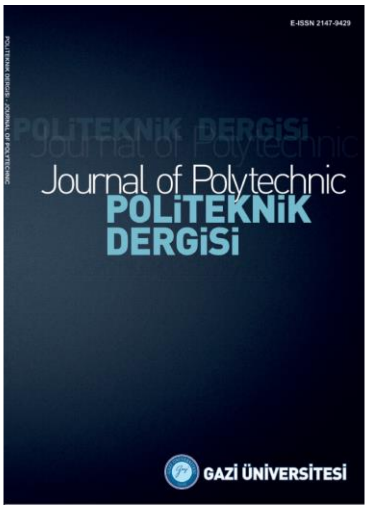


POLİTEKNİK DERGİSİ

JOURNAL of POLYTECHNIC

ISSN: 1302-0900 (PRINT), ISSN: 2147-9429 (ONLINE)
URL: http://dergipark.org.tr/politeknik

Enhancing cybersecurity against ransomware attacks using LSTM deep learning method: A case study on android devices

LSTM derin öğrenmeyi kullanarak fidye yazılımı saldırılarına karşı siber güvenliğin geliştirilmesi: Android cihazlarda bir vaka çalışması

***(Author(s))**: Hatice KARACA[1], Adem TEKEREK[2]*

ORCID[1]: 0009-0000-0294-816X

ORCID[2]: 0000-0002-0880-7955

__To cite to this article__: Karaca H. and Tekerek A., "Enhancing Cybersecurity against Ransomware Attacks Using LSTM Deep Learning: A Case Study on Android Devices", ***Journal of Polytechnic***, 28(2): 491-502, (2025).

__Bu makaleye şu şekilde atıfta bulunabilirsiniz:__ Karaca H. ve Tekerek A., "Enhancing Cybersecurity against Ransomware Attacks Using LSTM Deep Learning: A Case Study on Android Devices", ***Politeknik Dergisi***, 28(2): 491-502, (2025).

__Erişim linki (To link to this article):__ http://dergipark.org.tr/politeknik/archive

__DOI:__ 10.2339/politeknik.1508722

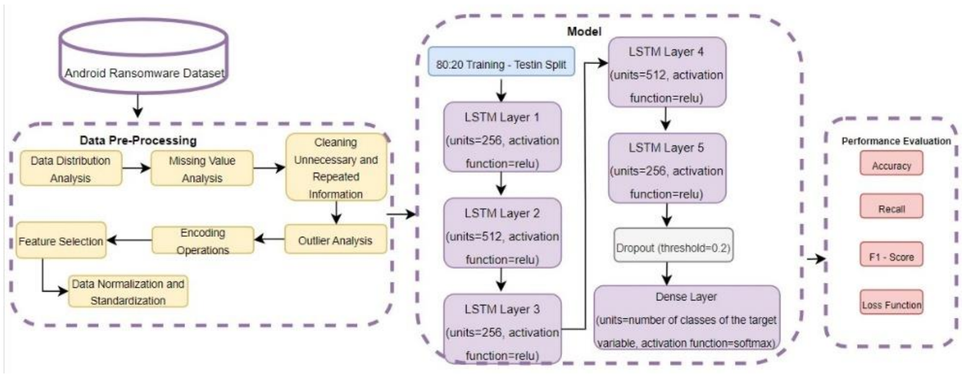
Enhancing Cybersecurity against Ransomware Attacks Using LSTM Deep Learning Method: A Case Study on Android Devices

Highlights

- *Detecting Android ransomware using the LSTM model.*
- *Applying dataset analysis and feature engineering.*
- *Detailing the model training and validation processes.*
- *Analyzing results with performance evaluation metrics.*

Graphical Abstract

In this study, the LSTM deep learning model was used to detect Android-based ransomware. Dataset analysis, feature engineering, model training, and performance evaluations were conducted, achieving a 99% accuracy rate.



Android Ransomware Dataset
Data Pre-Processing
Data Distribution Analysis
Missing Value Analysis
Cleaning Unnecessary and Repeated Information
Outlier Analysis
Encoding Operations
Feature Selection
Data Normalization and Standardization
Model
80:20 Training - Testin Split
LSTM Layer 1 (units=256, activation function=relu)
LSTM Layer 2 (units=512, activation function=relu)
LSTM Layer 3 (units=256, activation function=relu)
LSTM Layer 4 (units=512, activation function=relu)
LSTM Layer 5 (units=256, activation function=relu)
Dropout (threshold=0.2)
Dense Layer (units=number of classes of the target variable, activation function=softmax)
Performance Evaluation
Accuracy
Recall
F1 - Score
Loss Function

Figure. Proposed Methodology

Aim
To investigate the effectiveness of LSTM deep learning methods in detecting Android-based ransomware

Design & Methodology
Steps included dataset analysis, feature engineering, model training, and performance evaluation

Originality
This study is one of the rare works successfully applying LSTM models to detect Android ransomware.

Findings
Android ransomware was detected with 99% accuracy with LSTM.

Conclusion
LSTM has proven the effectiveness of deep learning in security by successfully detecting Android ransomware.

Declaration of Ethical Standards
The authors of this article declare that the materials and methods used in this study do not require ethical committee permission and/or legal-special permission

Enhancing Cybersecurity against Ransomware Attacks Using LSTM Deep Learning: A Case Study on Android Devices

Araştırma Makalesi / Research Article

Hatice KARACA, Adem TEKEREK[*]

Computer Engineering Department, Technology Faculty, Gazi University, Ankara, Türkiye

(Geliş/Received : 02.07.2024 ; Kabul/Accepted : 06.08.2024 ; Erken Görünüm/Early View : 09.08.2024)

ABSTRACT

The rapid advancement of technology brings new threats to the digital world. One of these threats is malicious ransomware attacks. Ransomware is malicious software that demands ransom from innocent users by blocking access to information systems. Since traditional methods are limited to predefined blacklists, they may be ineffective against unknown ransomware types. On the other hand, deep learning methods offer a sensitive defense mechanism against anomalies by learning standard behavior patterns. This study studied the Internet logs of Android devices consisting of 392,034 rows and 86 columns using the Long Short-Term Memory (LSTM) model. The dataset contains 14 different Android ransomware families and harmless traffic. Data preprocessing steps include missing data management, outlier analysis, feature selection, coding operations, and data normalization/standardization. The dataset was split at 80% training 20% test ratio, and it was determined that the 80% training 20% test split had the highest accuracy. The developed LSTM-based classification model achieved successful results with a 99% accuracy rate and 0.99 F1 score.

Keywords: Android, Ransomware, Deep Learning

LSTM Derin Öğrenmeyi Kullanarak Fidye Yazılımı Saldırılarına Karşı Siber Güvenliğin Geliştirilmesi: Android Cihazlarda Bir Vaka Çalışması

ÖZ

Teknolojinin hızla ilerlemesi dijital dünyada yeni tehditleri de beraberinde getiriyor. Bu tehditlerden biri kötücül fidye yazılımı saldırılarıdır. Fidye yazılımları, bilgi sistemlerine erişimi engelleyerek masum kullanıcılardan fidye talep eden kötü amaçlı yazılımlardır. Geleneksel yöntemler önceden tanımlanmış kara listelerle sınırlı olduğundan, bilinmeyen fidye yazılımı türlerine karşı etkisiz kalabilir. Derin öğrenme yöntemleri ise normal davranış kalıplarını öğrenerek anormalliklere karşı hassas bir savunma mekanizması sunar. Bu çalışmada Uzun Kısa Süreli Bellek (LSTM) modeli kullanılarak, 392.034 satır ve 86 sütundan oluşan Android cihazların İnternet günlükleri üzerinde çalışılmıştır. Veri seti, 14 farklı Android fidye yazılımı ailesi ve zararsız trafik içermektedir. Veri ön işleme adımları arasında eksik verilerin yönetimi, aykırı değer analizi, özellik seçimi, kodlama işlemleri ve veri normalleştirme/standartlaştırma bulunmaktadır. Veri kümesi %80 eğitim - %20 test oranında bölünmüş ve %80 eğitim - %20 test ayrımının en yüksek doğruluğa sahip olduğu belirlenmiştir. Geliştirilen LSTM tabanlı sınıflandırma modeli %99 doğruluk oranı ve 0,99 F1-skoru ile başarılı sonuçlar elde etmiştir

Anahtar Kelimeler: Android, Fidye Yazılımı, Derin Öğrenme

1. INTRODUCTION

Ransomware is malicious software that aims to damage or gain unauthorized access to computer systems, posing a severe threat to individual users, businesses, and government institutions. Ransomware can often take various forms, such as viruses, worms, trojans, spyware, and malicious adware. Ransomware is malicious software that infiltrates information systems, encrypts files, and then demands a ransom. It is one of the most common types of malicious software in the world [1].

Ransomware attacks have become a severe problem regarding information and computer security in recent years. These attacks can affect individuals and companies and cause significant financial losses. Ransomware attacks often have specific characteristics, such as anomaly network traffic or suspicious file behavior. Some events that occur as a result of ransomware attacks can be listed as follows. WannaCry ransomware infected an estimated 300,000 people in more than 150 countries in 2016 by exploiting a vulnerability in the Windows

[1]*Sorumlu Yazar (Corresponding Author)*

e-mail: atekerek@gazi.edu.tr

operating system. The cost of the WannaCry attack is $4 billion. According to a 2017 report from Verizon, 72% of all healthcare malware attacks in 2017 were ransomware [2]. In 2022, an airport operator suffered a ransomware attack that caused planes to be grounded and flight delays at Zurich International Airport. The attack on the airport operator, which provides air cargo operations and ground handling services, caused the delay of 22 flights [3]. More than $61 million in total ransoms were paid due to Ryuk malware attacks in 2018-2019. Between 2019 and 2020, US hospitals in the United Kingdom and Germany, California, New York, and Oregon were affected by Ryuk malware, resulting in difficulties accessing patient records and even disruption of intensive care [4].

The increase in ransomware attacks has led to the development of detection methods against malicious attacks. Antivirus programs are generally used to protect IT infrastructures from ransomware attacks. Using signature-based approaches in antivirus software cannot detect zero-day and complex malicious attacks [5]. Deep learning methods are artificial deep neural networks that can learn by training with large datasets. In this way, the complex and variable structures of ransomware can be effectively analyzed by deep learning networks. Deep learning provides more sensitive and successful results than signature-based approaches in file features, behavioral analysis, and anomaly detection. Deep learning networks can automatically detect the features of malicious files, and this is more effective than the blacklist creation process. Thus, deep learning provides a more flexible detection ability against constantly updated and evolving threats. In addition, deep learning models can predict unknown or zero-day threats thanks to the diversity of the dataset in the training processes. The use of deep learning models in ransomware detection has the potential to provide security that is more effective and more resistant to future threats compared to traditional methods. Deep learning algorithms are used to analyze malicious file behavior and classify anomalies. In particular, advances in this area can help cybersecurity experts detect ransomware effectively.

Deep learning models such as Convolutional Neural Networks (CNN), Long-Short-Term Memory (LSTM), and trainable feature extraction algorithms are used for ransomware detection. Additionally, deep learning models trained using large-scale datasets can detect ransomware's complex and variable nature. Transfer learning methods effectively detect ransomware threats and better generalize against new variants. Research in this area shows that deep learning methods can be used successfully in ransomware detection. Ransomware attackers constantly update their evasion methods by developing new tactics and attempting to bypass detection systems. Therefore, it is essential that deep learning-based ransomware detection systems can quickly adapt to these variable threats and operate with high accuracy rates. In this study, an LSTM-based experimental study was conducted to detect ransomware attacks on mobile devices.

2. LITERATURE REVIEW

Various ransomware detection models have been proposed in the literature. Zahoora and his team have developed a method that uses deep learning and zero-shot learning to detect ransomware. This method uses an autoencoder network to learn original data features and applies a voting-based ensemble learning technique for detection. However, this method requires obtaining seven dynamic functions in virtual environments such as Cuckoo Sandbox to run ransomware. This makes it challenging to meet early detection requirements and significantly extends the detection time [6]. The EldeRan model is based on the principle that ransomware behaves differently than regular software. In the feature selection of the EldeRan model, the mutual information criterion is used to obtain distinctive features. It detects ransomware using logistic regression, linear SVM, and Bayesian methods. While the detection rate of the EldeRan model is 96.34%, the accuracy rate is 92.19% for SVM and 94.53% for Naïve Bayes [7]. RansHunt is a framework that uses SVM to detect ransomware. RansHunt uses both dynamic and static features of 21 different ransomware families. The dataset includes 360 ransomware, 532 malware types, and 460 benign files. RansHunt uses SVM with normalized polynomial kernel and compares with Naïve Bayes and Decision Trees results. The RansHunt system reached an accuracy rate of 93.5% with the static data set, 96.1% with the dynamic data set, and 97.1% with the hybrid data set [8]. AbdulsalamYa'u et al. used a computer-monitoring deep learning technique to detect anomaly processes that could initiate suspicious movements in the computer's files and directories. The authors proposed a deep neural network algorithm using an autoencoder network to train the deep learning model they developed. Experiments have shown that the model has a classification rate of 99.7% [9]. Guo et al. proposed a visualization-based ransomware detection method. First, they converted the ransomware binaries and harmless programs into grayscale images. Then, the authors extracted image features with the VGG 16 neural network using transfer learning. Finally, they used the SVM machine learning model for classification. The proposed method achieved 96.7% classification accuracy [10]. Moreira et al. converted Portable Executable (PE) header files into color images in a sequential vector pattern to block ransomware attacks. Using the Xception Convolutional Neural Network (CNN) model, the authors applied static analysis to detect ransomware. Two separate data sets were created in the study [11]. Manavi et al. also proposed a ransomware detection method based on the PE header of an executable file and leveraging the LSTM network. As a result, their method achieved an accuracy rate of 93.25% [12]. Using the hybrid analysis technique, A. Gharib and A. Gharbani developed a DNA-Droid tool to detect ransomware. The tool developed by the authors performs static analysis before running and checks whether the application contains ransomware, then terminates the program if signs of ransomware are found. The results showed that this tool achieved high performance, and the false negative rate was below 1.5%, even for unknown ransomware [13]. Bae et al. proposed a method for ransomware detection between malware and trustworthy data. They used six machine learning algorithms to perform their experiments: RF, LR, NB, SGD, KNN, and SVM. The study used a dataset of 900 ransomware files and 300 harmless executables. Regarding ransomware detection, the obtained results reached a good level of accuracy between 97% and 98.65% [14].

Algorithms, attributes, datasets, and data preprocessing steps influence the success of ransomware detection. Complex algorithms, such as deep learning models, generally provide better results but require large amounts of labeled data, and training times can be extended. However, traditional machine learning algorithms can also be practical and require less data, but they often have lower levels of success. Data preprocessing steps can also significantly affect the results. Properly implementing data cleaning, feature selection, dimensionality reduction, and unbalanced class problems can improve the model's performance.

This study proposes a ransomware detection model for Android mobile applications. The study aims to develop an LSTM-based ransomware detection model, considering the latest technological developments in the field. The model developed using LSTM has been proven to have a higher accuracy rate than the methods proposed in previous studies. Since the ransomware dataset used in the study has not been used in any other studies before and is used for the first time in this study, the success of the proposed method cannot be compared to other methods in the literature. However, according to experimental results, the proposed model has achieved a success rate as high as 99%.

3. MATERIAL AND METHOD

Processes are followed to create a model or approach in data science. It is essential to use a similar approach to data mining processes in studies carried out in data science and data mining fields, such as deep learning [15]. The model proposed in this study, in which ransomware attacks on Android mobile devices were audited, is given in Figure 1. Android Ransomware Detection Dataset, available in the Kaggle data library, was used in the study. In the first stage of the pre-processing phase, data distribution analysis was performed. Then, the missing data was analyzed, and the missing data was completed. If some data has incomplete data that cannot be completed, this data is completely deleted. Redundant and repetitive data were removed from the dataset. Outliers were then analyzed to detect deviating data. After data encoding, feature selection was made to use the data in deep learning training. To make the data more stable, it is ready for classification through normalization and standardization processes. According to experimental studies, the best result in the classification process was obtained with the model in which the data was divided into 80% training and 20% testing.

3.1. Dataset

Android Ransomware Detection Dataset includes network monitoring records of Android devices to identify types of ransomware and malicious software operating on the user network. The dataset contains 392,034 rows and 86 columns, and all data has 10 kinds of Android Ransomware and Benign traffic. Ransomware types include SVpeng, PornDroid, Koler, RansomBO, Charger, Simplocker, WannaLocker, Jisut, Lockerpin, and Pletor [16]. The "Unnamed: 0" column in the dataset contains a unique identifier for each data point. The "Flow ID" column contains an identification number that uniquely identifies network flows. The "Source IP" and "Destination IP" columns specify the source and destination IP addresses that are communicating. The "Source Port" and "Destination Port" columns define the communication's source and destination port numbers. The “Protocol” column indicates the protocol type used in network communication and usually represents a numerical value. The most commonly used protocols have numerical codes. These codes are used in network traffic analysis and data sets (for example, 6: TCP, 17:UDP). The "Timestamp" column contains the timestamp of the communication event. The "Flow Duration" column shows the duration of a flow in milliseconds. The "Total Fwd Packets" and "Total Backward Packets" columns contain the forward and reverse packet counts, respectively. The "Total Length of Fwd Packets" and "Total Length of Bwd Packets" columns show the total lengths of forward and reverse packets. The “Fwd Packet Length Max,” “Fwd Packet Length Min,” “Fwd Packet Length Mean,” and “Fwd Packet Length Std” columns represent the statistical properties of forward packet lengths. Similarly, the "Bwd Packet Length Max," "Bwd Packet Length Min," "Bwd Packet Length Mean," and "Bwd Packet Length Std" columns show the statistical properties for backpacker lengths. Other columns include network characteristics such as communication rates, time intervals, flag counts, statistics on packet lengths, flow routings, and event processes. The "Label" column contains label information determining whether each data point is ransomware.

3.2. Data Preprocess

Data preprocessing is the fundamental step in data analytics and machine learning applications, and this stage affects the accuracy of analysis and modeling processes by improving the quality of the dataset used. This process begins with examining the dataset and includes stages such as identifying missing values, cleaning unnecessary information, and organizing the dataset [17]. Correctly handling missing data, identifying outliers, and cleaning unnecessary information ensures reliable results in the analysis and modeling processes of the dataset. Additionally, making the dataset more meaningful and effective through feature engineering is essential to the data preprocessing process. Some data preprocessing steps were applied to the dataset used in the study. Data distribution analysis, understanding the distribution of variables in a data set, determining the central tendency, measures of spread, and shape of the distribution of the variables.

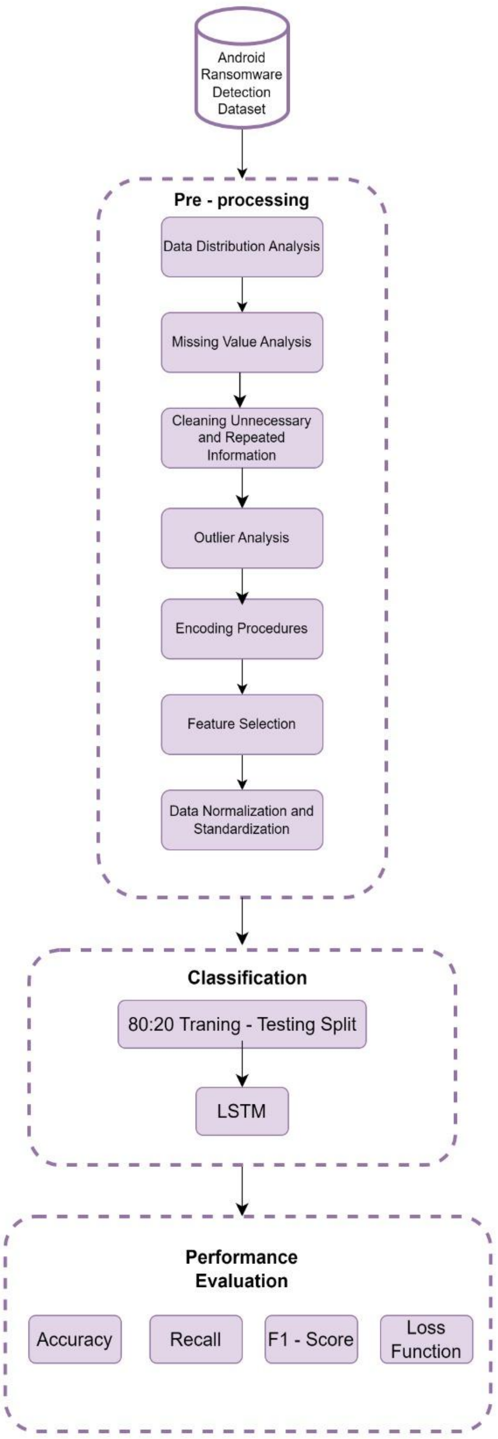

Android Ransomware Detection Dataset
Pre - processing
Data Distribution Analysis
Missing Value Analysis
Cleaning Unnecessary and Repeated Information
Outlier Analysis
Encoding Procedures
Feature Selection
Data Normalization and Standardization
Classification
80:20 Traning - Testing Split
LSTM
Performance Evaluation
Accuracy
Recall
F1 - Score
Loss Function

Figure 1. Proposed Methodology

Understanding the distribution of variables helps determine the correct approach in subsequent modeling processes. The variables' distribution in the study's data set was examined in this step, and how close they were to the normal distribution was evaluated. Examining and processing missing data is fundamental in data science and analysis processes. It helps obtain more reliable results by improving the dataset's quality when implemented correctly. Missing data is when empty or null values represent some observations or variables in a data set. When the data set used in the study was checked with the null function, no action was taken in this step since there were no missing values. The dataset is organized and optimized by deleting unnecessary and duplicate data. The study checked duplicate lines in the dataset with Pandas' duplicated() function. Since there are no repeating rows, no action was taken in this step. Outlier analysis is a method used to identify and understand abnormal or unexpected values in the dataset (Figure 2). Outliers are observations that generally do not follow a normal distribution, fall outside statistical criteria, or deviate significantly from other observations [18]. These values can mislead analysis and modeling processes, so identifying outliers is essential for developing a successful AI model.

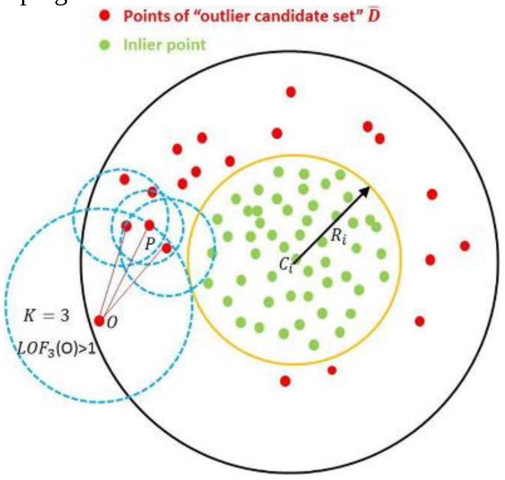


Figure 2. Outlier Analysis [19]

This study used two outlier analysis methods: IQR (Interquartile Range) and Z-Score. The two techniques were applied separately, and their effects on the model's success were compared during the model evaluation. According to the results obtained, the Z-score is more successful.

3.2.1. IQR

The dataset has four parts: Q1, Q2 (median), Q3 and Q4. The first quartile (Q1) represents the range from the smallest value to the median of the dataset. The median (Q2) divides the dataset into two equal parts and represents half of the total dataset. The third quartile (Q3) represents the first 75% of the dataset, covering the range from the median to the maximum value. The last quarter (Q4) refers to the entire dataset. That is, it represents the most significant value [20]. The term that describes the difference between the 75th and 25th percentiles of the data set is called the interquartile range. In other words, the Q1 and Q2 interquartile width represents the middle 50% of the data [21]. According to the IQR method, values 1.5 times less than the 25% quartile value and 1.5 times more than the 75% quartile value are classified as outliers.

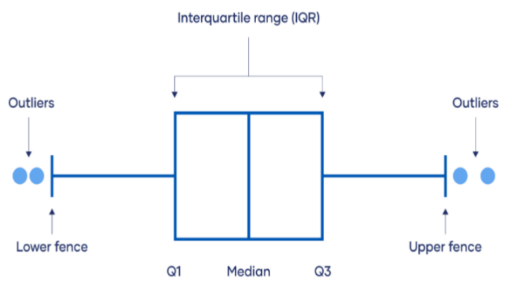


Figure 3. IQR box plot [22]

The vertical lines in the box plot (Figure 3) show Q1, median, and Q3. The lines at the ends show the highest and lowest values. The width of the box indicates the interquartile range. A small width means it has less dispersion, while a large width means it has more dispersion.

3.2.2. Z-Score

Z-score is a measure that expresses how far a value is from the mean in terms of standard deviation [23]. The Z-score indicates the position of a data point within the overall distribution.

μ= Mean
σ= Standard Deviation

$$z = \frac{x - \mu}{\sigma} \qquad (1)$$

The Z-score calculates how many standard deviations a value is from the mean [23]. This criterion, widely used in outlier definitions, is usually a limit set as $|z|>3$. If a value's z-score is more significant or less than 3, it is an outlier. This study applied suppression and deletion processes for outliers determined by IQR and z-score methods. In the suppression method, a minimum value is assigned to values that are less than the minimum value determined as the outlier limit. In contrast, a maximum value is assigned to values that are larger than the maximum value determined as the outlier limit. In particular, since the statistical measures of outliers can be misleading, it is aimed to make these measures more representative by suppressing these values. By deleting outliers, the statistical criteria in the data set are made more reliable, and the accuracy of the analyses is increased. Outliers can mislead the mean and standard deviation, so deleting outliers helps make the dataset more reliable. Both methods have advantages as well as some disadvantages. It was seen that the deletion method gave better results when applied to the data set separately, and it was decided to use this method in the final version of the model.

3.2.3. Encoding

Categorical variables can contain non-numeric categorical values. Such variables are converted to categorical numeric format. Because non-numeric categorical variables are not understood by algorithms [24]. Encoding operations are the methods used to perform these transformations. Commonly used encoding methods are one-hot, label, and race. The label encoding method assigns a unique number to each category. It is a preferred method, especially when working with ordered categorical variables. The model may misunderstand the category order when used on unordered categorical variables. In the one-hot encoding method, each category is represented by creating a new column, and each data point receives a value of 1 only in the column of the relevant category and a value of 0 in the other columns. In this way, each category is represented wholly and independently. One-hot encoding is especially suitable for working with unordered categorical variables, and thanks to this method, the model learns categorical information effectively. This study applied the one-hot encoding method to categorical variables in the data set. The one-hot encoding method was applied to all categorical variables except the target variable with the help of a function. After distinguishing between dependent and independent variables, a one-hot encoding process was applied to the target variable.

3.2.4. Feature selection

Feature selection is the process of selecting a subset of the features in a data set or determining the weight of the features based on specific criteria. This process is carried out to improve the model's performance, eliminate unnecessary or low-impact features, or reduce the computational cost [25]. Feature selection makes the model more straightforward and effective and helps prevent overfitting by reducing complexity [26]. There are different methods for feature selection. The chi-square method was preferred within the scope of the study. This test helps purify the model from unnecessary features, ensuring that the model is more straightforward, more meaningful, and has higher generalization ability. When the chi-square test was applied for each feature, the p-value was used to determine which features would be more effective in the modeling process. Features with low p-values significantly connect with the target variable and are added to the selected features. The p-value is evaluated based on a specified significance level (alpha level). Since the significance level is generally taken as 0.05 in the studies, this value was used first in the project. Then, the same processes were repeated with different values, and the results were compared.

3.2.5. Normalization and standardization

Normalization is scaling feature values to a range between 0 and 1. Standardization is accomplished by subtracting feature values from the mean and dividing by the standard deviation. Standardization makes feature values distributed more like a normal distribution. Standardization makes the model more resilient to outliers and large feature values. Both methods may be preferred depending on the structure of the dataset and the model used. Normalization and standardization increase model performance and contribute to faster and more effective operation of optimization algorithms. Therefore, using these techniques in the data preprocessing stage is essential to obtain more reliable and consistent results.

The standardization process centers feature values around the mean and scales them by standard deviation units. In this way, feature values are distributed more similarly, and the model becomes more effective in training. Many data analysis libraries, such as a scikit-learn library, provide ready-made functions for the standardization process. MinMax Scaling is a type of data normalization that refers to scaling feature values to a specific range. This method generally scales between 0 and 1 [27]. MinMax Scaling rescales property values based on minimum and maximum values. So, all property values fit within a specific range. The scikit-learn library in Python was used for MinMax Scaling in the project. Its formula is shown in Equation 2.

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (2)$$

In the study, both the standardization process and the MinMax Scaling process were applied. Observations have shown that the MinMax Scaling process gives better results.

3.3. Classification

The data set was divided in this study step, and the model to be used was determined.

3.3.1. Partitioning the dataset

Dividing the dataset is essential to detect and prevent problems of overfitting and underfitting, measure the model's overall performance, and obtain reliable results. Dividing the data set ensures the model can work accurately on real-world data and produce reliable results [28]. The method to be chosen may vary depending on the project's characteristics, the data set's size, and the problem type. A good dataset split ensures the model is reliable, generalizing, and can work successfully on real-world data. Within the scope of the study, the data set was divided into different proportions and tested. These divisions include various ratios such as 75% training, 25% testing, 70% training, 30% testing, and 80% training, 20% testing. As a result of the experiments, the most effective results were obtained in the 80% training and 20% test section.

3.3.2. Model

This step involves determining the most appropriate model according to the study's aims and the dataset's characteristics. When deciding between different model types, the dataset's structure and the problem type are considered. The model selection process includes trial and error methods and aims to select the model that best suits the project's requirements. Performance metrics are determined, and the performance of the selected models is evaluated using these metrics. The model's overall performance is evaluated using tuning hyperparameters and cross-validation techniques.

LSTM models provide significant advantages thanks to their ability to effectively maintain long-term connections, successfully learn patterns that change over time, and remember important information through memory cells. Since these features increase the effectiveness of LSTM and allow it to be preferred in various application areas, especially in areas where dynamic and hidden patterns need to be detected, such as security threats, it was chosen to use LSTM neural network, which is one of the deep learning methods, in the study. First, the data was suitable for LSTM, and the time series window size was determined. Different layer numbers were tried by trial and error, and the model that gave the best results was selected. As a result, a 5-layer LSTM model was used. LSTM is a recurrent neural network (RNN) cell for time series analysis, natural language processing, and other sequential data problems. Unlike traditional neural networks, LSTM includes feedback connections that can process entire data sequences and individual data points. This unique cell structure was explicitly developed to manage information storage and transfer [29].

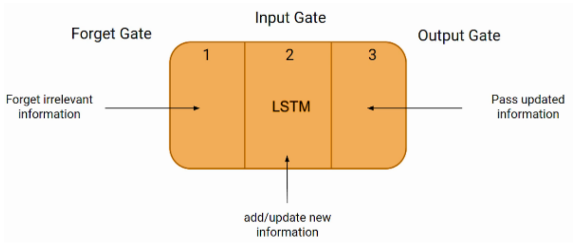


Figure 4. LSTM cell [30]

The LSTM cell shown in Figure 4 consists of three essential components: forget gates, input gates, and output gates. These components perform different functions. The cell contains two states, Cell State and Hidden State. These states are constantly updated and carry past information to current time steps. The cell state represents long-term memory, while the latent state represents short-term memory. Each row of LSTM cells in each layer is fed with the output of the previous cell. This allows the cell to retrieve previous entries and sequence information. The following steps occur in each LSTM cell:

- The forgetting gate is calculated.
- The entrance door value is calculated.
- Cell status is updated using the above two outputs.
- The output (hidden state) is calculated using the output gate [30].

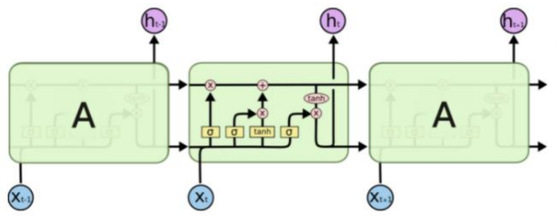


Figure 5. LSTM architecture [30]

Forget gates determine the information to be deleted from the cell state. This is used to clear the cell state when historical information is no longer needed [30]. LSTM architecture is given in Figure 5.

$$f_t = \sigma(x_t * U_f + H_{t-1} * W_f) \qquad (3)$$

Equation 3 In the forgetting gate equation,

X_t = Input current timestamp
U = Input weight
H_{t-1} = Hidden state of the previous timestamp
W_f = Weight matrix for hidden state [30].

Then, the Sigmoid activation function is applied to determine which information will be forgotten [30]. Entry gates control the addition of new information. A candidate value processed by the sigmoid activation function and scaled by the tanh activation function is checked by this gate before being added to the cell state.

$$i_t = \sigma(x_t * U_i + H_{t-1} * W_i) \qquad (4)$$

Eşitlik 4'deki Giriş kapısı denkleminde;

Xt = Input at current timestamp t
Uİ = Input weight matrix
Ht-1 = Hidden state of the previous timestamp
Wt = Hidden state weight matrix [30].

Output gates determine the information that emerges from the cell state. The tanh function normalizes the cell state processed by the sigmoid function and then goes through the control of the output gate [30]. The equation of the exit door, which is very similar to the previous two doors, is given in equation 5.

$$o_t = \sigma(x_t * U_o + H_{t-1} * W_o) \qquad (5)$$

Cells are a type of memory that saves information. This memory contains essential information from the past and is updated over time [30]. The equation of the cell state is given in Equation 6.

$$C_t = f_t * C_{t-1}, i_t * \bar{C}t \qquad (6)$$

3.4. Performance Evaluation

Performance evaluation is carried out with test data and is tested to determine the extent to which the created model meets the specified success criteria. Depending on the values obtained, the study can be terminated or returned to the first stage to review all stages. Evaluation with test data includes performance measures of the model such as accuracy, sensitivity, specificity, and sensitivity [31]. Evaluation helps check that the model avoids overfitting the training data; that is, it does not make learnings that are too specific to the training data. It is also essential to evaluate the generalization ability of the model to understand how it can deal with new and unknown data. Evaluation with test data is also used to identify potential errors or weaknesses in the model's performance. Understanding these errors guides developing and improving the model. Test data increases the model's reliability in application by determining how effective the model is in real-world scenarios. In this study, accuracy rate (Accuracy), sensitivity (Recall), loss function (Loss Function), and F1 score performance metrics were used to evaluate the classification success of the proposed model. A complexity matrix (Confusion Matrix) was created to calculate these performance metrics. In terms of performance metrics, metrics such as accuracy, sensitivity, recall, and F1 score are used to evaluate the performance of deep learning-based ransomware detection systems. Factors such as timing and resource utilization are also crucial because providing real-time protection and keeping the false alarm rate low is critical to minimizing risk.

3.4.1. Confusion matrix

The confusion matrix is a type of matrix used to evaluate the model's performance in classification problems. This matrix helps visualize different performance metrics by comparing the classification predictions made by a model to their actual classes.

The confusion matrix consists of four basic categories: True Positives (TP), the number of samples the model classifies as true positives; true Negatives (TN), the number of samples the model classifies as true negatives; false Positives (FP), the number of samples the model classifies as positive when they are harmful, and False Negatives (FN) refer to the number of samples that the model classifies as negative but positive.

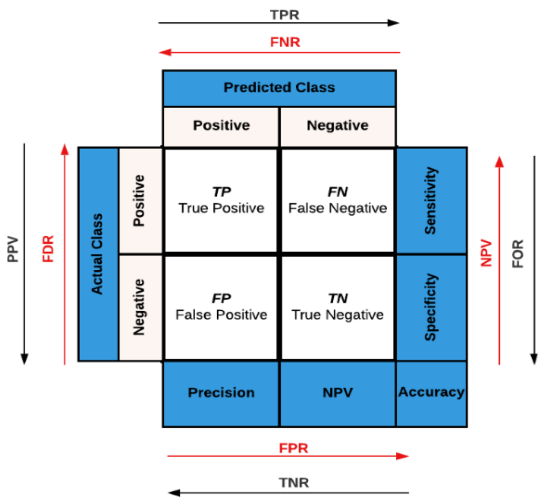


Figure 6. Confusion matrix [32]

3.4.2. Accuracy

Accuracy is a performance metric that measures how accurately a classification model makes predictions. As seen in Equation 7, the accuracy rate expresses the ratio of correctly predicted samples to the total number of samples.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \qquad (7)$$

The accuracy metric is usually expressed as a percentage. A high accuracy rate means the model classifies the data correctly, but more than the accuracy rate is required to reflect the model's performance fully. The accuracy rate can be misleading, mainly if the dataset consists of unbalanced classes, that is, if there are significant differences in the number of samples between classes. Other performance metrics and evaluation methods, such as classification matrix, should also be considered in this case.

3.4.3. Recall

Sensitivity evaluates the performance of classification models, measuring the ability to correctly classify all truly positive examples as positive. Sensitivity is especially considered when detecting a class, as it is essential and the potential cost of false negatives is high. In Equation 8, sensitivity expresses the ratio of true positives to the sum of total true positives and false negatives.

$$Recall = \frac{TP}{TP+FN} \qquad (8)$$

Sensitivity indicates that the model recognizes truly positive examples and effectively identifies examples from this class.

3.4.4. Loss function

The loss function is a performance metric showing the difference between the model's actual and predicted outputs. The purpose of the loss function is to guide and optimize the learning process of the model. It produces a score that measures how close the model is to its targeted output. The choice of loss function significantly impacts the model's learning process and generalization ability. Choosing a correct loss function can help the model learn desired behaviors and generalize. For this reason, choosing the appropriate loss function for a specific problem and minimizing this function can increase the model's performance.

3.4.5. F1 score

The F1 score is the metric that balances the accuracy, precision, and sensitivity performance measurements of the classification model. It is used to evaluate the model's performance more comprehensively, especially in data sets with unbalanced classes. The F1 score formula is shown in Equation 9.

$$F_1 = \frac{2TP}{2TP+FN+FP} \qquad (9)$$

The F1 score is used to evaluate the model's overall performance with a single metric in classification problems, especially when there is an imbalance in the number of samples between classes. F1 score takes values between 0 and 1. 1 represents the best performance, and 0 represents the worst performance. A high F1 score indicates the model's ability to minimize false positives and negatives. Therefore, it is an important measure that shows the model's overall performance. The main reason why F1 Score is preferred over accuracy is the ability to evaluate model performance more balancedly in unevenly distributed datasets. Additionally, F1 Score is used in cases that cover all error types, not just False Negative or False Positive errors [32].

4. RESULT and DISCUSSION

In this study, the LSTM neural network, one of the deep learning methods, was preferred. After model training was completed, classification success was evaluated with the test dataset.

In this study, different LSTM models were created and tested according to various features, and the most effective model was selected. The selected model contains five repeating LSTM layers. The first LSTM layer has 256 units and produces output for consecutive time series. It also includes a 20% Dropout layer to reduce the risk of overfitting. The second layer has the same characteristics as the 512 LSTM unit. The third and fourth layers have 256 and 512 LSTM units operating on sequential time series. The fifth and final layer contains 256 LSTM units. The output layer of the model is designed to solve a multi-class problem, and the 'softmax' activation function is used in this layer. Once the model is created, parameters are selected to train the model. In the model step, 'Adam' was chosen as the optimizer because Adam is an effective optimization algorithm that combines the adaptive momentum method and the root mean square error (RMSprop) method. This algorithm can speed up the training process and allow the model to learn faster and more effectively. Additionally, 'categorical_crossentropy' was chosen as the loss function because it was stated that a multi-class classification problem was addressed in the project. This loss function measures the differences between multiple classes and encourages the model to perform correct classification. Finally, 'accuracy' was chosen as the metric because this metric measures the accuracy of the model and is stated as one of the project's success criteria.

Different epoch values were tested for the proposed model to learn the patterns in the data set and improve its generalization ability, and training was performed with 100 epochs. Again, different batch sizes were tried, and the batch size was 64 in the final model. These choices were made considering the project's specific requirements and the dataset's characteristics, thus ensuring the model was effectively trained and its performance evaluated. Evaluation of the F1 score, accuracy, and sensitivity success of the created model is shown in Table 1.

Table 1. Model Performance Results

Accuracy	**Recall**	**F1 Score**
%99.3	0.99	0.99

A comparison of the results of this study with other studies in the literature is presented in Table 2. Since the dataset used in this study has yet to be used in any previous study, the comparison in Table 2 was made with studies using different datasets.

Table 2. Comparison of studies

Study	**Dataset**	**Methods**	**Results (Accuracy)**
Zahoora et al.[6]	VirusShare4	Zero-shot Learning (ZSL)	%92
Sgandurra et al. [7]	Unpublished Dataset	Lojistic Regression	%96,34
Hasan, M.; Rahman, M. RansHunt [8]	VirusShare	SVM	Statik dataset: %93,5, Dinamic Dataset: %96,1, Hibrit Dataset: %97,1
AbdulsalamYa'u et al. [9]	Resilient Information Security System (RISS)	Deep Neural Network	%99.7
Guo et al. [10]	Unpublished Dataset	SVM	%96.7
Bae et al. [14]	Unpublished Dataset	RF, LR, NB, SGD, KNN and SVM	%98.65
Manavi, F., Hamzeh, A [12]	Unpublished Dataset	LSTM	%93
Moreira et al. [11]	Unpublished Dataset	Random Forest	%93.73
Proposed Study	Android Ransomware Detection [16]	LSTM	%99.3

The dataset contains 392034 rows and 85 columns and 10 types of Android Ransomware and Malicious traffic in all types. After the data preprocessing process, 292753 data remained. The types of Ransomware include SVpeng, PornDroid, Koler, RansomBO, Charger, Simplocker, WannaLocker, Jisut, Lockerpin, and Pletor and Benign. The confusion matrix only shows test results. The confusion matrix is shown in Figure 7.

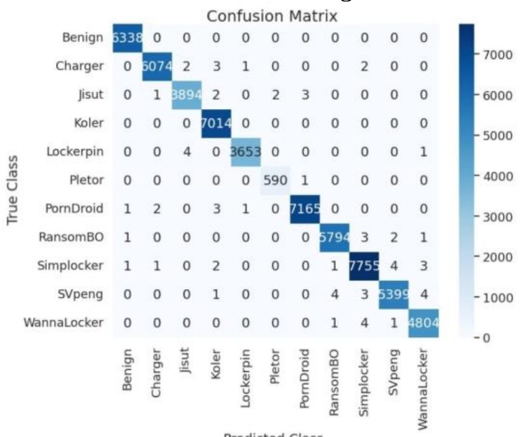


Figure 7. Confusion matrix of the proposed model

The preferred loss function in the project is the Categorical cross-entropy loss function, which is suitable for a multi-class classification task. Categorical Cross entropy is a loss function that measures the difference between the classes predicted by the model and the actual labels. It calculates how accurate the model's predictions are and how many errors it makes by evaluating the difference in the probability distribution for each class. Categorical Crossentropy also shows optimal performance when used with the softmax activation function. This combination transforms the model's outputs into probability distributions, and the loss function evaluates how well these probabilities match the actual labels. As a result of training with 100 epochs, the loss value, accuracy, and val_loss values for the last 10 epochs are shown in Table 3.

Table 3. Loss Function Evaluation of the Model

Epoch	**Loss**	**Accuracy**	**Loss**
90	0.018	0.9944	0.01
91	0.017	0.9948	0.0096
92	0.017	0.9949	0.0089
93	0.017	0.9949	0.0099
94	0.017	0.9948	0.0102
95	0.018	0.9943	0.017
96	0.017	0.9946	0.0123
97	0.016	0.9949	0.0093
98	0.016	0.9949	0.008
99	0.017	0.9947	0.015
100	0.016	0.9949	0.0082

The difference between loss and loss values shows how well the model fits the training dataset and performs its generalization ability on the validation dataset. Therefore, loss is generally the values that reflect training performance, and val_loss is the value that reflects validation performance. A good model should achieve low loss during training and exhibit good generalization ability on the validation set.

The fact that the val_loss value is lower than the loss value (the better the val_loss value is) indicates that this model can successfully generalize the patterns from the training dataset to the test dataset. This indicates that the model is trying to better fit real-world data without overfitting. The accuracy graph is the graph that shows the accuracy values on the training set and validation set during the training of the model. The loss graph is the graph that shows the loss values on the training set and validation set during the training of the model. These graphs are used to understand how well the model has learned and gained generalization ability.

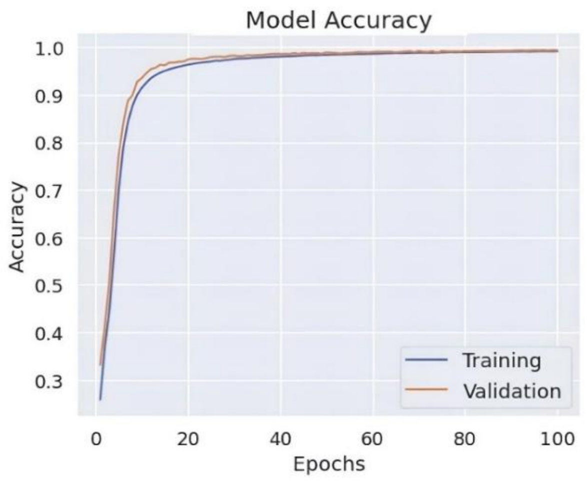


Figure 8. Accuracy graph of the model

The accuracy graph of the model created in the study is shown in Figure 8.

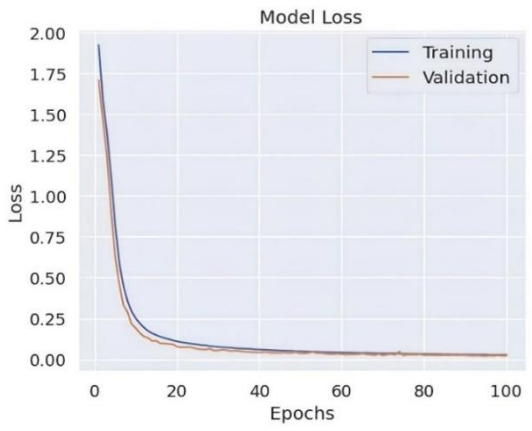


Figure 9. Loss graph of the model

The loss graph of the model created in the study is shown in Figure 9.

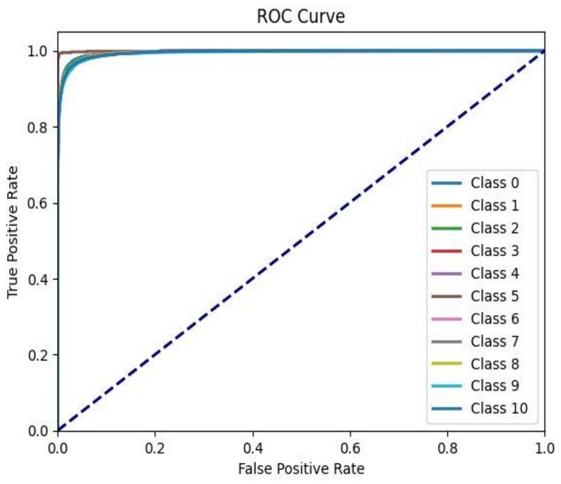


Figure 10. ROC curve of model

The ROC curve of the model created in the study is shown in Figure 10.

5. CONCLUSION

In this study, ransomware attacks on mobile devices were audited with the Android Ransomware Detection dataset using the LSTM neural network, one of the deep learning methods. The proposed model was trained on a complex dataset containing ten different classes, and the F1 Score, accuracy, and sensitivity metric values show that the model successfully classifies ransomware types. In addition, low loss values in the training set of the model indicate that the learned patterns are comprehended and the model adapts appropriately to the training data. The low loss values obtained in the test data set indicate that the model's generalization ability is strong and can perform effectively for new data sets. Experimental studies show that deep learning techniques have an important role in cyber security and can be used in critical tasks such as detecting malicious software such as ransomware that threatens mobile applications. Future studies may further expand the potential in this field by including in-depth research in areas such as different deep learning architectures, larger data sets, and integrating different features into models. This study reveals that deep learning models have significant potential in terms of cyber security.

ACKNOWLEDGEMENT

TÜBİTAK supports this work under grant number 1919B012303087.

DECLARATION OF ETHICAL STANDARDS

There is no need to obtain permission from the ethics committee for the article prepared. The article prepared has no conflict of interest with any person/institution.

AUTHORS' CONTRIBUTIONS

Hatice KARACA: Methodology, software, performed the experiments and analysis of the results, validation, visualization, writing-original draft preparation.

Adem TEKEREK: Methodology, analysis of the results, supervision.

CONFLICT OF INTEREST

There is no conflict of interest in this study.

REFERENCES

[1] Teymourlouei, H., "Preventative measures in cyber & ransomware attacks for home & small businesses' data", ***Proceedings of the International Conference on Scientific Computing (CSC)***, 87–93 (2018).

[2] Verizon. Data Breach Investigations Report. (2017).

[3] Ransomware Attacks on European Transportation Targets, ***I-HLS***, (2022).

[4] Barry, Ellen; Perlroth, Nicole "Patients of a Vermont Hospital Are Left 'in the Dark' After a Cyberattack". ***New York Times***, (2020).

[5] Masdari, Mohammad, and Hemn Khezri. "A survey and taxonomy of the fuzzy signature-based intrusion detection systems." ***Applied Soft Computing*** 92 (2020).

[6] Zahoora, Umme, et al. "Zero-day ransomware attack detection using deep contractive autoencoder and voting based ensemble classifier." ***Applied Intelligence*** 52.12 (2022).

[7] Sgandurra, Daniele, et al. "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection." ***arXiv preprint*** (2016).

[8] Hasan, Md Mahbub, and Md Mahbubur Rahman. "RansHunt: A support vector machines based ransomware analysis framework with integrated feature set." ***2017 20th international conference of computer and information technology (ICCIT)***. IEEE, (2017).

[9] AbdulsalamYa'u, Gital, et al. "Deep learning for detecting ransomware in edge computing devices based on autoencoder classifier." ***2019 4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)***. IEEE, (2019).

[10] Chen, C.-Q., Cuo, C., Shen, G.-W.: "A ransomware classification method based on visualization", ***Netinfo Security***. 20(4), 31–39, (2020).

[11] Moreira, Caio C., Davi C. Moreira, and Claudomiro de S. de Sales Jr. "Improving ransomware detection based on portable executable header using xception convolutional neural network." ***Computers & Security*** 130, 103265, (2023).

[12] Manavi, Farnoush, and Ali Hamzeh. "Static detection of ransomware using LSTM network and PE header." ***2021 26th international computer conference, Computer Society of Iran (CSICC). IEEE***, (2021).

[13] Gharib, Amirhossein, and Ali Ghorbani. "Dna-droid: A real-time android ransomware detection framework." ***Network and System Security: 11th International Conference, NSS 2017, Helsinki, Finland, August 21–23, 2017, Proceedings 11. Springer International Publishing***, (2017).

[14] Bae, Seong Il, Gyu Bin Lee, and Eul Gyu Im. "Ransomware detection using machine learning algorithms." ***Concurrency and Computation: Practice and Experience*** 32.18 (2020).

[15] Mansyur, M., Indra Budi, and Yova Ruldeviyani. "Utilization of Data Mining Classification Technique for Civil Servant Mutation Pattern: A Case Study of Pangkajene and Kepulauan District Government." ***2018 International Conference on Applied Information Technology and Innovation (ICAITI)***. IEEE, (2018).

[16] Internet: "Android Ransomware Detection", https://www.kaggle.com/datasets/subhajournal/android-ransomware-detection, (2024).

[17] Agarwal, V., "Research on data preprocessing and categorization technique for smartphone review analysis", ***International Journal of Computer Applications***, 131(4), 30-36, (2015).

[18] Modi, Krishna, and Bhavesh Oza. "Outlier analysis approaches in data mining." ***International Journal of Innovative Research in Technology***, 3(7), 6-12, (2016).

[19] Liu, J., Cao, Y., Li, Y., Guo, Y., & Deng, W., "Analysis and prediction of power distribution network loss based on machine learning", ***International Journal of Numerical Modelling: Electronic Networks, Devices and Fields***, 36(4), (2023).

[20] Singh, N., & Oorkavalan, U. (2018). "Triple Threshold Statistical Detection filter for removing high density random-valued impulse noise in images", ***EURASIP Journal on Image and Video Processing***, 1-16, (2018).

[21] Perez, H., & Tah, J. H. M. "Improving the Accuracy of Convolutional Neural Networks by Identifying and Removing Outlier Images in Datasets Using t-SNE", ***Mathematics***, 8, 662, (2020).

[22] Whaley III, "Dewey Lonzo. The interquartile range: Theory and estimation", ***MS thesis. East Tennessee State University***, (2005).

[23] Anggoro, D. A., & Supriyanti, W., "Improving accuracy by applying Z-score normalization in linear regression and polynomial regression model for real estate data", ***International Journal of Emerging Trends in Engineering Research***, 7(11), 549-555, (2019).

[24] Nurnoby, M. Faisal, and El-Sayed M. El-Alfy. "Overview and Case Study for Ransomware Classification Using Deep Neural Network." ***2019 2nd IEEE Middle East and***

North Africa COMMunications Conference (MENACOMM). IEEE, (2019).

[25] Li, Zhida, Ana Laura Gonzalez Rios, and Ljiljana Trajković. "Machine learning for detecting the WestRock ransomware attack using BGP routing records." ***IEEE Communications Magazine***, 61(3), 20-26, (2022).

[26] Anusha, Peruri Venkata, et al. "Detecting outliers in high dimensional data sets using Z-score methodology", ***International Journal of Innovative Technology and Exploring Engineering*** 9.1, 48-53, (2019).

[27] Singh, Amardeep, et al. "Enhancing ransomware attack detection using transfer learning and deep learning ensemble models on cloud-encrypted data." ***Electronics***, 12.18, 3899, (2023).

[28] Kahloot, Khalid M., and Peter Ekler. "Algorithmic splitting: A method for dataset preparation." ***IEEE Access***, 9, 125229-125237, (2021).

[29] Homayoun, Sajad, et al. "DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer." ***Future Generation Computer Systems***, 90, 94-104, (2019).

[30] Saxena, Shipra. "Introduction to long short term memory (LSTM)." ***Analytics Vidhya*** (2021).

[31] Ciaramella, Giovanni, et al. "Explainable ransomware detection with deep learning techniques." ***Journal of Computer Virology and Hacking Techniques*** 20(2), 317-330, (2024).

[32] Almomani, I., Alkhayer, A., & El-Shafai, W., "E2E-RDS: Efficient End-to-End ransomware detection system based on Static-Based ML and Vision-Based DL approaches". ***Sensors***, 23(9), 4467, (2023).