision %***	***Recall %***	***F1-score %***	***Accuracy %***	***Training time***
Original	45	43	43	45	2min 25s
20 selected features	71	69	70	72.01	1min 56s
15 selected features	72	70	71	72.8	18.1 s
5 selected features	**75**	**75**	**75**	**74.8**	9.69 s
Model	**DT**				
Metrics	***Precision %***	***Recall %***	***F1-score %***	***Accuracy %***	***Training time***
Original	**99.67**	**99.64**	**99.66**	**99.79**	10.9 s
20 selected features	98.69	98.63	98.66	98.58	2.59 s
15 selected features	98.75	98.62	98.69	98.9	1.75 s
5 selected features	97.82	97.77	97.8	97.59	1.24 s
Model	**RF**				
Metrics	***Precision %***	***Recall %***	***F1-score %***	***Accuracy %***	***Training time***
Original	95.56	93.93	94.65	95.82	3min 57s
20 selected features	97.2	96.35	96.75	97.26	1min 27s
15 selected features	**97.58**	**96.84**	**97.19**	**97.65**	1min 15s
5 selected features	96.51	96.5	96.5	96.27	1min 12s
Model	**AdaBoost**				
Metrics	***Precision %***	***Recall %***	***F1-score %***	***Accuracy %***	***Training time***
Original	37.64	46.12	39.74	54.96	1min 32s
20 selected features	**37.7**	**46.18**	**39.81**	**55.077**	31 s
15 selected features	37.7	46.18	39.81	55.7	22.2 s
5 selected features	23.35	32.88	25.77	41.5	30.4 s

Model	**XGB**				
Metrics	***Precision %***	***Recall %***	***F1-score %***	***Accuracy %***	***Training time***
Original	**99.86**	**99.8**	**99.83**	**99.87**	18min 16s
20 selected features	99.3	99.08	99.19	99.28	8min 45s
15 selected features	99.26	99.03	99.15	99.26	6min 24s
5 selected features	97.47	97.42	97.44	97.13	3min 39s
Model	**1D-CNN**				
Metrics	***Precision %***	***Recall %***	***F1-score %***	***Accuracy %***	***Training time***
Original	73.3	59.92	63.73	64.17	6min 23s
20 selected features	**84.45**	**76.96**	**79.63**	**79.45**	6min 8s
15 selected features	80.61	73.65	75.76	75.98	5min 22s
5 selected features	26.31	27.31	23.35	33.17	5min 28s
Model	**LSTM**				
Metrics	***Precision %***	***Recall %***	***F1-score %***	***Accuracy %***	***Training time***
Original	**85.76**	**83.13**	**84.05**	**86.46**	4min 35s
20 selected features	67.56	64.85	65.17	70.3	3min 3s
15 selected features	83.2	80.67	81.47	84.94	2min 23s
5 selected features	30.94	30.06	25.41	35.32	2min 25s

Table 4.6 shows that the LGBM and XGB models (the best models in case of no feature selection) are also the best in all feature selection scenarios. For the XGB model, although the feature selection method minimizes the number of features by more than 4 times, the precision, recall, F1-score, and accuracy are reduced by only 0.56%, 0.72%, 0.64%, and 0.59, respectively. This means that the percentage of features which is only (20/84) 23.8% of the entire features achieved almost the same performance as the entire dataset's features (100%), and this clarifies the importance and efficiency of the feature selection algorithm proposed in this study. Table 4.7 includes the precision, recall, F1-score, and accuracy of all three scenarios.

Table 4.7. Precision, recall, F1-score, and accuracy of all scenarios (original and feature selection) for all fusion/ensemble ML and DL models.

Model	**ML-Based Fusion**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***Accuracy %***
Original	**99.7**	**99.54**	**99.62**	**99.76**
20 selected features	99.26	99.11	99.18	99.255
15 selected features	99.29	99.07	99.18	99.3
5 selected features	97.37	99.37	97.37	97.12
Model	**ML-Based Ensemble**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***Accuracy %***
Original	**99.7**	**99.59**	**99.67**	**99.79**
20 selected features	99.27	99.1	99.18	99.28
15 selected features	99.24	99.04	99.14	99.27
5 selected features	97.36	97.35	97.35	97.08
Model	**DL-Based Fusion**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***Accuracy %***
Original	86.15	83.02	84.12	86.55
20 selected features	84.2	82.25	82.99	84.68
15 selected features	**85.77**	**83.27**	**84.16**	**86.7**
5 selected features	28.63	28.87	23.57	34.41
Model	**DL-Based Ensemble**			
Metrics	***Precision %***	***Recall %***	***F1-score %***	***Accuracy %***
Original	88.93	86.8	87.63	89.58
20 selected features	92.45	91.48	91.9	92.78
15 selected features	**93.77**	**93.02**	**93.37**	**94.34**
5 selected features	38.66	29.48	23.76	35.32

4.5. DISCUSSION

4.5.1. Discussion of the Effect of Feature Selection Algorithm on The Performance of Ransomware Detection Models

Table 4.5 and Table 4.6 include detailed performance metrics (precision, recall, F1-score, accuracy, and training time). The training time is decreasing with more feature reduction rates. Using only 20 features (23.8% of the entire features) reduced the training time 3.38 times (23 min 17s / 6 min 53 s) for the LGBM model. Similarly, training time is decreased 3.78 times. For a more detailed comparison, Figure 4.8 includes a complete comparison of training time with and without feature selection for all individual ML/DL models. For all models, the training time is decreased by applying the feature selection algorithm. Figure 4.9 shows the performance metrics of all models to compare the performance.

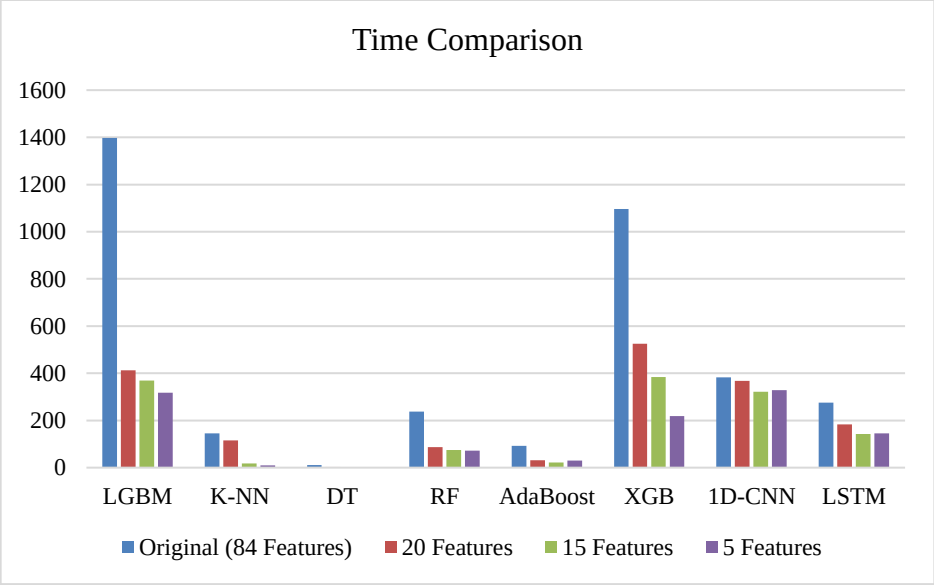
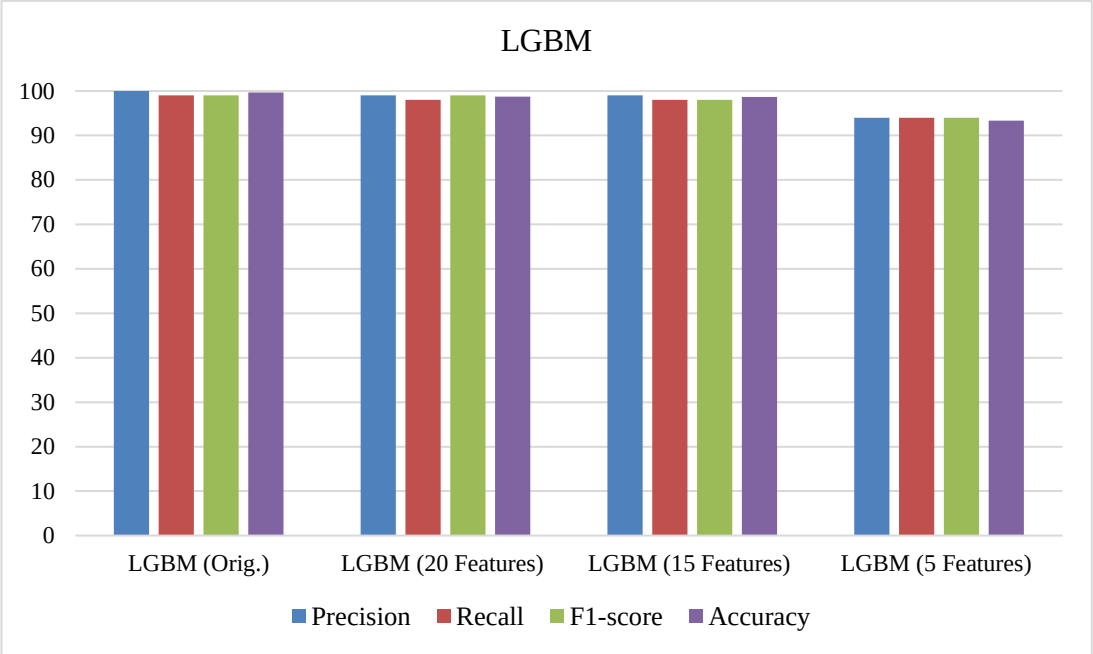
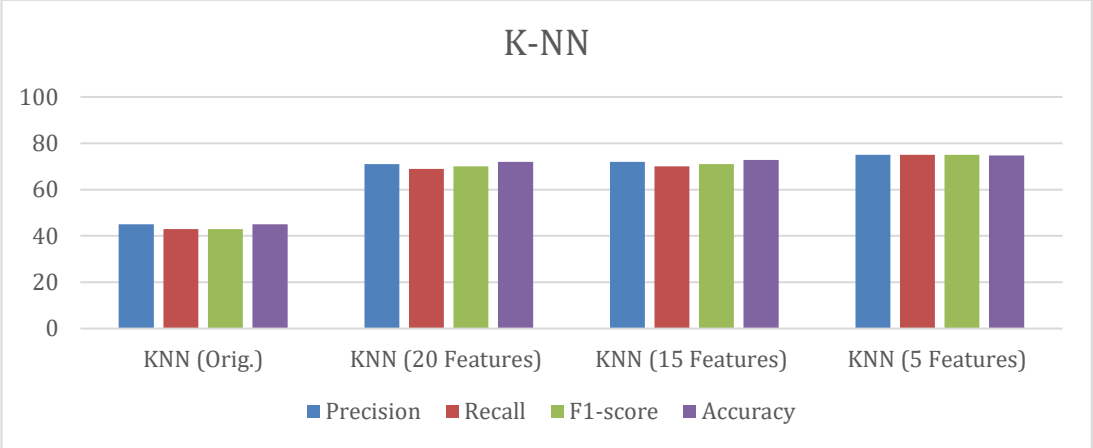
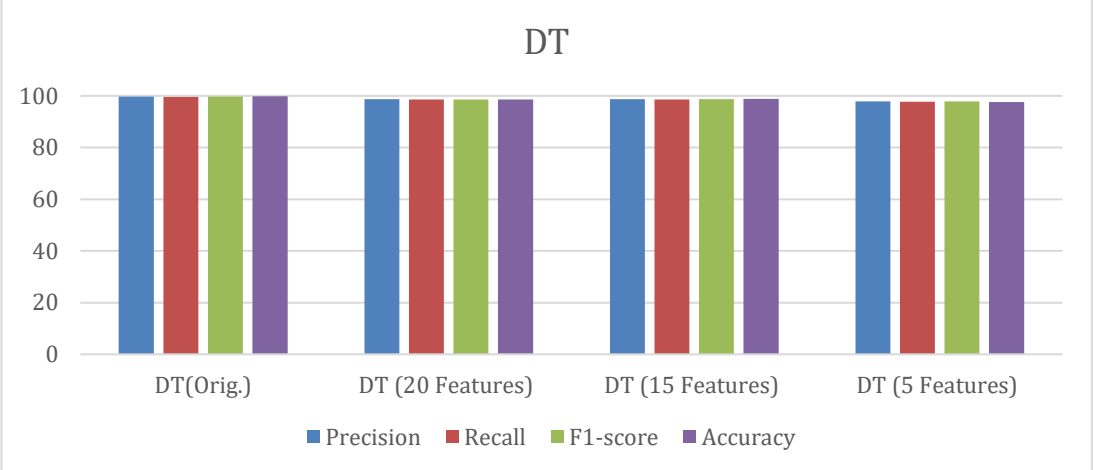


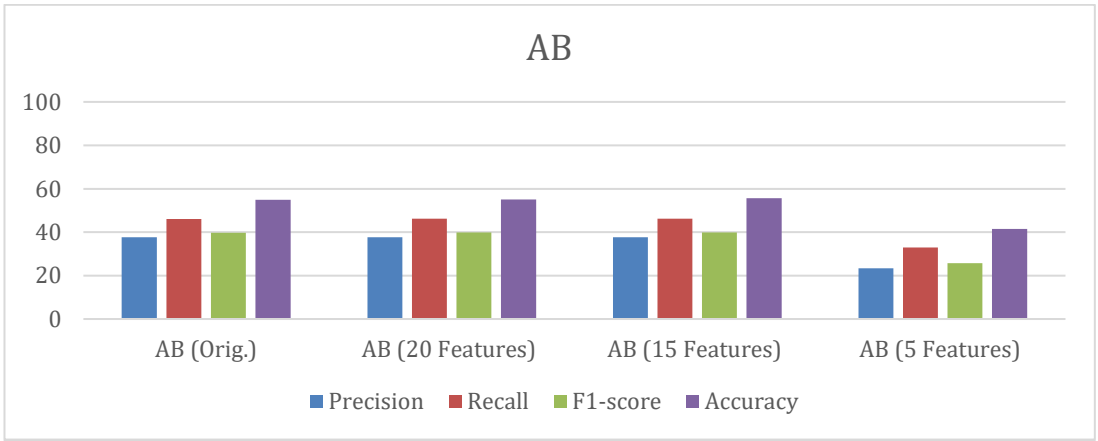
Figure 4.8. Training time comparison before and after applying the proposed feature selection algorithm.


LGBM
100
90
80
70
60
50
40
30
20
10
0
LGBM (Orig.)
LGBM (20 Features)
LGBM (15 Features)
LGBM (5 Features)
Precision
Recall
F1-score
Accuracy


K-NN
100
80
60
40
20
0
KNN (Orig.)
KNN (20 Features)
KNN (15 Features)
KNN (5 Features)
Precision
Recall
F1-score
Accuracy


DT
100
80
60
40
20
0
DT(Orig.)
DT (20 Features)
DT (15 Features)
DT (5 Features)
Precision
Recall
F1-score
Accuracy


RF
100
80
60
40
20
0
RF(Orig.)
RF (20 Features)
RF (15 Features)
RF (5 Features)
Precision
Recall
F1-score
Accuracy


AB
100
80
60
40
20
0
AB (Orig.)
AB (20 Features)
AB (15 Features)
AB (5 Features)
Precision
Recall
F1-score
Accuracy

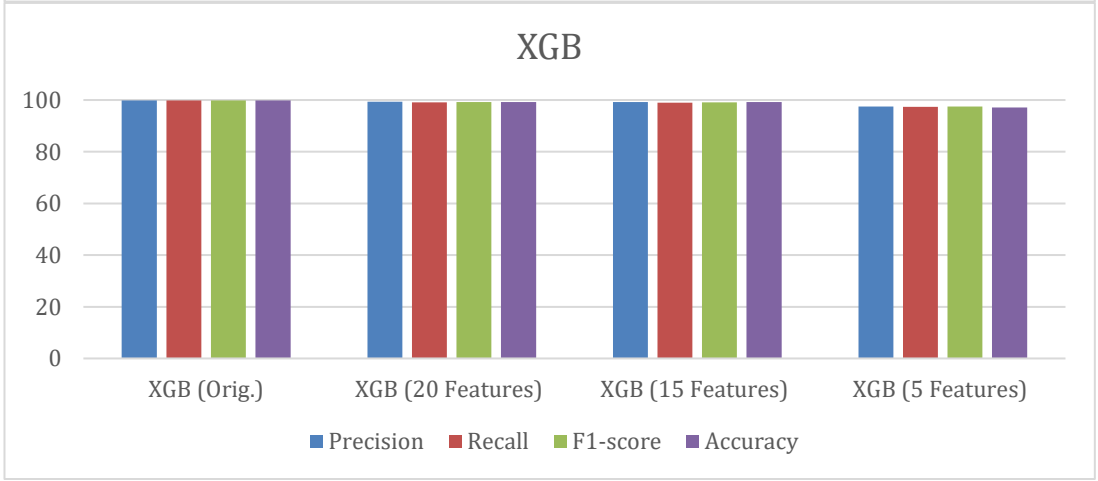

XGB
100
80
60
40
20
0
XGB (Orig.)
XGB (20 Features)
XGB (15 Features)
XGB (5 Features)
Precision
Recall
F1-score
Accuracy

Figure 4.9. Performance metrics of all models with and without feature selection.

Figure 4.9 illustrates many results: First, the robust models like XGB, LGBM, DT, and RF have almost similar performance in the four scenarios (without feature selection, with 20 selected features, with 15 selected features, with 5 selected features). The performance of the K-NN model has been increased by minimizing the number of features. In some cases, the best performance corresponds to the case of 15 selected features. Results show that with 5 selected features, the performance degrades again.

4.5.2. Discussion of the Fusion/Ensemble Dl/Ml Models Results

4.5.2.1. ML Fusion and Ensemble Discussion

For the ML fusion and ensemble model, results show that the fusion enhanced the low-performance models but it preserves the performance of robust models and in some cases of feature selection, it improves the performance. Figure 4.10 shows a performance comparison between the ML fusion and ensemble models in the case of different feature selection scenarios.

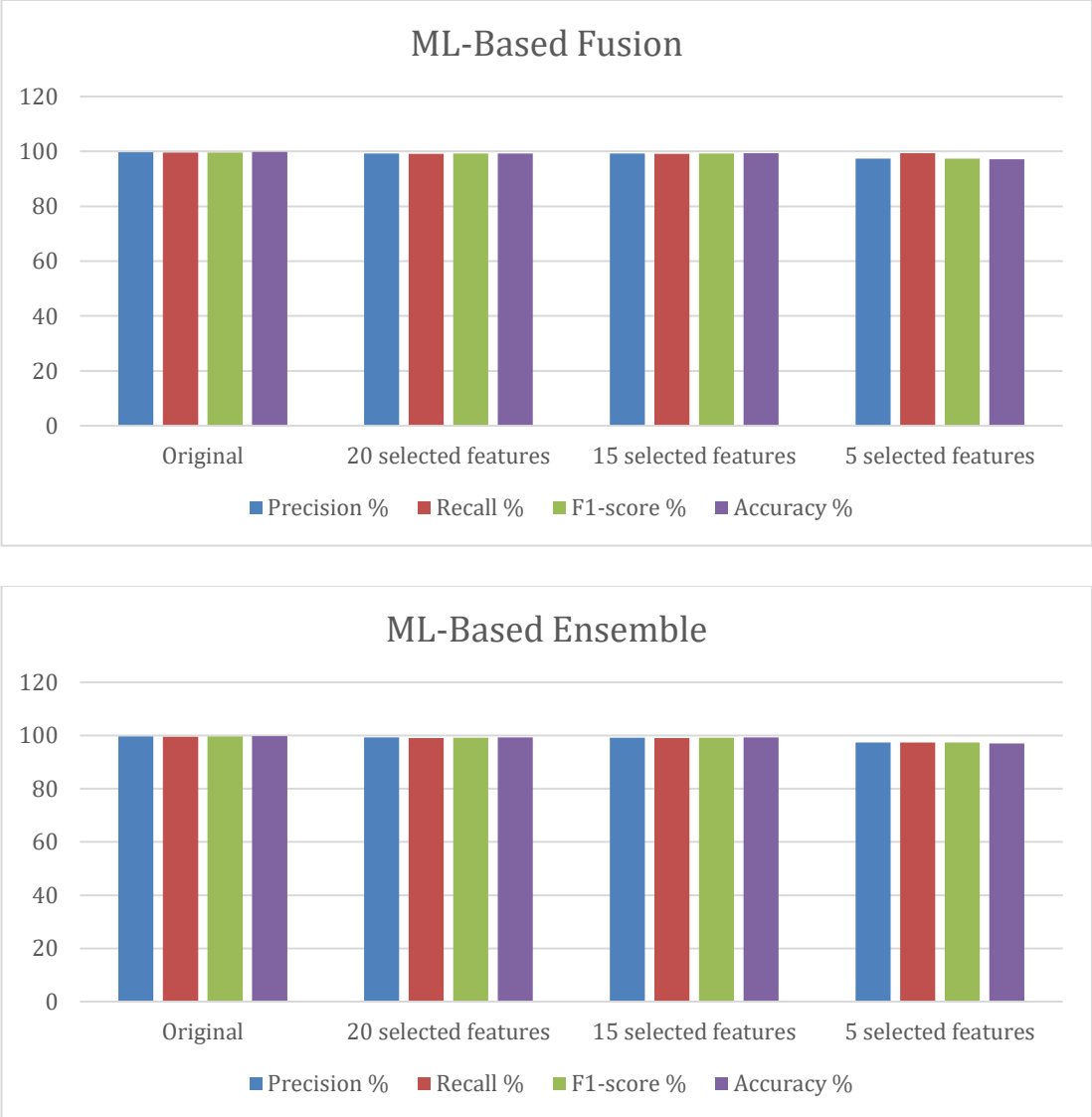


Figure 4.10. Performance metrics of Fusion/Ensemble ML models with and without feature selection.

In the case of ML fusion and ensemble models, the feature selection method decreased the performance a little bit (for example, the precision in the case of a 20-feature-selected scenario decreased by only 0.11% compared to the original 84 features).

4.5.2.2. DL Fusion and Ensemble Discussion

Figure 4.11 includes a performance comparison between the DL fusion and ensemble models in case of different feature selection scenarios. However, the ML fusion and ensemble models achieved a better performance compared to the DL fusion and ensemble models.

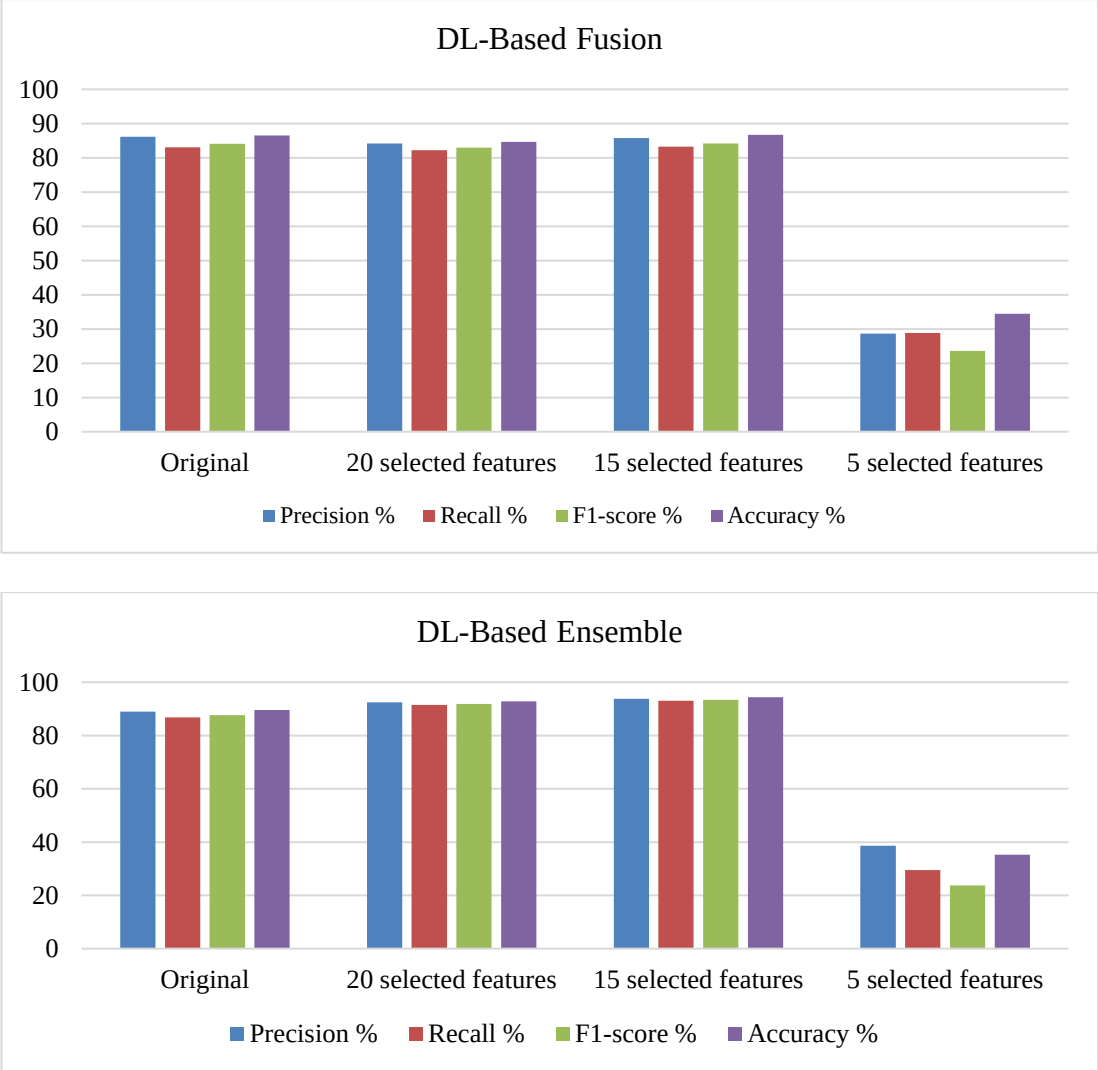


Figure 4.11. Performance metrics of Fusion/Ensemble DL models with and without feature selection.

The main remark in these two scenarios is that the 20-selected-features and 15- 15-selected-feature scenarios improved the performance compared to the original 84-feature scenario. However, with less than 10 features, performance starts to degrade.

4.6. COMPARISON WITH THE CURRENT STATE-OF-ART

Table 4.8 includes a comprehensive comparison between the current study and the previous state-of-the-art ransomware detection and classification methodologies.

Table 4.8. A comprehensive comparison between the current study and the previous state-of-art ransomware detection and classification methodologies.

Study	ML/DL Methods	Dataset	Results	Notes
Takeuchi et al. [32]	SVM	276 ransomware, 312 benign	Accuracy: 97.5%	Small dataset
Qin et al. [58]	TextCNN	1000 ransomware, 1000 benign	Accuracy: 95.9%	Small dataset
Homayoun et al. [59]	Sequential pattern mining	1624 ransomware, 220 benign	Accuracy: 99.4%	Small number of ransomware types
Khammas [60]	Random Forests	840 ransomware, 840 benign	Accuracy: 97.7%	The small number of ransomware types
Manavi and Hamzeh [35]	CNN	1000 ransomware, 1000 benign	Accuracy: 93.33%	Small number of ransomware types
Almousa et al. [36]	RF, K-NN, SVM	500 ransomware, 500 benign	Accuracy: 99.18%	Small dataset
Rani and Dhavale [40]	DT, RF, K-NN, SVM, XGBoost, LR	582 ransomware, 942 benign	Best Accuracy: 98.21%	Small dataset
Silva and Alvarez [43]	Boosted Trees, Naïve Bayes, RF, NN	2000 ransomware, benign	Accuracy: ~99%	Small dataset
Hitaj et al. [44]	RF	ShieldFS, Cerberus	Accuracy: 99.45%	Detects ransomware occurrence, not its type
Current Study	LGBM, XBG, K-NN, DT, RF, AB, Fusion, Ensemble, 1DCNN, LSTM + Feature Selection	392034 records and 84 features, 11 types of ransomware	Best Accuracy: 99.87%	Detects 11 types of ransomware, Applies feature selection

Table 4.8 proves that the current study outperforms all previous state-of-the-art methodologies.

PART 5

CONCLUSION

This study proposes a new ransomware detection and classification methodology, including a main feature selection algorithm that combines wrapper and filter-based feature selection methods at the same algorithm to get the best efficiency and performance. The study used a ransomware dataset containing many challenges; the dataset contains 84 predictors (high dimensionality) and 392034 records (huge size). First, the dataset is preprocessed using the data-cleaning operations. The encoding operations and the normalization are also performed to prepare the dataset for the ML and DL models. The following steps included the dataset split into training and test sets. The feature selection algorithm was also applied to the preprocessed dataset to get the most essential predictors (features) and remove the redundant ones. The feature selection algorithm was based on two main approaches: the ANOVA feature selection method and the RF estimator. The feature selection method is adaptive so that the number of selected features can vary based on the parameters of the algorithm. In our implementation, the results showed three different cases of the feature selection algorithm with 20 selected features, 15 selected features, and 5 selected features with 23.8%, 17.85%, and 5.9% selection percentages, respectively. In the next step, many ML and DL models were trained using the training set (once without feature selection and once with feature selection). The results are categorized into four scenarios: one without feature selection and three with feature selection. The fusion and ensemble of the ML models were also performed to improve the performance. The same fusion and ensemble were also applied to the DL models. Results showed that the best ML models are the XGB and LGBM in all scenarios.

The results also indicated that the worst models were the AdaBoost and K-NN. However, K-NN performance enhanced in the case of feature selection. Results also indicated that the feature selection preserves the high performance of the robust models (like XGB and LGBM) with a very high feature reduction rate, resulting in a low computational time compared to the original dataset (84 features). The LGBM model's training time was minimized from 23 minutes and 17 seconds to only 6 minutes and 53 seconds using the 20 selected features and minimized more with 15 and 5 selected features without any remarkable performance degradation. Results also indicated enhancement in the performance of ML and DL models in the case of ensemble learning. The DL model's performance was improved significantly using ensemble learning. The comparison of the current study with the previous state-of-art studies proved that this study outperformed all previous studies' performance. According to previous results, the best choice of feature selection is the 15-feature selection scenario (since it reduces time and preserves or improves performance).

Next, studies can focus on enhancing deep learning models by using hyperparameter optimizations for better performance. Other studies can try different ransomware datasets and evaluate the performance under different conditions. Other studies can try different feature selection algorithms and compare their results with current ones.

REFERENCES

[1] U. Zahoora, A. Khan, M. Rajarajan, S. H. Khan, M. Asam, and T. Jamal, "Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive Pareto Ensemble classifier," **Scientific Reports**, vol. 12, no. 1, p. 15647, 2022.

[2] C. Comito, A. Forestiero, and C. Pizzuti, "Word embedding based clustering to detect topics in social media," in ***IEEE/WIC/ACM International Conference on Web Intelligence***, 2019, pp. 192-199.

[3] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions," ***Artificial Intelligence*** Review, vol. 55, no. 1, pp. 453.563, 2022.

[4] J. Singh, K. Sharma, M. Wazid, and A. K. Das, "SINN-RD: Spline interpolation-envisioned neural network-based ransomware detection scheme," ***Computers and Electrical Engineering***, vol. 106, p. 108601, 2023.

[5] H. Sun, G. Xu, Z. Wu, and R. Quan, "Android Malware Detection Based on Feature Selection and Weight Measurement," ***Intelligent Automation & Soft Computing***, vol. 33, no. 1, 2022.

[6] D. W. Fernando and N. Komninos, "FeSA: Feature selection architecture for ransomware detection under concept drift," ***Computers & Security***, vol. 116, p. 102659, 2022.

[7] R. Ali, A. Ali, F. Iqbal, M. Hussain, and F. Ullah, "Deep Learning Methods for Malware and Intrusion Detection: A Systematic Literature Review," ***Security and Communication Networks***, vol. 2022, 2022.

[8] C. COP. Android Ransomware Detection, doi: 10.34740/kaggle/dsv/4987535.

[9] G. Mott et al., "Between a rock and a hard (ending) place: Cyber insurance in the ransomware era," ***Computers & Security***, vol. 128, p. 103162, 2023.

[10] D. Su, J. Liu, X. Wang, and W. Wang, "Detecting Android locker-ransomware on chinese social networks," ***IEEE Access***, vol. 7, pp. 20381-20393, 2018.

[11] D. Braue. (2022) Global Ransomware Damage Costs Predicted To Exceed $265 Billion By 2031. Cybercrime magazine. Available: https://cybersecurityventures.com/global-ransomware-damage-costs-predicted-to-reach-250-billion-usd-by-2031/

[12] A. Petrosyan. "Countries targeted by ransomware 2022. [**Online**]" © Statista 2023.https://www.statista.com/statistics/1377918/ransomware-target-countries/ (accessed Jun 1 2023, 2023).

[13] Prrofpoint.com, "The State of Ransomware 2022," 2022 [**Online**]. Available: https://www.proofpoint.com/sites/default/files/threat-reports/pfpt-us-tr-state-of-the-phish-2022.pdf

[14] K. Begovic, A. Al-Ali, and Q. Malluhi, "Cryptographic Ransomware Encryption Detection: Survey," Computers & Security, p. 103349, 2023.

[15] C. Lamers, E. Spoerl, G. Levey, N. Choudhury, and M. Ahmed, "Ransomware: A Threat to Cyber Smart Cities," in Cybersecurity for Smart Cities: Practices and Challenges: ***Springer***, 2023, pp. 185-204.

[16] T. Baker and A. Shortland, "Insurance and enterprise: cyber insurance for ransomware," ***The Geneva Papers on Risk and Insurance-Issues and Practice***, vol. 48, no. 2, pp. 275-299, 2023.

[17] S. Thakur, S. Chaudhari, and B. Joshi, "Ransomware: Threats, Identification and Prevention," ***Cyber Security and Digital Forensics***, pp. 361-387, 2022.

[18] M. Humayun, N. Jhanjhi, A. Alsayat, and V. Ponnusamy, "Internet of things and ransomware: Evolution, mitigation and prevention," ***Egyptian Informatics Journal***, vol. 22, no. 1, pp. 105-117, 2021.

[19] M. Keshavarzi and H. R. Ghaffary, "I2CE3: A dedicated and separated attack chain for ransomware offenses as the most infamous cyber extortion," ***Computer Science*** Review, vol. 36, p. 100233, 2020.

[20] E. B. Karbab, M. Debbabi, and A. Derhab, "SwiftR: Cross-platform ransomware fingerprinting using hierarchical neural networks on hybrid features," ***Expert Systems with Applications***, vol. 225, p. 120017, 2023.

[21] S. Alsoghyer and I. Almomani, "Ransomware detection system for Android applications," ***Electronics***, vol. 8, no. 8, p. 868, 2019.

[22] B. Marais, T. Quertier, and S. Morucci, "AI-based Malware and Ransomware Detection Models," in Conference on ***Artificial Intelligence for Defense***, 2022.

[23] P. Sharma, S. Kapoor, and R. Sharma, "Ransomware detection, prevention and protection in IoT devices using ML techniques based on dynamic analysis approach," ***International Journal of System Assurance Engineering and Management***, vol. 14, no. 1, pp. 287-296, 2023.

[24] A. Almaleh, R. Almushabb, and R. Ogran, "Malware API Calls Detection Using Hybrid Logistic Regression and RNN Model," ***Applied Sciences***, vol. 13, no. 9, p. 5439, 2023.

[25] H. Bakır and R. Bakır, "DroidEncoder: Malware detection using auto-encoder based feature extractor and machine learning algorithms," ***Computers and Electrical Engineering***, vol. 110, p. 108804, 2023.

[26] M. Mijwil, I. E. Salem, and M. M. Ismaeel, "The Significance of Machine Learning and Deep Learning Techniques in Cybersecurity: A Comprehensive Review," ***Iraqi Journal For Computer Science and Mathematics***, vol. 4, no. 1, pp. 87-101, 2023.

[27] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection," ***arXiv*** preprint arXiv:1609.03020, 2016.

[28] R. Vinayakumar, K. Soman, K. S. Velan, and S. Ganorkar, "Evaluating shallow and deep networks for ransomware detection and classification," in 2017 **international conference on advances in Computing, communications and informatics (ICACCI), 2017: *IEEE***, pp. 259-265.

[29] A. Cohen and N. Nissim, "Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory," ***Expert Systems with Applications***, vol. 102, pp. 158-178, 2018.

[30] S. K. Shaukat and V. J. Ribeiro, "RansomWall: A layered defense system against cryptographic ransomware attacks using machine learning," in 2018 ***10th international conference on communication systems & networks (COMSNETS), 2018: IEEE***, pp. 356-363.

[31] S. Maniath, A. Ashok, P. Poornachandran, V. Sujadevi, P. S. AU, and S. Jan, "Deep learning LSTM based ransomware detection," in 2017 Recent Developments in Control, ***Automation & Power Engineering (RDCAPE), 2017: IEEE***, pp. 442-446.

[32] Y. Takeuchi, K. Sakai, and S. Fukumoto, "Detecting ransomware using support vector machines," in Workshop Proceedings of the ***47th International Conference on Parallel Processing***, 2018, pp. 1-6.

[33] J. Hwang, J. Kim, S. Lee, and K. Kim, "Two-stage ransomware detection using dynamic analysis and machine learning techniques," ***Wireless Personal Communications***, vol. 112, pp. 2597-2609, 2020.

[34] H. Zuhair, A. Selamat, and O. Krejcar, "A multi-tier streaming analytics model of 0-day ransomware detection using machine learning," Applied Sciences, vol. 10, no. 9, p. 3210, 2020.

[35] F. Manavi and A. Hamzeh, "A new method for ransomware detection based on PE header using convolutional neural networks," in 2020 17th International ***ISC*** Conference on Information Security and Cryptology (ISCISC), 2020: IEEE, pp. 82-87.

[36] M. Almousa, S. Basavaraju, and M. Anwar, "API-Based Ransomware Detection Using Machine Learning-Based Threat Detection Models," in 2021 18th

International Conference on Privacy, Security and Trust (PST), 2021: ***IEEE***, pp. 1-7.

[37] M. Masum, M. J. H. Faruk, H. Shahriar, K. Qian, D. Lo, and M. I. Adnan, "Ransomware classification and detection with machine learning algorithms," in 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), 2022: ***IEEE***, pp. 0316-0322.

[38] K. A. Alissa et al., "Dwarf mongoose optimization with machine-learning-driven ransomware detection in internet of things environment," ***Applied Sciences***, vol. 12, no. 19, p. 9513, 2022.

[39] R. Bold, H. Al-Khateeb, and N. Ersotelos, "Reducing False Negatives in Ransomware Detection: A Critical Evaluation of Machine Learning Algorithms," ***Applied Sciences***, vol. 12, no. 24, p. 12941, 2022.

[40] N. Rani and S. V. Dhavale, "Leveraging machine learning for ransomware detection," arXiv preprint ***arXiv***:2206.01919, 2022.

[41] Y. A. Ahmed et al., "A weighted minimum redundancy maximum relevance technique for ransomware early detection in industrial IoT," ***Sustainability***, vol. 14, no. 3, p. 1231, 2022.

[42] Q. M. Yaseen, "The Effect of the Ransomware Dataset Age on the Detection Accuracy of Machine Learning Models," ***Information***, vol. 14, no. 3, p. 193, 2023.

[43] J. A. Herrera-Silva and M. Hernández-Álvarez, "Dynamic Feature Dataset for Ransomware Detection Using Machine Learning Algorithms," ***Sensors***, vol. 23, no. 3, p. 1053, 2023.

[44] D. Hitaj, G. Pagnotta, F. De Gaspari, L. De Carli, and L. V. Mancini, "Minerva: A File-Based Ransomware Detector," arXiv preprint ***arXiv***:2301.11050, 2023.

[45] L. Bekkers, S. van't Hoff-de Goede, E. Misana-ter Huurne, Y. van Houten, R. Spithoven, and E. R. Leukfeldt, "Protecting your business against ransomware attacks? Explaining the motivations of entrepreneurs to take future protective measures against cybercrimes using an extended protection motivation theory model," ***Computers & Security***, vol. 127, p. 103099, 2023.

[46] S. Song, B. Kim, and S. Lee, "The effective ransomware prevention technique using process monitoring on android platform," ***Mobile Information System***s, vol. 2016, 2016.

[47] A. AlSabeh, H. Safa, E. Bou-Harb, and J. Crichigno, "Exploiting ransomware paranoia for execution prevention," in ICC 2020-2020 IEEE International Conference on Communications (ICC), 2020: ***IEEE***, pp. 1-6.

[48] A. Wani and S. Revathi, "Ransomware protection in loT using software defined networking," ***Int. J. Electr. Comput. Eng***, vol. 10, no. 3, pp. 3166-3175, 2020.

[49] A. Ibrahim, U. Tariq, T. Ahamed Ahanger, B. Tariq, and F. Gebali, "Retaliation against Ransomware in Cloud-Enabled PureOS System," Mathematics, vol. 11, no. 1, p. 249, 2023.

[50] Y.-S. Lin and C.-F. Lee, "Ransomware Detection and Prevention through Strategically Hidden Decoy File," ***International Journal of Network Security***, vol. 25, no. 2, pp. 212-220, 2023.

[51] S. Kok, A. Abdullah, and N. Jhanjhi, "Early detection of crypto-ransomware using pre-encryption detection algorithm," ***Journal of King Saud University-Computer and Information Sciences,*** vol. 34, no. 5, pp. 1984-1999, 2022.

[52] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation," in AI 2006: Advances in Artificial Intelligence: 19th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, December 4-8, 2006. Proceedings 19, 2006: ***Springer***, pp. 1015-1021.

[53] S. Tangirala, "Evaluating the impact of GINI index and information gain on classification using decision tree classifier algorithm," ***International Journal of Advanced Computer Science and Applications***, vol. 11, no. 2, pp. 612-619, 2020.

[54] M. Dener, G. Ok, and A. Orman, "Malware detection using memory analysis data in big data environment," ***Applied Sciences***, vol. 12, no. 17, p. 8604, 2022.

[55] S. Deng, L. Wang, S. Guan, and M. Li, "Non-parametric Nearest Neighbor Classification Based on Global Variance Difference," ***International Journal of Computational Intelligence Systems***, vol. 16, no. 1, p. 26, 2023.

[56] S. Demir and E. K. Sahin, "An investigation of feature selection methods for soil liquefaction prediction based on tree-based ensemble algorithms using AdaBoost, gradient boosting, and XGBoost," ***Neural Computing and Applications***, vol. 35, no. 4, pp. 3173.3190, 2023.

[57] Z. Faska, L. Khrissi, K. Haddouch, and N. El Akkad, "A robust and consistent stack generalized ensemble-learning framework for image segmentation," ***Journal of Engineering and Applied Science***, vol. 70, no. 1, pp. 1-20, 2023.

[58] B. Qin, Y. Wang, and C. Ma, "API call based ransomware dynamic detection approach using textCNN," in 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), 2020: ***IEEE***, pp. 162-166.

[59] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence," ***IEEE transactions on emerging topics in computing***, vol. 8, no. 2, pp. 341-351, 2017.

[60] B. M. Khammas, "Ransomware detection using random forest technique," ***ICT Express***, vol. 6, no. 4, pp. 325-331, 2020.

RESUME

Elaf Talib Abduljabbar ABDULJABBAR began her academic journey in Salaheldin, Iraq. She completed her secondary education at Al Masafy High School in the 2008-2009 academic year. Subsequently, she pursued her undergraduate studies at Al-Rafidain University, graduating in 2012-2013. In 2021, She moved to Karabuk, Turkey, to undertake postgraduate studies. She enrolled in a Master of Science program in Computer Engineering at Karabuk University.