computers

MDPI

Article

Proactive Ransomware Detection Using Extremely Fast Decision Tree (EFDT) Algorithm: A Case Study

Ibrahim Ba'abbad * and Omar Batarfi *

Department of Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
* Correspondence: eabad@stu.kau.edu.sa (I.B.); obatarfi@kau.edu.sa (O.B.)

Abstract: Several malware variants have attacked systems and data over time. Ransomware is among the most harmful malware since it causes huge losses. In order to get a ransom, ransomware is software that locks the victim's machine or encrypts his personal information. Numerous research has been conducted to stop and quickly recognize ransomware attacks. For proactive forecasting, artificial intelligence (AI) techniques are used. Traditional machine learning/deep learning (ML/DL) techniques, however, take a lot of time and decrease the accuracy and latency performance of network monitoring. In this study, we utilized the Hoeffding trees classifier as one of the stream data mining classification techniques to detect and prevent ransomware attacks. Three Hoeffding trees classifier algorithms are selected to be applied to the Resilient Information Systems Security (RISS) research group dataset. After configuration, Massive Online Analysis (MOA) software is utilized as a testing framework. The results of Hoeffding tree classifier algorithms are then assessed to choose the enhanced model with the highest accuracy and latency performance. In conclusion, the 99.41% classification accuracy was the highest result achieved by the EFDT algorithm in 66 ms.

Keywords: ransomware attacks; cyber threat hunting; proactive approach; Hoeffding trees classifier; classification accuracy; classification latency

check for updates

Citation: Ba'abbad, I.; Batarfi, O. Proactive Ransomware Detection Using Extremely Fast Decision Tree (EFDT) Algorithm: A Case Study. *Computers* **2023**, *12*, 121. https://doi.org/10.3390/computers12060121

Academic Editor: Leandros Maglaras

Received: 11 March 2023
Revised: 3 June 2023
Accepted: 6 June 2023
Published: 15 June 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The rapid growth and widespread internet use have transformed the business landscape, providing numerous benefits to businesses operating in various sectors. Utilizing the Internet as a remote digital business environment offers services such as resource management, operations analysis, information retrieval, and business forecasting [1]. However, alongside these advantages, there is an ever-increasing concern within the Internet security research community regarding the security of online business environments.

Cybercriminals continuously evolve their tactics and develop new forms of malicious software to target and exploit legitimate businesses. Ransomware stands out among the various cyber threats because of the severe consequences it causes for its victims. Ransomware assaults can wreck an association's assets, prompting huge financial misfortunes and reputational harm. Ransomware's global spread and the development of "Ransomware as a Service" (RaaS) have increased the risks associated with these attacks even further [1].

In recent years, ransomware attacks have stood out due to their troublesome nature and their difficulties in recuperation. Most of the time, ransomware locks a victim's computer and encrypts their sensitive data, with the attackers requesting a ransom in exchange for the decryption key [1]. Therefore, ransomware can be isolated into two primary classifications: locker ransomware, which locks the setback's framework, and crypto-ransomware, which scrambles their data. Crypto ransomware has become especially famous due to the challenges in recuperating encoded information [1].

Researchers have explored how to utilize encryption to shield delicate information from ransomware attacks and make it safer. One choice is a mixture encryption framework joining symmetric and asymmetric encryption. The casualty's information is scrambled

utilizing vigorous block symmetric encryption strategies, while the symmetric key utilized for encryption is itself encrypted utilizing the beneficiary's public key. Only the associated private key can be used to decrypt the encrypted data [1].

More than 64% of ransomware attacks originate from electronic mail, making it the most common infection source. Attackers often send phishing emails containing malicious attachments or links to unsuspecting victims. The ransomware is activated when the victim interacts with these emails and unknowingly provides their system details to the Command and Control (C & C) server to obtain the encryption key. The victim's files are then encrypted and presented with on-screen messages demanding a ransom payment in cryptocurrencies such as Bitcoin [1].

Researchers have conducted numerous investigations to proactively identify remedies for detecting and preventing ransomware attacks. Proactive monitoring using artificial intelligence (AI) techniques has emerged as an effective approach for proactive forecasting and timely recognition of ransomware attacks [2]. However, traditional machine learning (ML) algorithms used for feature extraction can be time-consuming and hamper the proactivity of network monitoring [3]. This has led to a limited exploration of deep learning (DL) techniques, despite their potential to improve accuracy and latency performance [4].

To achieve the highest levels of accuracy and latency performance in proactive network monitoring, novel strategies need to be developed [4]. Stream data mining techniques provide a promising avenue for efficiently processing massive amounts of data. Unlike batch processing, stream data mining requires only a single pass of the data, enabling real-time processing and storage of statistical information in main memory [5]. Data streams can be classified into offline streams, which arrive in periodic bulks, and online streams, which are processed one at a time and include real-time data from sensors and online monitoring systems.

This study focuses on utilizing stream data mining techniques to detect and prevent ransomware attacks. Specifically, we employ the Hoeffding trees classifier as one of the tree-based classification techniques for stream data mining. We test three Hoeffding trees classifier algorithms: the VFDT algorithm, the random Hoeffding trees algorithm, and the EFDT algorithm.

Banks, government agencies, and industrial factories are examples of entities that stand to gain the most from improved ransomware detection and prevention models. These organizations often handle sensitive and critical data, making them prime targets for ransomware attacks. Therefore, implementing proactive monitoring systems that can protect user data and enhance system security is paramount.

In particular, the research contribution can be listed in the following points:

- Analyzing six articles that involve proactive monitoring models to detect and prevent ransomware attacks;
- Determining Hoeffding trees classifier as one of the tree-based classification stream data mining techniques;
- Applying three Hoeffding trees classifier algorithms to the RISS research group ransomware dataset using MOA software as a testing framework;
- EFDT algorithm achieves 99.41% classification accuracy as the highest result at the highest latency performance of 66 ms.

The organization of this article is as follows: in Section 2, the research gap in the field is discovered by analyzing recent efforts to develop proactive ransomware detection methods. The research methodology, which includes the general model block diagram, is described in Section 3. In Section 4, along with a description of the testing environment, the proposed model for the Hoeffding trees classifier as one of the stream data mining classification techniques is presented. After acquiring the model's testing results, a discussion is offered in Section 5 to further compare current methods with the suggested model. In Section 6, the paper is summarized along with some recommendations for future research. Appendix A shows the screenshots of the testing phase of the Hoeffding trees classifier used including configuration and results using MOA tool.

2. Related Work

Network monitoring is one solution that is crucial to intrusion detection in general. System administrators should continue to monitor network activity for unusual activity. The goal of passive attacks is to secretly gather the system information of the victim in order to carry out an active attack with more damage. Network monitoring does, however, help in both passive and active attack prevention. Numerous connection factors, such as but not limited to geographical data, connection time, and connection count, are tracked and compared with the values predicted for those factors. A series of proactive preventative instructions are carried out if it is discovered [2].

Since network edge computing uses network edge rather than conventional remote servers, it increases the I/O operations rate between network users and resources. Devices used for edge computing, however, have security flaws. Ransomware is one of the most common forms of malware in the concepts of network edge computing and IoT. Without the requirement for the feature extraction procedure, Abdulsalam et al. [3] suggested a deep neural network model with an autoencoder to monitor the behaviors of the ransomware. 942 benign samples and 582 ransomware samples are used from an open-source dataset. MATLAB software is employed as a simulation tool. The proposed model's true positive rate is 99.7%. De-noising autoencoder based on dropout training will be implemented as an alternative training method in future work to improve the model's accuracy to detect more than ransomware attacks.

Detecting encrypted ransomware attacks through secure socket layer/transport layer security (SSL/TLS) protocol is a crucial task. Zhang et al. in [6] proposed a system based on dual generative adversarial networks (GAN) for encrypted ransomware detection. The system is named a transferred-generating adversarial network intrusion detection system (TGAN-IDS). Deep convolutional GAN (DCGAN) and transferred GAN (TGAN) are the two primary parts of TGAN-IDS. DCGAN is based on a convolutional neural network (CNN) deep learning algorithm. DCGAN optimizes the performance of its generator (G_D) while TGAN is used to optimize its discriminator (D_T) performance. TGAN is the core component of the system since its discriminator (D_T) reacts as the final detector of anomaly behavior. Three websites—Malware Traffic Analysis.net, PacketTotal, and Interactive Online Malware Analysis Sandbox—are used to gather 8175 ransomware instances from seven different families that are all encrypted with the TLS protocol. A random sample of 20,000 normal instances is chosen from the CIC-IDS2017 dataset. A computer running the Ubuntu 18.04 operating system and Python 2.7 as a software development tool are utilized as the testing environment. According to tests, TGAN-IDS increases classification accuracy for all seven families of ransomware by an average of 93.8% in 2.44 s. Improvement to the system is considered in the future to detect more malicious attacks with enhanced performance.

As a novel idea, Homayoun et al. [7] put out the deep ransomware threat hunting and intelligence system (DRTHIS) to quickly and effectively identify and hunt the ransomware threat. After examining the application activities, the long short-term memory (LSTM) concept and convolutional neural network (CNN) are utilized as the system's primary classifiers. Within the first 10 s of the application's operation, DRTHIS can identify the ransomware and its family. The suggested model is tested using a dataset of 204 good-ware samples and 660 ransomware samples. Results indicate that the true positive rate for DRTHIS is 97.2%. A wide range of deep learning training techniques is being considered for use in the future to hunt various infections.

A deep learning model based on a neural network algorithm was proposed by Berrueta et al. [8] as an additional effort. In a client-server architecture, all crucial files are saved on the servers, and the client only has access to them via a networked file-sharing service. However, if a crypto ransomware attack was to target only one client accessing the shared file server, it might infect all the workgroup files. The suggested model tracks file-sharing traffic to look for unusual actions taken by a crypto ransomware attack, like reading files, writing the encrypted version of files, and deleting files. A dataset is created

using 70 samples of crypto ransomware and 50 h of benign applications from users who accessed file-sharing servers during 2528 working hours. 80% of the formed dataset is used for training and the remaining 20% is used for testing. Ten additional undiscovered crypto ransomware samples are exclusively utilized for validation. Among the three algorithms that were looked at—decision trees, ensemble trees, and artificial neural networks—a neural network algorithm with three hidden layers was chosen. According to the results, the model has a false positive rate of 0.004% with more than 2400 h of user working traffic. 99.9% of crypto-ransomware attacks can be detected in an average of 30.2 s. Only Microsoft Windows desktop and laptop operating systems are considered client computers in this study; operating systems for mobile phones are not. Since the current version of the model exclusively addresses crypto ransomware threats, an adaptive version is needed to expand the model's usability.

Using ML algorithms, Adamu and Awan [9] analyze the ransomware behaviour across 5 chosen features out of 30,000 features. The ransomware dataset from the Resilient Information Systems Security (RISS) research group, which includes 942 examples of good-ware and 582 examples of ransomware from 11 different families, is used. As a result, the support vector machine (SVM) algorithm achieves an accuracy of 88.2%. The next improvement step is believed to be network behavior prediction utilizing machine learning and agent-based approaches.

Homayoun et al. [10] offer a model with a sequence pattern mining (SPM) technique to identify crypto ransomware activities in the network. As a dataset, 1624 examples of ransomware from three different families and 220 typical applications were employed. The model is trained using the J48, random forest, bagging, and MLP algorithms. The accuracy of the results in distinguishing between good-ware and ransomware is 99%. For further research, taking into account stream data mining techniques might be a great option. This model's application to the detection of ransomware, IoT platforms, and mobile malware are all worthwhile research topics.

The authors of [11] conduct a comparison study between batch data mining algorithms (decision tree (J48) and projective adaptive resonance theory (PART)) and stream data mining algorithms (Hoeffding tree (HT) and OzaBagAdwin (OBA)). The dataset UNSW-NB15, which includes examples of normal activity and nine different kinds of intrusion network attacks, is used. The objective is to evaluate the algorithms' performance for problems involving binary class and multiclass classification. Streaming algorithms outperform batch data algorithms in the binary classification problem (HT gets 98.38%, OBA gets 99.67%, J48 gets 94.70%, and PART gets 92.83%), where the algorithms classify the instance as normal or attack. In terms of latency, HT's result of 0.91 s is preferred to OBA's result of 4.08 s.

According to the recently studied literature, current proactive monitoring analysis approach models [3,6–9] get significantly efficient results in terms of accuracy with implemented techniques, new techniques are still required to provide the best performance in terms of accuracy and latency. Therefore, these models are facing the ransomware attacks' rapid evolution. In addition to maintaining high accuracy levels, stream data mining techniques [10,11] can minimize model latency.

3. Methodology

After examining recent studies in the literature, it is demonstrated that existing ransomware analysis approaches lake of an optimum solution with high accuracy and latency performance, where the model can quickly differentiate ransomware from good-ware. Therefore, a three-phase methodology is proposed to be used in order to accomplish our objectives.

3.1. Preparing the Ransomware Dataset

Based on the high latency performance characteristics, three stream data mining classification algorithms will be selected to be applied to the same dataset used in [3,9]. The Resilient Information Systems Security (RISS) research group dataset [12] is consisting of

942 good-ware and 582 ransomware from 11 different families. Data must be organized in a reliable way for the testing environment to produce correct findings. Monitoring and analysis of network traffic activity are done to proactively identify and prevent ransomware attacks. The dataset is preprocessed using Microsoft Excel and the Waikato Environment for Knowledge Analysis (WEKA) software.

3.2. Configuring Massive Online Analysis (MOA) Software

In this study, MOA software released on 07.2021 will be used as a testing framework. MOA can analyze on time a massive amount of data with limited memory and time resources. Thus, MOA will help in lowering the latency of network traffic analysis compared to traditional ML/DL algorithms [13]. However, MOA must be adjusted properly to obtain proper results. The results of Hoeffding trees classifier algorithms are then assessed to choose the enhanced model with the highest accuracy and latency performance.

3.3. Evaluating the Experimented Algorithms Accuracy and Latency

We will consider the classification accuracy and latency to assess the tested algorithms. Through the usage of MOA software, accuracy and latency are both observed and recorded. Then, ten tests will be run to ensure the correctness of the outputs from the MOA software for each stream data mining classification algorithm [14–16]. After that, accuracy and latency mean and standard deviation are determined. MS Excel is used as a calculation tool. The means are calculated using the AVERAGE() function, and the standard deviation is calculated using the STDEV.P() function. The last stage is to compare the experimental findings of the suggested model to all other findings as well as to previously published approaches. The study conclusion is then made. The overall model block diagram is illustrated in Figure 1.

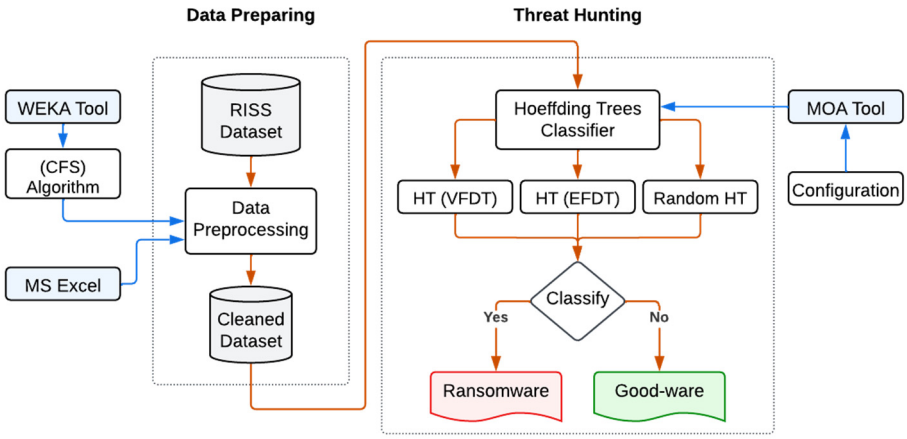

Figure 1. Model Implementation Block Diagram.

4. Proposed Solution

4.1. Hoeffding Trees Classifier

Hoeffding trees classifier is incremental decision trees suggested by Domingos and Hulten [17] to be suitable for extremely large datasets. The name of the classifier comes from a statistical idea known as the Hoeffding bound. Hoeffding bound ensures the certainty of a choice to separate new child leaves from old nodes according to a specified amount of passed data samples [13]. Then, leaves carrying the class label only are transferred to be internal nodes that carry the Hoeffding bound information for the next level [18]. Equation (1) calculates the statistical value for the Hoeffding bound [18,19].

$$HB = \sqrt{\frac{R^2 \ln \frac{1}{\delta}}{2n}} \quad (1)$$

where:

R: range of possible values of instances ($\text{Value}_{\text{max}}$—$\text{Value}_{\text{min}}$).

n: number of passing instances in each time interval.

δ: the error tolerance probability (given by the user).

The resulting tree is subject to change according to incoming Hoeffding bound information of data samples. It is a ready-to-use classifier after the first data sample is passed. The accuracy increased as time passed to examine more data samples using a sliding window. The Hoeffding tree classifier pseudocode [16] is displayed in Algorithm 1.

Algorithm 1 Hoeffding Tree Classifier Pseudocode

1: Let HT be a tree with a single leaf (the root)
2: for all training examples do
3: Sort example into leaf l using HT
4: Update sufficient statistics in l
5: Increment n_l, the number of examples seen at l
6: if $n_l mod\ n_{min} = 0$ and examples seen at l not all of same class then
7: Compute $\bar{G}_l(X_l)$ for each attribute
8: Let X_a be attribute with highest $\bar{G}_l$
9: Let X_b be attribute with second-highest $\bar{G}_l$
10: Compute Hoeffding bound $\epsilon = \sqrt{\frac{R^2 \ln \frac{1}{\delta}}{2n_l}}$
11: if $X_a \neq X_\varnothing$ and $(\bar{G}_l(X_a) - \bar{G}_l(X_b) > \epsilon \ or \ \epsilon < \tau)$ then
12: Replace l with an internal node that splits on X_a
13: for all branches of the split do
14: Add a new leaf with initialized sufficient statistics
15: end for
16: end if
17: end if
18: end for

Using just a root node, Line 1 initially constructs the tree structure. Lines 2–18 represent a loop that examines every instance. Depending on the statistical information included in each tree node, each instance will be sorted through the tree until it reaches the appropriate leaf. The node statistics are updated and the number of passed instances is incremented with every pass (Lines 4–5). The leaf statistical information is sufficient to decide whether or not further splitting is required. If the instance is not fit within the leaf, the splitting criterion function (G), which involves information gain, is calculated for the two top attributes (Lines 6–9). Then, the Hoeffding bound is calculated (Line 10). Then, the difference between the two top attribute gains is examined and compared to the Hoeffding bound. If the highest attribute exists and G exceeds the Hoeffding bound, splitting with new child nodes is carried out (Lines 11–12). Finally, initialize each node with the appropriate statistical data (Lines 13–14). A tiebreak (τ) is introduced by the user to control the unnecessary tree growth with a default value of 0.05 [16].

Hoeffding trees classifier is utilized in many algorithms. Very fast decision trees (VFDT), the first algorithm, was created using a Hoeffding tree classifier. VFDT is a decision tree learning algorithm based on Hoeffding trees. It acquires all Hoeffding tress characteristics. It offers extra features when the dataset is tiny, or the stream moves slowly enough to allow rescanning of the passed data samples in order to retain the highest levels of accuracy. The random Hoeffding tree is the random decision tree for stream data. In a random decision tree, the tree is built primally without using the training set of data. The root node is selected at random from one of the attributes in the supplied dataset. The tree will continue to grow until it reaches its maximum size. The statistical data for each node is then updated using the training data [20].

Recently, Manapragada et al. [21] developed an enhanced version of the VFDT algorithm known as the extremely fast decision trees (EFDT) algorithm. The basic idea is to split a node as soon as a beneficial split is noticed, and then to replace this split with a superior one when it is noticed. This is accomplished by comparing the worthiness of the current split attribute with the worthiness of no split choice rather than just the two top attributes. On the other hand, the traditional Hoeffding trees classifier seeks to only split the node whenever a better split is discovered, but without reviewing the splitting decision onwards. This unintentionally helps in concept-driven solving. Hence, EFDT is an overall-case scenario rather than the concept-adapting very fast decision tree (CVFDT) algorithm which is primarily intended for the drifting situation. As a result, EFDT produces fewer errors and learns faster [21].

4.2. Tools

4.2.1. Microsoft Excel

Microsoft Excel is spreadsheets manipulation software. It is dependent upon tables with rows and columns. A dataset is represented as a file with the comma-separated values extension (.CSV). Rows indicate the sample instances, whereas columns represent the attributes [22].

4.2.2. Waikato Environment of Knowledge Analysis (WEKA)

WEKA is an open-source software used to train a machine with different machine learning algorithms. It is one of the most cutting-edge software that is used to apply data mining concepts to batch data. It addresses the majority of data mining issues, including attribute selection, clustering, regression, and classification. It offers more services than just training, among them the preparation of the dataset and input data visualization [23].

4.2.3. Massive Online Analysis (MOA)

MOA is an open-source framework built in JAVA programming language to manipulate machine learning models for stream data. It is specialized and expandable software that allows researchers to change the present code or even add new functionalities depending on their demands in stream data mining investigations, as described in [24]. In this research, MOA software will be used as a testing framework. MOA can analyze on time a massive amount of data with limited memory and time resources [13]. To obtain the most accurate results, the testbed tool must be configured. MOA offer an extensive list of implemented algorithms used in various stream data mining techniques. Regression, clustering, and classification are a few examples.

4.3. Dataset Preparing

In order to compare the research findings with existing ransomware analysis approaches, the RISS research group ransomware dataset [12] is utilized. The dataset is open-source consisting of 942 good-ware and 582 ransomware from 11 different families. The number of feature headings is 16,830 features [9]. Pre-processing is required in order to get the dataset ready for use with the MOA testing tool. Since the features heading column is available in separate (.csv) file, the column is transposed using TRANSPOSE() function in MS Excel from being column to be the first raw in the main (.csv) file. The feature names are then changed to their serial numbers, such as feature1, feature2, . . . , and feature16380. Then, following the feature columns, the ransomware family names, and the label columns are pushed to the end. The delimiter between data cells is then changed from a tap to a comma. Then, the main (.csv) file is transform into an attribute-relation file (.arff) using WEKA software.

Furthermore, WEKA software is used to carry out the feature selection process. Correlation-based Feature Subset (CFS) algorithm was proposed by Hall, M. A. [25]. CFS is a filter algorithm that sort feature subsets as per the correlation-based evaluation function. The purpose is to reduce the entire feature set of a dataset to a subset of features that are

highly correlated with class and not correlated with one another. In other words, the CFS algorithm will accept a feature if its prediction class level exceeds the other examined features in an instance space. The CFS algorithm fundamental component is the correlation-based evaluation function. M_s is the calculated merit of feature subset (S) containing (k) features. M_s is given by Equation (2).

$$M_s = \frac{k\bar{r_{cf}}}{\sqrt{k + k(k-1)\bar{r_{ff}}}} \quad (2)$$

The mean feature-class correlation $\bar{r_{cf}}$ and the mean feature-feature inter-correlation $\bar{r_{ff}}$ are the two parameters that the formula primarily relies on. The correlation between each feature and each class is represented by the term "feature-class correlation", which is included in the equation numerator. The denominator of the equation includes "feature-feature inter-correlation", which is the inter-correlation of each feature with each other. $\bar{r_{cf}}$ and $\bar{r_{ff}}$ are calculated using one of three attribute quality measures: symmetrical uncertainty, Relief algorithm, or minimum description length (MDL) [25].

There are 16,380 attributes in the original set. CFS algorithm reduces the number to 26 features. Then, using the WEKA software's NumericToNominal filter, the attribute's type is transformed from numeric to nominal. Means the "0" then becomes "no" while the "1" becomes "yes". The ID Column is then removed, and the samples are then randomized using WEKA software's Randomize filter. For the more, the family number is changed to names, and the label values "yes" and "no" are changed to "Ransomware" and "Goodware", respectively.

4.4. MOA Software Configuration

Since the RISS research group dataset [12] is labeled, the classification technique is used. In traditional classification, the holdout set is used with the cross-validation process in which the time is consumed. The prequential scheme, Evaluate Prequential Delayed scheme, also known as the interleaved-test-then-train scheme, is a classification mode that performs testing prior training in level of batches of k instances in stream data. The level of blocks is used to ensure the speed of classification operation instead of the one-by-one mode. k indicates the number of instances in a block that the model will evaluate at once. The accuracy has grown throughout time in which all instances are utilized to get their maximum potential. Hence, instead of using the traditional holdout set classification mode where that time is consumed through a dedicated training procedure, this classification mode will enhance both accuracy and latency performance.

Given that dataset has 1524 instances altogether, the number picked for k is 1000. Each batch is processed by a sliding window to train the model on the initial window of 100 instances onwards. Every 10 instances are considered as one sample and the memory is examined. Configuration setting for the classification task in MOA is shown in Table 1.

Table 1. MOA Tool Configuration Setup for Classification Task.

No.	Parameter	Chosen Value
1	Classification Task	Evaluate Prequential Delayed
2	k	1000 instances
3	Initial window	100 instances
4	Training on initial window	Enable
5	Training in batches	Enable
6	Sample frequency	10 instances
7	Memory check Frequency	10 instances

5. Results and Discussion

5.1. Results

The study utilized three Hoeffding tree classifier algorithms (EFDT, VFDT, and random Hoeffding trees) to detect and prevent ransomware attacks. The algorithms were selected based on the high latency performance characteristics to be applied to the RISS research group dataset [12] after pre-processing. The testing was performed on a machine with Windows 7 Home Premium edition, 2.4 GHz i5 Intel CPU, and 6.00 GB RAM. The testing was conducted using the MOA graphical user interface (GUI).

Figure 2 presents a screenshot of the classification task configuration for the EFDT algorithm in the MOA software. The configuration steps involved selecting the Evaluate Prequential Delayed classification mode, choosing the EFDT algorithm as the learner, selecting the (.arff) file stream generator to input the processed dataset, and using the Windows Classification Performance Evaluator compatible with the installed Windows OS. Finally, the configuration process is then continued following Table 1. The default values are maintained for the remaining fields.

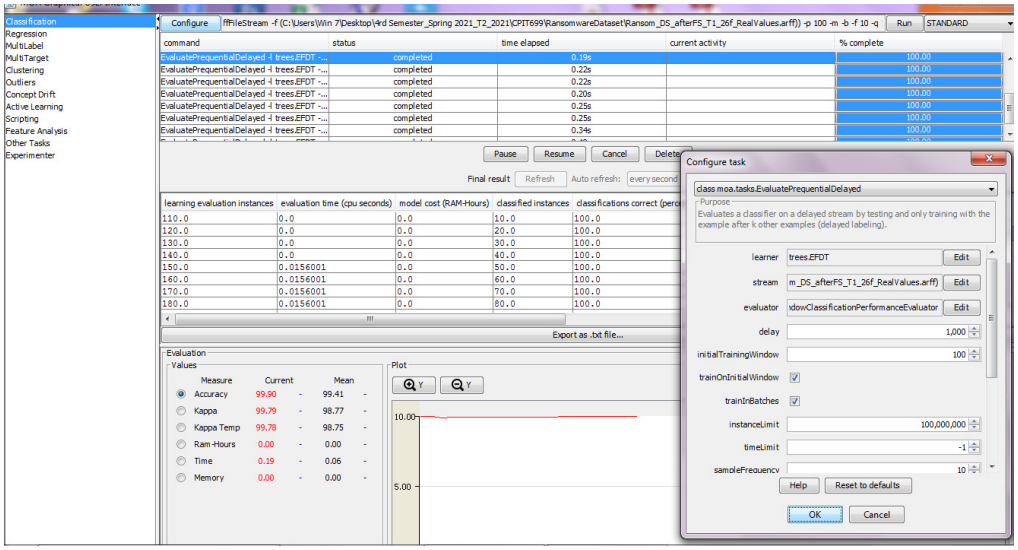

Figure 2. MOA Tool Configuration of EFDT Algorithm with Results.

Once the configuration was completed, the MOA software was ready for testing. The GUI displayed the task status, including information about the number of instances evaluated, evaluation time, percentage of instances correctly classified, number of tree nodes, and number of tree leaves for each classification interval. The GUI also presented evaluation metrics, such as accuracy and latency, with the current and mean values across all testing intervals. As it indicates the result after all dataset instances have been evaluated, the mean value is taken into account.

A ten-time test was conducted using the MOA software to ensure the reliability of the recorded accuracy and latency metrics [14,15]. Accuracy and latency values were recorded in an MS Excel sheet for each test. This process was repeated for all three Hoeffding trees algorithms, with the learner changed accordingly. The remaining screenshots of each algorithm are accessible in Appendix A.

After obtaining the results from the ten tests for each algorithm, the average and standard deviation (S) of the accuracy and latency metrics were calculated using the AVERAGE() and STDEV.P() functions in MS Excel. Table 2 provides a detailed overview of all the acquired results for the three algorithms.

Table 2. Results of Hoeffding Trees Classifier Algorithms in RISS dataset.

Test No.	Hoeffding Trees (EFDT)		Hoeffding Trees (VFDT)		Random Hoeffding Trees	
	Acc (%)	T (ms)	Acc (%)	T (ms)	Acc (%)	T (ms)
T1	99.41	60	99.41	70	64.31	40
T2	99.41	80	99.41	70	64.31	30
T3	99.41	70	99.41	90	64.31	30
T4	99.41	60	99.41	60	64.31	30
T5	99.41	40	99.41	70	64.31	30
T6	99.41	90	99.41	60	64.31	20
T7	99.41	40	99.41	70	64.31	10
T8	99.41	60	99.41	60	64.31	10
T9	99.41	80	99.41	60	64.31	10
T10	99.41	80	99.41	70	64.31	10
Avg	99.4	66	99.4	68	64.3	22
S	0	16.2	0	8.7	0	10.8

Acc = Accuracy (%), T = Latency (ms), Avg = Average, S = Standard Deviation.

Based on the results presented in Table 2, the following analysis and interpretation are made:

1. Accuracy: Both Hoeffding trees classifier algorithms (EFDT, and VFDT) consistently achieved an accuracy of 99.4% across all ten tests (T1 to T10). This indicates that the algorithms performed consistently well in accurately classifying instances in the RISS dataset. On the other hand, the random Hoeffding trees algorithm receives 64.3% because the starting root node was chosen at random. The Hoeffding tree algorithms use the Hoeffding bound, a statistical inequality, to make early decisions and avoid unnecessary computations, making them efficient.
2. Latency: The latency values varied for each algorithm and across different tests. The EFDT algorithm had an average latency of 66 ms, VFDT had an average latency of 68 ms, and random Hoeffding trees had the lowest average latency of 22 ms. Latency refers to the time taken to process and classify each instance. The variations in latency can be attributed to the different splitting criteria and tree construction strategies employed by each algorithm. With its lower latency, the random Hoeffding trees algorithm may have a more efficient decision-making process due to its randomization-based approach, which allows for faster classification. This proposes that it very well might be more effective regarding handling time contrasted with EFDT and VFDT. In any case, further examination is expected to decide whether this distinction in latency is measurably significant. In addition, the standard deviation (S) values indicate the variation in latency across the ten tests, with VFDT having the lowest deviation of 8.7 ms and EFDT having the highest deviation of 16.2 ms.
3. Performance Consistency: The high accuracy achieved by EFDT and VFDT algorithms indicates their consistency in correctly classifying instances in the RISS dataset. The similar accuracy values obtained across all tests suggest that the algorithms are robust and perform consistently under varying conditions. This consistency can be attributed to the adaptive nature of Hoeffding tree algorithms, which can update their models incrementally and adapt to changes in the data distribution.
4. Stability: The low standard deviation values for latency indicate relatively stable performance across the tests for each algorithm. This suggests that the algorithms are relatively sensitive to variations in the dataset or testing conditions.

Overall, the results indicate that EFDT and VFDT Hoeffding trees classifier algorithms performed well in terms of accuracy on the RISS dataset. The random Hoeffding trees algorithm showed the lowest latency, indicating potential efficiency advantages.

5.2. Discussion

Analyzing the currently employed approaches in the literature is advised in order to compare the proposed model findings. From [9], six ML algorithms are used to classify ransomware from good-ware in the RISS research group dataset. Support vector machine (SVM) was the most accurate algorithm with 88.2%. Furthermore, authors in [3] used deep neural network model with autoencoder that gets an accuracy rate of 99.7%. Published results are shown in Table 3 and illustrated in Figure 3.

Table 3. Accuracy of ML/DL Algorithms Used in Literature.

No.	Algorithms (ML/DL)	Accuracy (%)
1	Support Vector Machines (SVM)	88.2
2	Random Forest (RF)	84.0
3	Decision Trees (DT)	81.4
4	Bayesian Network (BN)	52.5
5	Artificial Neural Network (ANN)	86.0
6	Logistic Regression (LR)	65.7
7	Deep Neural Network with Autoencoder	99.7

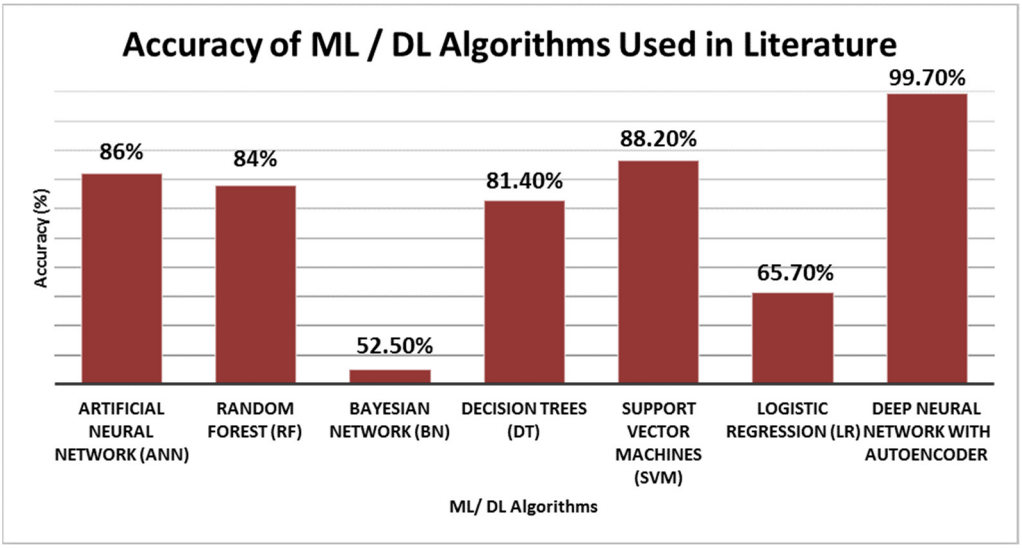

Figure 3. Accuracy of ML/DL Algorithms Used in Literature.

After computing the averages of classification accuracy and latency of experimented algorithms, results of accuracy and latency levels are summarized in Table 4 and respectively shown in Figures 4 and 5.

Table 4. Hoeffding Trees Classifier Algorithms Accuracy and Latency.

No.	Hoeffding Trees Classifier Algorithms	Accuracy (%)	Latency (ms)
1	Hoeffding Trees (EFDT)	99.4	66
2	Hoeffding Trees (VFDT)	99.4	68
3	Random Hoeffding Trees	64.3	22

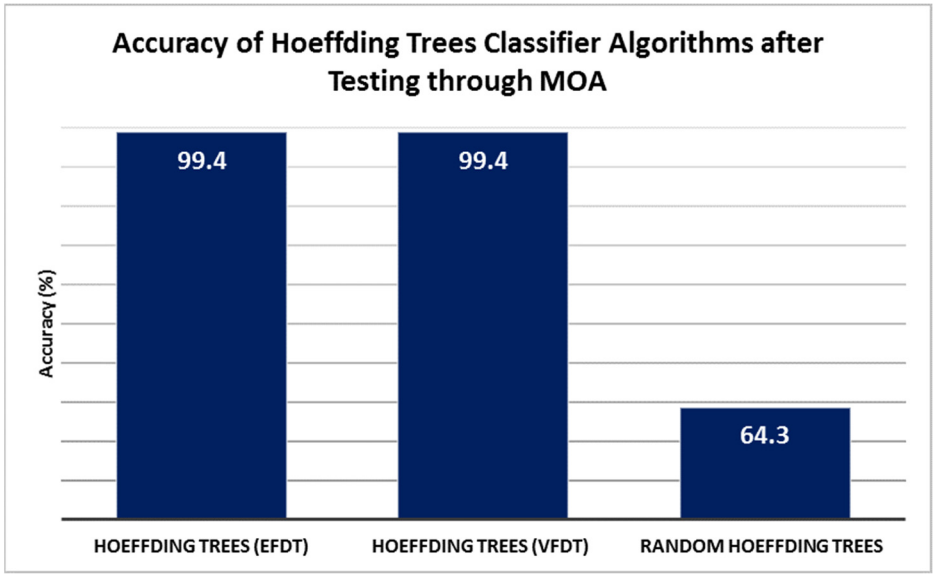

Figure 4. Accuracy of Hoeffding Trees Classifier Algorithms.

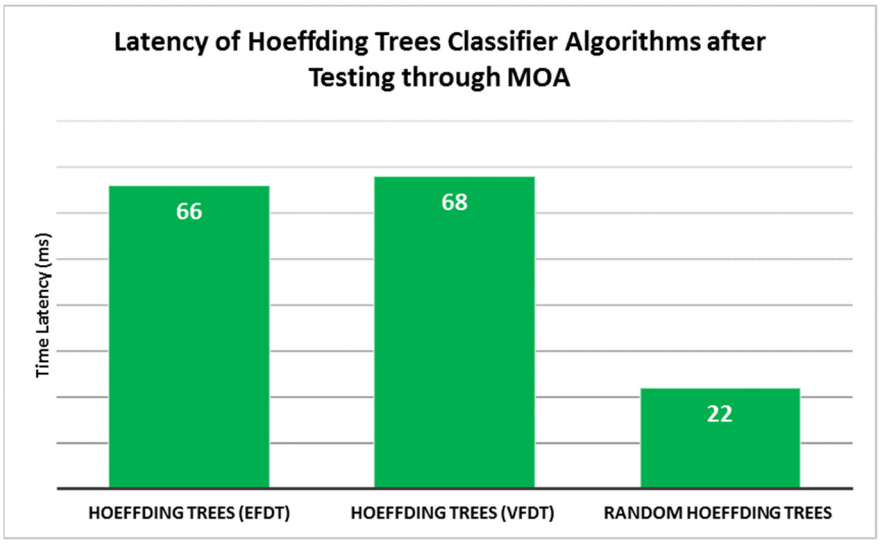

Figure 5. Latency of Hoeffding Trees Classifier Algorithms.

In terms of classification accuracy, both Hoeffding trees as EFDT and VFDT algorithms produce the highest accuracy results as of 99.4% of ransomware detection. Somehow the two algorithms resemble each other quite a bit since the EFDT algorithm is an enhanced version of VFDT algorithm. They are stream data mining classification algorithms. Hence, the difference will show up as the steam is huge enough. At this point, the latency will be the basis of the assessment. For both EFDT and VFDT algorithms, the false alarm percentage is estimated to be 0.6%.

Furthermore, when compared to ML algorithms utilized in literature, they both similarly achieve the highest accuracy. In traditional classification, the holdout set is utilized along with the time-consuming cross-validation procedure. Evaluate Prequential Delayed is a classification mode used in stream data mining classification techniques that performs testing prior training in batches of k instances to ensure the speed of classification operation instead of the one-by-one mode. The accuracy has grown throughout time in which all instances are utilized to get their maximum potential, especially with large stream data. However, when compared to the DL algorithm that gets 99.7% is higher than our model. DL techniques that involve several layers can increase accuracy.

On the other hand, the random Hoeffding trees algorithm receives 64.3% because the starting root node was chosen at random. Hence, the classification process may not benefit from this attribute selection randomness. Thus, the false alarm percentage is estimated to be 35.7%.

In terms of classification latency, the EFDT and VFDT algorithms produce nearly equal results at 66 and 68 ms, respectively. Because the used dataset is fairly small. As the dataset has a huge number of instances, the difference will be visible. In our situation, for instance, the dataset has 1524 instances, and the difference is 2 ms. Therefore, it is possible to forecast that the difference will be 2 s if we have a dataset containing a multiple of 1000 times instances of the present one.

However, the EFDT algorithm is taken a reasonably short amount of time to classify the ransomware for only 66 ms. Given that the VFDT algorithm waits before splitting a node and only splits if a better split is found. It differs from the EFDT algorithm in that it does not further review the splitting decision. In other words, EFDT splits as soon as it is confident that the new splitting node is more accurate than the current one. This suggests that the EFDT algorithm will function more effectively as the dataset grows. The random Hoeffding trees algorithm, on the other hand, has the highest latency performance, taking only 22 ms to construct the tree by randomly selecting an attribute as the root node.

To the best of our knowledge, the majority of research in the literature that used ML/DL algorithms for ransomware detection was more concerned with maximizing accuracy than minimizing latency. Even though the DL algorithm [3] gets higher than our model in accuracy, DL algorithms require extensive memory and time resources. On the other hand, recent rapid growth in ransomware development requires the fastest-ever proactive detection of ransomware. We adopt that idea in this study by examining the Hoeffding trees algorithms as one of the stream data mining classification techniques.

In the end, we evaluate the tested algorithms by considering both classification accuracy and latency. Therefore, it can be claimed that the EFDT algorithm is the best Hoeffding trees classifier algorithm in terms of classification performance in detecting ransomware attacks, with a 99.4% accuracy rate and latency of 66 ms.

6. Conclusions and Future Work

In conclusion, this study aimed to proactively protect users' systems and sensitive data from ransomware attacks by introducing an enhanced cyber threat hunting technique. Through analyzing six proactive monitoring model articles, we proposed an optimized model to overcome the limitations of existing cyber threat-hunting techniques.

Our approach focused on utilizing the Hoeffding trees classifier as a stream data mining classification technique to enhance the performance of the cyber threat hunting technique. Specifically, we employed three Hoeffding trees classifier algorithms: VFDT, random Hoeffding trees, and EFDT. By detecting and preventing ransomware attacks early, we aimed to achieve high classification accuracy while minimizing latency in behavioral analysis.

To evaluate our model, we utilized the RISS research group ransomware dataset after preprocessing it using Microsoft Excel and the WEKA software. The MOA software, configured as the testing framework, provided the platform for observing the classification accuracy and latency of detecting ransomware attacks. The testing phase was conducted on a Windows 7 Home Premium edition machine, featuring a 2.4 GHz i5 Intel CPU and 6.00 GB RAM.

The results of Hoeffding tree classifier algorithms are then assessed to choose the enhanced model with the highest accuracy and latency performance. Our findings indicated that the EFDT algorithm achieved the highest classification accuracy of 99.41%, with a latency performance of 66 ms. These results demonstrate the effectiveness of our model in enhancing the cyber threat hunting technique through accurate and efficient detection of ransomware attacks.

Despite the fact that the relatively small dataset used in this study serves as a proof-of-concept for our model, it provides significant insights and implications for subsequent research in this field. Moving forward, we prescribe the accompanying activities to reinforce our discoveries further:

1. Incorporating real-life network traffic datasets from the industry as input streams will validate and support our model's effectiveness;
2. Considering classification latency alongside accuracy when evaluating ransomware detection and prevention models will offer a more comprehensive performance assessment;
3. Exploring the accuracy and latency performance of other stream data mining classification algorithms, including rule-based, ensemble-based, nearest neighbors, and statistical-based approaches, will enable a broader comparison and evaluation of techniques;
4. Developing a hybrid model that combines the accuracy levels of the EFDT algorithm with the rapid response capabilities of the random Hoeffding trees algorithm could significantly enhance the performance of the cyber threat hunting technique;
5. Utilizing machines with advanced specifications will further optimize the model's performance and scalability;
6. Extending the applicability of our model to accurately and promptly identify other types of cyber attacks beyond ransomware will enhance its versatility and practicality;
7. Implementing our model as standalone software will facilitate its adoption and usage by end users, offering a comprehensive solution for protecting systems against ransomware attacks and preventing data loss.

By addressing these future research directions, the field of cyber threat hunting and ransomware detection can advance further, leading to more robust and effective solutions for safeguarding digital systems and data. Our research contributes valuable insights and establishes a foundation for continued exploration and improvement in this critical area of cybersecurity.

Author Contributions: Conceptualization, O.B.; methodology, I.B.; validation, O.B.; formal analysis, I.B.; investigation I.B. and O.B.; data curation, I.B.; writing—original draft preparation, I.B.; writing—review and editing, I.B. and O.B.; visualization, I.B.; supervision, O.B.; project administration, O.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: http://rissgroup.org/ransomware-dataset/, accessed on 4 February 2023.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. MOA Software Screenshots

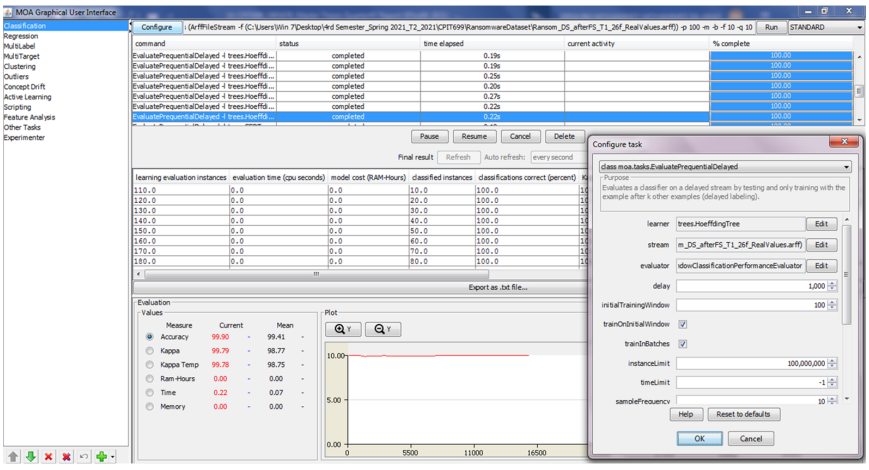

Figure A1. MOA Tool Configuration of VFDT Algorithm with Results.

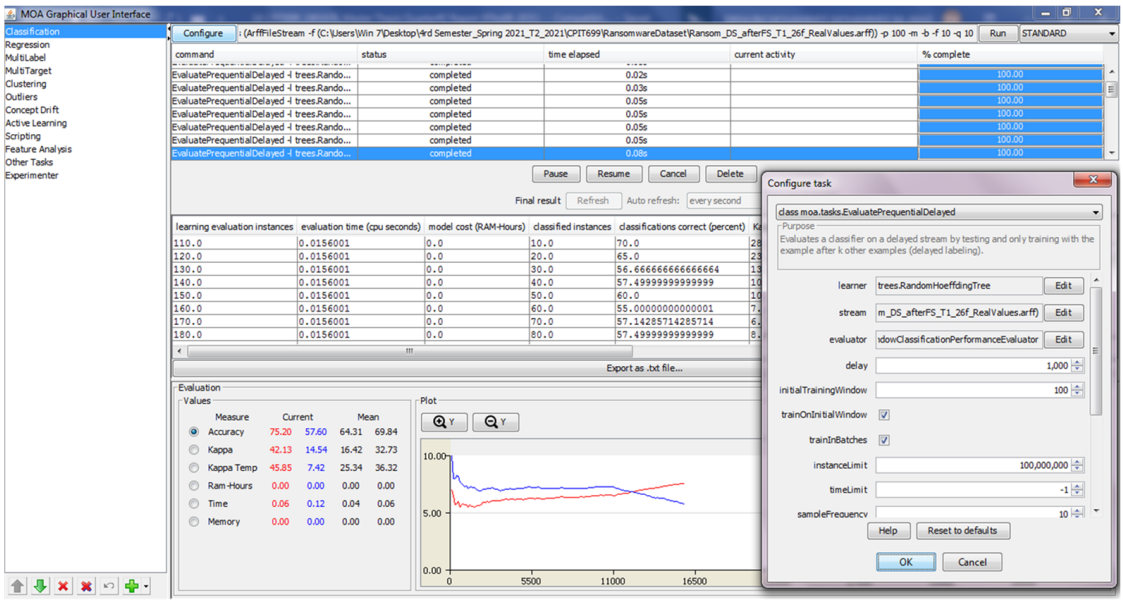

Figure A2. MOA Tool Configuration of Random Hoeffding Trees Algorithm with Results.

References

1. Kok, S.H.; Abdullah, A.; Jhanjhi, N. Early Detection of Crypto-Ransomware Using Pre-Encryption Detection Algorithm. *J. King Saud Univ. Comput. Inf. Sci.* **2020**, *34*, 1984–1999. [CrossRef]
2. Nguyen, G.; Dlugolinsky, S.; Tran, V.; Lopez Garcia, A. Deep Learning for Proactive Network Monitoring and Security Protection. *IEEE Access* **2020**, *8*, 19696–19716. [CrossRef]
3. AbdulsalamYa'u, G.; Job, G.K.; Waziri, S.M.; Jaafar, B.; SabonGari, N.A.; Yakubu, I.Z. Deep Learning for Detecting Ransomware in Edge Computing Devices Based on Autoencoder Classifier. In Proceedings of the 2019 4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT), Mysuru, India, 13–14 December 2019; IEEE: New York, NY, USA, 2019; pp. 240–243. [CrossRef]
4. Hindy, H.; Brosset, D.; Bayne, E.; Seeam, A.K.; Tachtatzis, C.; Atkinson, R.; Bellekens, X. A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems. *IEEE Access* **2020**, *8*, 104650–104675. [CrossRef]
5. Hulten, G.; Domingos, P.; Spencer, L. *Mining Massive Data Streams*; ProQuest Information and Learning Company: Ann Arbor, MI, USA, 2005.
6. Zhang, X.; Wang, J.; Zhu, S. Dual Generative Adversarial Networks Based Unknown Encryption Ransomware Attack Detection. *IEEE Access* **2022**, *10*, 900–913. [CrossRef]
7. Homayoun, S.; Dehghantanha, A.; Ahmadzadeh, M.; Hashemi, S.; Khayami, R.; Choo, K.-K.R.; Newton, D.E. DRTHIS: Deep Ransomware Threat Hunting and Intelligence System at the Fog Layer. *Future Gener. Comput. Syst.* **2019**, *90*, 94–104. [CrossRef]
8. Berrueta, E.; Morato, D.; Magaña, E.; Izal, M. Crypto-Ransomware Detection Using Machine Learning Models in File-Sharing Network Scenarios with Encrypted Traffic. *Expert Syst. Appl.* **2022**, *209*, 118299. [CrossRef]
9. Adamu, U.; Awan, I. Ransomware Prediction Using Supervised Learning Algorithms. In Proceedings of the 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), Istanbul, Turkey, 26–28 August 2019; IEEE: New York, NY, USA, 2019; pp. 57–63. [CrossRef]
10. Homayoun, S.; Dehghantanha, A.; Ahmadzadeh, M.; Hashemi, S.; Khayami, R. Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence. *IEEE Trans. Emerg. Top. Comput.* **2020**, *8*, 341–351. [CrossRef]
11. Adewole, K.S.; Salau-Ibrahim, T.T.; Imoize, A.L.; Oladipo, I.D.; AbdulRaheem, M.; Awotunde, J.B.; Balogun, A.O.; Isiaka, R.M.; Aro, T.O. Empirical Analysis of Data Streaming and Batch Learning Models for Network Intrusion Detection. *Electronics* **2022**, *11*, 3109. [CrossRef]
12. Ransomware Dataset—RISS. Available online: http://rissgroup.org/ransomware-dataset/ (accessed on 4 February 2023).
13. Kumar, A.; Kaur, P.; Sharma, P. A Survey on Hoeffding Tree Stream Data Classification Algorithms. *CPUH-Res.* **2015**, *5*, 28–32.
14. Garcia-Martin, E.; Bifet, A.; Lavesson, N.; König, R.; Linusson, H. Green Accelerated Hoeffding Tree. *arXiv* **2022**, arXiv:2205.03184.
15. Brownlee, J. How to Choose the Right Test Options when Evaluating Machine Learning Algorithms. MachineLearningMastery.com. Available online: https://machinelearningmastery.com/how-to-choose-the-right-test-options-when-evaluating-machine-learning-algorithms/ (accessed on 31 January 2023).
16. Srimani, P.K.; Patil, M.M. Performance Analysis of Hoeffding Trees in Data Streams by Using Massive Online Analysis Framework. *Int. J. Data Min. Model. Manag.* **2015**, *7*, 293. [CrossRef]

17. Domingos, P.; Hulten, G. Mining High-Speed Data Streams. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD '00, Boston, MA, USA, 20–23 August 2000; ACM Press: Boston, MA, USA, 2000; pp. 71–80. [CrossRef]
18. Yang, H.; Xu, A.; Chen, H.; Yuan, C. A Review: The Effects of Imperfect Data on Incremental Decision Tree. In Proceedings of the 2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Guangdong, China, 8–10 November 2014; IEEE: New York, NY, USA, 2014; pp. 34–41. [CrossRef]
19. da Costa, V.G.T.; Carvalho, A.C.P.D.L.F.D.; Junior, S.B. Strict Very Fast Decision Tree: A Memory Conservative Algorithm for Data Stream Mining. *Pattern Recognit. Lett.* **2018**, *116*, 22–28. [CrossRef]
20. Lomte, V.M.; Deorukhakar, H.B. A Survey of Random Decision Tree Framework Privacy Preserving Data Mining. *Int. J. Sci. Res. (IJSR)* **2012**, *3*, 11, 3135–3139.
21. Manapragada, C.; Webb, G.I.; Salehi, M. Extremely Fast Decision Tree. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; ACM: London, UK, 2018; pp. 1953–1962. [CrossRef]
22. Divisi, D.; Di Leonardo, G.; Zaccagna, G.; Crisci, R. Basic Statistics with Microsoft Excel: A Review. *J. Thorac. Dis.* **2017**, *9*, 1734–1740. [CrossRef] [PubMed]
23. Frank, E.; Hall, M.; Holmes, G.; Kirkby, R.; Pfahringer, B.; Witten, I.H.; Trigg, L. Weka-A Machine Learning Workbench for Data Mining. In *Data Mining and Knowledge Discovery Handbook*; Maimon, O., Rokach, L., Eds.; Springer US: Boston, MA, USA, 2009; pp. 1269–1277. [CrossRef]
24. Bifet, A.; Holmes, G.; Kirkby, R.; Pfahringer, B. MOA: Massive Online Analysis. *J. Mach. Learn. Res.* **2010**, *11*, 1601–1604.
25. Hall, M.A. *Correlation-Based Feature Selection for Machine Learning*; Department of Computer Science, The University of Waikato: Hamilton, New Zealand, 1999.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.