information

Article

The Effect of the Ransomware Dataset Age on the Detection Accuracy of Machine Learning Models

Qussai M. Yaseen [1,2]

[1] Artificial Intelligence Research Center (AIRC), College of Engineering and Information Technology, Ajman University, Ajman 20550, United Arab Emirates; q.yaseen@ajman.ac.ae
[2] Department of Computer Information Systems, Faculty of Computer and Information Technology, Jordan University of Science and Technology, Irbid 22110, Jordan

Abstract: Several supervised machine learning models have been proposed and used to detect Android ransomware. These models were trained using different datasets from different sources. However, the age of the ransomware datasets was not considered when training and testing these models. Therefore, the detection accuracy for those models is inaccurate since they learned using features from specific ransomware, old or new ransomware, and they did not learn using diverse ransomware features from different ages. This paper sheds light on the importance of considering the age of ransomware datasets and its effects on the detection accuracy of supervised machine learning models. This proves that supervised machine learning models trained using new ransomware dataset are inefficient in detecting old types of ransomware and vice versa. Moreover, this paper collected a large and diverse dataset of ransomware applications that comprises new and old ransomware developed during the period 2008–2020. Furthermore, the paper proposes a supervised machine learning model that is trained and tested using the diverse dataset. The experiments show that the proposed model is efficient in detecting Android ransomware regardless of its age by achieving an accuracy of approximately 97.48%. Moreover, the results shows that the proposed model outperforms the state-of-the-art approaches considered in this work.

Keywords: Android malware; information security; supervised machine learning; ransomware

Citation: Yaseen, Q.M. The Effect of the Ransomware Dataset Age on the Detection Accuracy of Machine Learning Models. *Information* **2023**, *14*, 193. https://doi.org/10.3390/info14030193

Academic Editor: Georgios Kambourakis

Received: 30 January 2023
Revised: 10 March 2023
Accepted: 11 March 2023
Published: 20 March 2023

1. Introduction

The rapid development of computer networks and technology has led to the exposure of smartphone functionalities, which are provided by different applications and operating systems. The Android operating system ranks first in terms of market share, followed by iOS and Samsung operating systems [1]. As shown in Table 1, as of July 2022, Android has gained more than 72% of the market share. The reason behind this is the flexibility that Google Play offers for developers in comparison to other stores [2].

The spread and flexibility of functionalities of smartphones made smartphones more convenient to the public; however, it has also brought security risks due to malicious applications. One of the most severe malicious applications is Android ransomware, which steals data and encrypts it, where users have only one choice by paying a ransom to the criminal to decrypt their data [3]. Figure 1 shows the process of Android ransomware attack. Android ransomware is gaining in momentum because of its increasing spread and severity on both individuals and companies assets. Therefore, there is a need for effective mechanisms that can prevent and detect Android ransomware.

Kaspersky, which is one of the most common antiviruses, has detected more than 17 k new mobile ransomwares out of three millions malicious applications in 2021 [4]. Cyber-criminals keep increasing their profits by implementing more and more applications [5], where we can find different types of Android ransomwares such as WannaCry, Locky, Cryptwall, etc. For this reason, the detection of Android ransomware has become one of the most prominent research topics in cybersecurity. The most common antiviruses are using signature-based approaches for detecting ransomware applications [6]. However, signature-based approaches have a limitation regarding detection, as these can only detect well-known applications. However, this makes it easier for the criminals to develop new ransomware applications to launch their cyberattacks.

Table 1. The percentage of market share for mobile operating systems.

Mobile Operating Systems	Percentage Market Share
Android	72.11%
IOS	27.22%
Samsung	0.42%

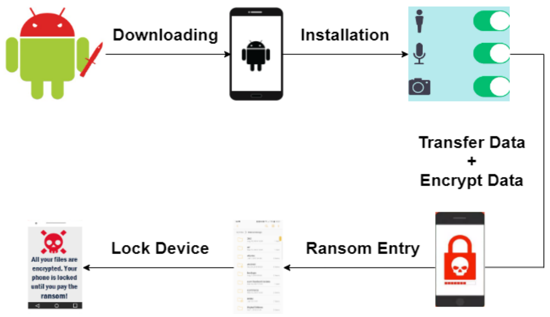
Downloading
Installation
Transfer Data
+
Encrypt Data
Ransom Entry
Lock Device
All your files are encrypted. Your phone is locked until you pay the ransom!

Figure 1. Android ransomware attack.

More research has been focusing on developing new approaches for the detection of Android ransomware applications. Machine learning methods have proved their efficiency in detecting Android malware such as the work in [7–10]. The efficiency of these approaches differs based on the extracted features from applications, besides the algorithms used for the detection task. Two analytical methods are used for feature extraction, which are the static and dynamic analyses. The static analysis works by decompiling ransomware applications to gather their metadata [11], where different types of features can be extracted from the manifest file such as signatures, permissions, and API-calls. Meanwhile, dynamic analysis requires a virtual environment to run applications and trigger their behavior to extract features [12]. Figure 2 shows the different features of static and dynamic analysis. One of the advantages of static analysis is the time consumption, since it consumes less time in comparison with the dynamic approach. Moreover, the permissions features, which are extracted in static analysis, are considered an efficient indicator that can be utilized for the detection of ransomware applications [13].

There are different studies conducted based on the permission features to detect Android ransomware. However, these studies used different datasets, where the ransomware were collected over specific periods from open source websites or acquired manually. These studies did not consider the variation of permission features requested by ransomware. In fact, once the implemented ransomware is detected by the models, the cybercriminals try to develop new types of ransomware. Moreover, as the Android operating system is updated and smartphones' functionalities are increased or enhanced, new permissions arise and are used by different malicious applications. That is, new applications may request new a set of permissions that did not exist in previous versions. Therefore, it is important to include the features of old and new ransomware applications in training the detection model, so that it accurately performs the task. Otherwise, there will be a gap in the models trained on old features and other models trained on new features to detect each other.

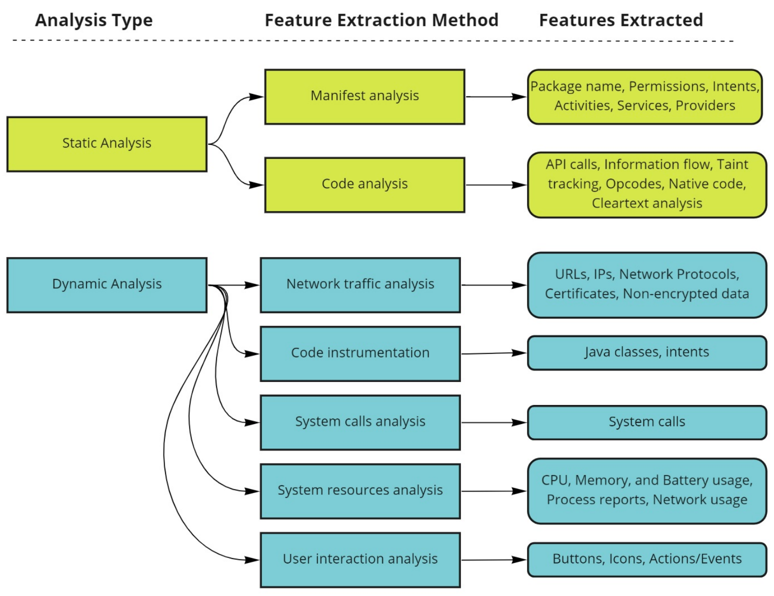
Analysis Type
Feature Extraction Method
Features Extracted
Static Analysis
Manifest analysis
Package name, Permissions, Intents, Activities, Services, Providers
Code analysis
API calls, Information flow, Taint tracking, Opcodes, Native code, Cleartext analysis
Dynamic Analysis
Network traffic analysis
URLs, IPs, Network Protocols, Certificates, Non-encrypted data
Code instrumentation
Java classes, intents
System calls analysis
System calls
System resources analysis
CPU, Memory, and Battery usage, Process reports, Network usage
User interaction analysis
Buttons, Icons, Actions/Events

Figure 2. Static and Dynamic Analysis Features.

An interesting work in this direction was proposed by Ceschin et al. [14]. The authors discussed the concept drift in Android malware in general. The authors used two datasets, namely Drebin [15] and Androzoo [16], collected during the period (2008–2018). The datasets were used to train and test the adaptive random forest classifier (ARF) and stochastic gradient descent (SFD). Moreover, they used four drift detectors (DDM, EDDM, ADWIN, andKSWIN) to prove their claim. Based on the results, the authors recommended updating the classifiers based on the emerging features to enhance the accuracy of the classifiers. Moreover, the authors recommended that resetting the classifiers when drift changes were detected is better than periodically resetting them based on a time window. Although their paper achieved significant findings, its scope is different from the scope of this paper that focuses on Android ransomware. Similarly, Kouliaridis et al. [17] used diverse datasets to train and test their proposed machine learning model. They collected features from different Android malware datasets as follows: Drebin dataset [15] from 2010 to 2012, VirusShare [18] from 2014 to 2017, and Androzoo [16] from 2017 to 2020. Their paper achieved high accuracy, however, its scope is different from the scope of this paper that focuses on Android ransomware, which is a severe problem nowadays. Moreover, this paper used more diverse Android ransomware samples from 2008 to 2020 by considering instances from ever year during this period. This paper considers one of the most prominent problems these days due to the high risk it poses to individuals, companies, organizations and countries. This highlights the problem of old classifiers in detecting new emerging ransomware, which could be a flaw of a high risk. Therefore, this paper, based on the achieved results, highly recommends retraining the classifiers to enhance their accuracy in preventing or detecting new and old Android ransomware. The contributions of this work are summarized as follows.

1. The paper evaluates the efficiency of features extracted from old ransomware applications in detecting new ransomware applications.
2. The paper evaluates the efficiency of features extracted from new ransomware applications in detecting old ransomware applications.
3. The paper proves that supervised machine learning classifiers only trained by the old ransomware dataset is not effective in detecting new ransomware. Moreover, the paper proves that supervised machine learning classifiers trained by the new ransomware dataset only is not effective in detecting old Android ransomware.
4. The paper creates a balanced and mixed dataset of old and new ransomware to effectively train supervised machine learning classifiers.

5. The paper proposes a permission-based Android ransomware detection model that can detect the ransomware applications of different ages. It used the top most supervised machine learning models, which are SVM, decision tree (DT), logistic regression (LG), K-near neighbor (K-NN) and random forest (RF) [19]. Moreover, it shows how the proposed model outperforms the state-of-the-art approaches in detecting Android ransomware applications.

The rest of the paper is organized as follows. Section 2 discusses some related work about Android ransomware detection. Section 3 demonstrates and discusses the proposed framework. Section 4 demonstrates and discusses the experiments and results, and compares between the efficiency of the proposed approach and some state-of-the-art approaches. Finally, Section 5 concludes the work.

2. Related Work

The use of machine learning algorithms and deep learning algorithms to detect Android malware have been studied extensively. Many of these approaches achieved very high accuracy, such as the work in [20–22]. However, The HelDroid tool [23], which was implemented in 2015, was the first method used to detect ransomware in the Android operating system. It was based on using the behavioral analysis of ransomware. HelDroid used textual features with natural language processing, and a Smali emulation technique to detect the locking scheme as indicators for file-encrypting flows and ransomware. Moreover, it used monitoring encryption calls and ransom text as static analysis for ransomware. HelDroid achieved a good accuracy of approximately 86%. However, this highly depends on the availability of textual features which is not always available, especially in some languages such as Chinese and Japanese. Furthermore, using encryption and other obfuscation methods can bypass HelDroid [24]. Recently, many approaches and research were proposed to enhance the detection of ransomware. This section discusses some related work about Android ransomware detection, and separates them according to the analysis approaches they used: namely the static, dynamic or hybrid approach.

2.1. Static Analysis

This section discusses some related work that used static features and machine learning algorithms to detect Android ransomware.

Zhang et al. [25] proposed a machine learning model that uses random forest to detect Android ransomware and their families. The proposed approach is based on using term frequency (TF) and N-gram to form a feature vector, and using opcode sequences as features. Different machine learning algorithms were tested to evaluate their effectiveness against the proposed model, which are DT, K-nearest neighbor (KNN), NB, and gradient boosting decision tree (GBDT). The results showed that RF outperformed other algorithms by achieving an accuracy of approximately 99.3%. However, the proposed model had some limitations in detecting some ransomware families such as exclusively locky, cryptowall, and reventon. The authors justified this limitation as due to the binary classification of the model. Similarly, Scalas et al. [26] proposed a random forest model for Android ransomware detection. Different static features were fed to the model such as API calls, packages, classes, and methods. The results show that random forest outperformed stochastic gradient descent (SGD) and SVM classifiers, which were tested in the experiments. However, the proposed model has some flaws that affect the accuracy of the API-based model because it replaces system-related entities with semantically equivalent or user-implemented entities.

Gaur et al. [27] proposed a machine learning model and extracted static features, such as opcodes and imported dlls, from a large dataset of about 23,000 ransomwares. The authors tested several conventional machine learning algorithms and neural network algorithms such as KNN, SVM, RF, DBSCAN, and neural networks. They claimed that their model achieved an accuracy of 99.68%.

Alsoghyer et al. [24] proposed an approach, called API-RDS, that uses static analysis for Android malware detection. The proposed approach used API calls as features to discriminate ransomware. The approach tested different algorithms, which are the naïve Bayes, decision tree, and random forest, and used three datasets collected from VirusTotal [28], RProber [29], and Koodous [30]. The authors claimed that their model achieved an accuracy of approximately 97%, and reduced the classification complexity by 26% after features reduction.

Su et al. [31] proposed a light-weight method for locker-ransomware detection. The authors performed a comprehensive analysis of ransomware behaviors and extracted display texts and background operation to avoid the obfuscation problem. Moreover, the authors applied the ensemble learning of different machine learning algorithms, which are logistic regression, support vector machines, random forests, and decision trees. These algorithms were tested using two datasets, which are the Anzhi Market [32] and Ransomware-transaction QQ groups [33]. The authors claimed that their approach achieved an accuracy of 99.98%.

Table 2 summarizes more related work about the static analysis of Android ransomware.

2.2. Dynamic Analysis

This section discusses some related work that used dynamic features and machine learning algorithms to detect Android ransomware.

Zakaria et al. [34] proposed a machine learning framework, called RENTAKA, that uses dynamic analysis to detect Android ransomware. The proposed approach extracted the used features based on the phases of ransomware lifecycle. The selected features are API calls, registration key, dropped file, files and directory operation, and embedded strings. These features were fed to several machine learning algorithms for testing, which are naïve Bayes, kNN, SVM, random forest, and J48. The algorithms were tested using a dataset collected from the Resilient Information System Security (RISS) research group from Imperial College London [35]. The authors claimed that support vector machines (SVMs) outperformed other algorithms by achieving the best accuracy of approximately 97.05%.

Abdullah et al. [36] proposed an Android ransomware detection model based on dynamic features. The proposed approach used system calls only as features extracted from a dataset collected by VirusTotal [28]. The selected features were fed to several machine learning algorithms for testing, which are random forest, J48, and naïve Bayes. The experimental results provided by the authors claimed that random forest algorithm outperformed other algorithms and achieved an accuracy of 98.31%.

Bibi et al. [37] proposed a deep learning malware detection model for Android ransomware using the long short-term memory (LSTM) algorithm. The proposed model used eight different algorithms to select features from the Packets information and Headers. A majority voting method was used to select approximately nineteen features. The model was tested using a dataset collected by the Canadian Institute of Cybersecurity [38]. The authors claimed that their model achieved an accuracy of approximately 97.08%.

Chen et al. [29] proposed a model called RansomProber that analyzes user interface widgets and users' finger movements of related activities, and coordinates them to check whether encryption operations are performed by users or by a crypto-ransomware. The model was tested using a dataset of 2721 ransomware collected from HelDroid [23] as well as some ransomware collected by the authors. The authors argued that some anti-virus tools perform poorly in ransomware detection. Meanwhile, they claimed that their approach detects crypto-ransomware with a high accuracy of approximately 99%.

Table 2 summarizes more related work about the dynamic analysis of Android ransomware.

Table 2. A comparison between some related work.

Work	Year	Analysis	Feature(s)	Dataset(s)	Algorithms	Accuracy	Ransomware Dataset Age
Gaur et al. [27]	2021	Static	Structural features such as imported dlls, opcode	Virusshare [18]	KNN, SVM, RF, DBSCAN, and neural networks	99.68%	Virusshare: Unknown
Alsoghyer and Almomani [24]	2019	Static	API calls and their frequencies	VirusTotal [28], RProber [29], and Koodous [30]	NB, DT, SMO, and RF	97%	RProper:HelDroid 2010–2014, HelDroid: collected 2010–2014, VirusTotal: Submitted 2017–2018, Koodous: Unknown
Su et al. [31]	2019	Static	Text, system operations, admin operations, permissions	Ransomware-transaction QQ groups [33]	LR, SVM, RF, and DT	99.98%	Ransomware-transaction QQ groups: Collected 2019
Kim et al. [39]	2019	Static	API calls, Permissions, opcodes, and environment	Malgenome [40] VirusShare [18]	SVM, RF, and multimodal deep learning	98%	Malgenome: 2010–2011, VirusShare: unknown
Scalas et al. [26]	2019	Static	API calls, classes, and methods	VirusTotal [28] and HelDroid [23]	RF	97%	VirusTotal:Unknown, HelDroid 2010–2014.
Zhang et al. [25]	2019	Static	Opcode sequences	VirusTotal [28]	DT, RF, K-NN, naïve Bayes, and GBD	99.3%	VirusTotal: 2012–2017
Zakaria et al. [34]	2022	Dynamic	API, registration key, embedded strings	RISS [35,41]	Naïve Bayes, kNN, SVM, RF, and J48	97.05%	RISS: 2016
Abdullah et al. [36]	2020	Dynamic	System calls	VirusTotal [28]	NB, J48 and RF	98.31%	VirusTotal: Unknown
Bibi et al. [37]	2019	Dynamic	Packets information and headers	CIC-ANDMal2017 [38]	LSTM	97.08%	CIC-ANDMal2017: 2017
Chen et al. [29]	2018	Dynamic	widget, activity, texts, buttons	HelDroid [23] and private dataset	User interface UI analysis technique to judge the legality of encryption operations.	99%	HelDroid 2010–2014, private dataset: 2013–2015
Aurangzeb et al. [42]	2022	Hybrid	DLLs, strings, and PE header,	VirusShare [18], EldeRan [43]	SVM, RF, KNN, XGBoost, and neural network	98.7%	VirusShare: unknown, EldeRan: 2016
Deepa et al. [15]	2019	Hybrid	System calls	Koodous [30] and Drebin [44]	RF, AdaBoost	99.9%	Koodous: unknown, Drebin: 2010–2012

Table 2. *Cont.*

Work	Year	Analysis	Feature(s)	Dataset(s)	Algorithms	Accuracy	Ransomware Dataset Age
Gharib et al. [45]	2017	Hybrid	Permissions, text, and API calls	R-PackDroid [46], HelDroid [23] and Contagio [47] Koodous [30]	NB, SVM, RF, and DNN	97.5%	R-PackDroid: 2015–2016, HelDroid: 2014–2015, Contagio: unknown, Koodous: Unknown
Ferrante et al. [48]	2017	Hybrid	Opcodes, CPU usage, and network traffic	HelDroid [23]	DT, LR, NB	100%	HelDroid: 2014–2015

2.3. Hybrid Analysis

This section discusses some related work that used both static and dynamic features and machine learning algorithms to detect Android ransomware.

Aurangzeb et al. [42] proposed BigRC-EML, which is a framework for ransomware detection using static and dynamic features. The proposed framework used ensemble machine learning methods and a big dataset to train several machine learning algorithms for ransomware detection. The machine learning algorithms that were tested are SVM, random forests, KNN, XGBoost, and neural network. These algorithms were tested using two datasets, namely VirusShare [18] and EldeRan [43], where many features were extracted. To decrease the number of features and select the most effective ones, the principle component analysis (PCA) was performed. The experiments showed that the best accuracy, which is 98%, was achieved by neural networks.

Deepa et al. [**?**] proposed a novel two-step feature selection approach, called RSST, which is based on a rough set and statistical test. The proposed approach used a hybrid technique that uses both dynamic and static system calls. Moreover, to reduce the complexity of feature space, the authors proposed a feature selection approach. The proposed methods were tested using the machine learning algorithms random forest and AdaBoost on the datasets Koodous [30] and Drebin [44]. Moreover, the authors tested the proposed feature selection method against different features' selection algorithms, which are information gain, CFsSubsetEval, ChiSquare, FreqSel, and symmetric uncertainty. Furthermore, the authors derived other features such as permissions, opcodes, API, methods, call graphs, Droidbox attributes, and network traces, and tested the proposed approach using this set of features and the derived system calls features. The experiments showed that the proposed approach using the proposed features algorithm and the derived system calls outperformed other feature selection algorithms and feature sets. The accuracy achieved by the proposed method was approximately 99.9%.

Gharib et al. [45] proposed a two-layer detection framework for Android malware, called the DNA-Droid. In this model, a dynamic analysis layer that works on the top of the static analysis layer was implemented. The DNA-droid used features such as permissions, text, images of logos, system, and API calls. In addition, it tested the algorithms NB, SVM, RF, and DNN. To profile ransomware families, DNA-Droid used sequence alignment techniques, which help in the detection of ransomware behavior early before the infection. The proposed method was tested using different datasets, which are PackDroid [46], HelDroid [23], and Contagio [47]. The results show that DNA-Droid achieved an accuracy of approximately 97.5%.

More research on using the hybrid analysis of Android ransomware detection was proposed, such as Sun et al. [49], who proposed a hybrid Android ransomware detection method, called MONET. The model was based on monitoring the Android application behavior and compared it against the behavior of malicious application. The method used API calls as features to distinguish malicious apps from benign ones. Similarly, the SAMADroid [33] and StormDroid [48] techniques offered hybrid analysis tools to detect Android ransomware. Table 2 summarizes more related work about the hybrid analysis of Android ransomware.

2.4. Issues in Related Work

Table 2 compares between some related works in the field of Android ransomware detection using machine learning models. Besides the different important information that is shown in the table, the table shows the collection date of the ransomware datasets used in the mentioned approaches. Obviously, many approaches did not even mention the age or the collection date of some or all of the used dataset, such as the datasets used in [15,26,27,36,42,45]. Generally speaking, mentioning the age of the ransomware dataset was not an important issue in many of the considered related works. Moreover, most of the mentioned related work approaches used a dataset collected in a specific period and did not consider extracting features from both old and new ransomware datasets. However, some approaches used datasets collected over a long period such as the work in [24], which used datasets collected from 2010 to 2014 and from 2017 to 2018, and the work in [25], which used a dataset collected from 2012 to 2017. However, they did not discuss the effect of the age of the Android ransomware dataset on the efficiency of their models. Meanwhile, this work used a ransomware dataset which was collected from 2008 to 2020 to show the importance of considering the age of the ransomware dataset on the detection accuracy of supervised machine learning models.

3. The Proposed Framework

The proposed framework aims to assess the robustness of old and new ransomware features in detecting each other. Furthermore, it aims to devise a detection model that can outperform the state-of-the-art work in terms of Android ransomware detection. Figure 3 shows the proposed methodology in detail. The following subsections presents a comprehensive discussion of the proposed framework.

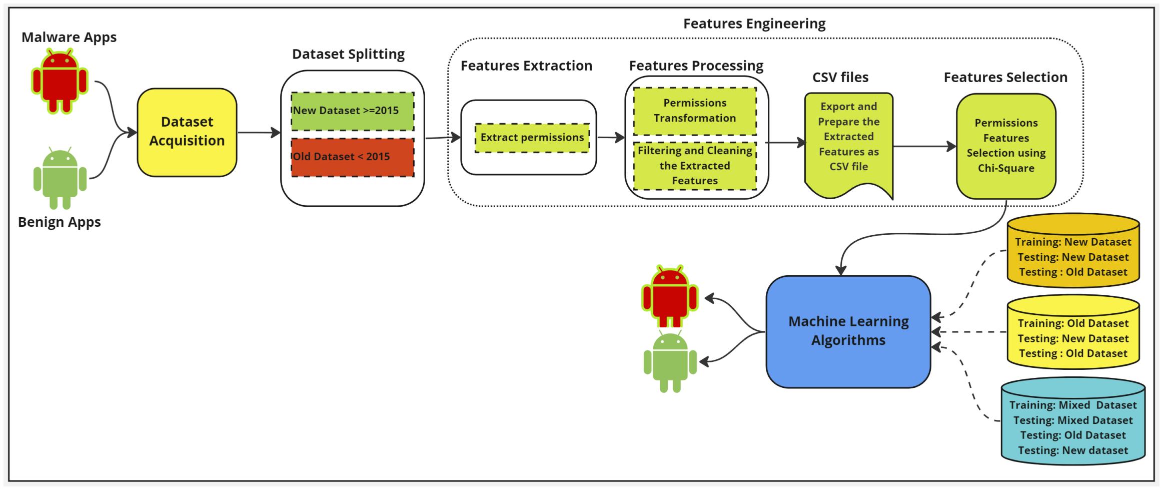

Figure 3. The methodology of the proposed work.

3.1. Dataset Acquisition

The dataset of this work was manually acquired from the AndroZoo [16], which is an open source repository that provides researchers with an extensive amount of malicious APKs. AndroZoo contains more than 19 million APKs, where these APKs may be benign or malicious applications. AndroZoo provides users with an up-to-date compressed CSV file that contains different fields. The collected malicious APKs were checked using VirusTotal to label ransomware ones.

VirusTotal is an open source website that offers a scanning service using approximately 70 third-party antivirus scanners [50]. This allows users to perform the submission in different ways, namely file, URL, and search. In this work, the search property was used for scanning, allowing users to perform scanning through searching using URL, IP address, domain, or file hash. The dataset was acquired as follows.

1. Use the up-to-date compressed CSV file provided by AndroZoo to create two lists of sha256 hash code for benign and malicious APKs.
2. Start scanning each malicious APK using the VirusTotal website.
3. Clean malicious list by excluding APKs which was not detected by at least one scanner as a ransomware.
4. Split the malicious list based on the dex date into two lists; the first one is the new list of ransomware APKs with a dex date after 2016–2020, whereas the remaining APKs are in the second list of old ransomware APKs.

5. Start downloading the APKs of the three lists, which are benign, new ransomware, and old ransomware datasets, and store them in distinct folders, where the downloading process is conducted using the official link provided by AndroZoo.

According to the process of data acquisition described above, two datasets will be produced, which are benign and ransomware APKs datasets. The ransomware dataset is labeled based on the dex_date in the compressed CSV file. Then, the ransomware dataset is separated into two datasets: the old dataset, which contains ransomware APKs that existed on or before 2015, and the new dataset, which contains ransomware APKs which existed between 2016 and 2020.

3.2. Feature Engineering

This paper used Androguard for the feature extraction. Androguard is an open source Python library that helps in performing the static analysis of APK files [51]. It provides a Dex2jar4 function, which works by decompiling the Dalvik bytecode into a compiled code of Java (.jar file). One of Androguard's functionalities is extracting the static features from the files inside the package file of the APK. Therefore, the permission features were extracted from each APK file in benign APKs, old ransomware APKs, and new ransomware APKs. Then, three datasets were constructed in CSV files; the first one contains permission features for both old ransomware and benign APKs; the second one contains permission features for both new ransomware and benign APKs; and the third contains the mixture of old and new ransomware, besides the benign APKs. Figure 4 demonstrates the methodology of generating the three datasets in this work.

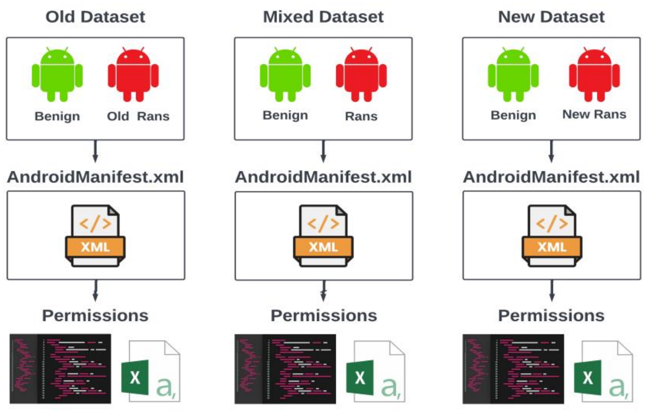

Figure 4. The methodology of creating the datasets.

There is a difference between each APK source code, which implies that each APK file requests different permissions. Therefore, the aggregation of permission features for all APK files, benign and ransomware APK files, in one dataset produces a huge number of features, which supervised machine learning algorithms cannot handle [52]. Moreover, the inappropriate features included in the dataset may affect the performance of supervised machine learning algorithms.

The final form of the CSV files contains categorical features with binary values that indicate the status of requesting each permission. Therefore, chi-square is utilized for feature selection [53]. chi-square is a statistical method that is used to rank the features based on their importance in predicting the target class. It is a statistical method that is used to assess the significance of a relationship between categorical variables. The dataset of this work contains the features of Boolean values, in which each feature has true or false values, which that is why chi-square is the best technique that can be performed for feature selection. Basically, the test identifies the significant difference between expectations E and observations O of the two categorical variables, where the high distance between E and

O indicates the strong correlation between variables. First, two hypotheses are defined as follows.

H0 (null): *Accept the independency between variables (irrelevant feature).*

H1 (alternative): *Reject the independency between variables (relevant feature).*

The score of Chi-square is calculated as follows.

$$\sum = \frac{(O-E)^2}{E} \tag{1}$$

where O is the observed value and E is the expected value.

The rejection of the H0 depends on the p-value, where H0 is rejected in case (p-value < 0.05), which means the feature is important for the detection task.

3.3. Detection Models

The final phase of this work is that of building the supervised machine learning detection models based on the dataset generated by the feature engineering phase. supervised machine learning algorithms help in detecting the unusual patterns, besides predicting the future behaviors to prevent criminals from doing their cyber-attacks [54]. In this work, the top supervised machine learning algorithms [19] were utilized for detecting ransomware files, which are decision tree, logistic regression, K-NN, SVM, and random forest.

3.3.1. Decision Tree

Decision tree (DT) is a supervised learning algorithm that is used for classification tasks. It is a rule-based algorithm that works by splitting the input into distinct output classes. The structure of the tree consists of root and leaf nodes, where the top feature is selected for node creation based on the values of entropy. The data points of the highest value of entropy exist in the tree root. Thus, it is split into smaller pieces for entropy value reduction, and this step is performed until the entropy value of the leaf node becomes zero. The entropy of low value indicates the higher value of the information gain, whereas the maximum value of the information gain indicates the optimal path in the decision tree. The entropy and information gain are computed as follows:

$$Entropy(T) = \sum_{i=0}^{T} -Pi \times log2(pi) \tag{2}$$

$$Entropy(T,S) = \sum_{i=0}^{C} -P(C) \times Entropy(C) \tag{3}$$

$$Information\ Gain(T,S) = Entropy(T) - Entropy(T,S) \tag{4}$$

where P is the probability function of class samples from the total number samples.

3.3.2. Logistic Regression

Logistic Regression (LR) is a supervised machine learning algorithm that is used for binary classification tasks. Suppose that X represents the features of datapoints, the logistic regression performs the binary classification and generates the output as follows:

$$\hat{y} = sigmoid(W^T X + b) \tag{5}$$

where X represents the features, W^T is the transposed N-dimensional vector, and b is the bias.

3.3.3. K-Nearest Neighbor

K-nearest neighbors (KNN) is a supervised machine learning algorithm that is used for classification tasks. It is considered a lazy learner, as it performs instance-based classification. The classification process works by finding the similarity of two datapoints in a single category, which is computed based on distance metrics. Then, a number k of nearest neighbors votes for which category the datapoints belong to. The number k of nearest neighbors is computed based on distance metrics and the cross-validation technique.

3.3.4. Support Vector Machine

Support vector machine (SVM) is a supervised machine learning algorithm that is used for the classification and regression tasks. Suppose that datapoints are distributed in space, the SVM works by finding a hyperplane that has the maximum margin between the classes. The margin is called the support vector, and it is defined as the distance between data points close to the class and the hyperplane that separates the data points into groups belonging to different classes.

3.3.5. Random Forest

Random forest (RF) is a supervised ensemble learning algorithm that is used for classification and regression tasks. It combines the decisions based on multiple construction, which can derive stronger results than a single decision tree. It uses the bagging technique, where the decision tree models are combined in parallel to avoid any dependency.

4. Experimental Results

The experiments were carried out using a 2.21 GHz CPU Intel Core (TM) i7-850H, with 32 GB memory. The operating system used is Windows 11, and the software that was used is Python (3.9) in the Anaconda environment.

The GridSearchCV technique [55] was used to tune each model by selecting the optimal hyperparameters that can generate the optimal results for the detection tasks. This works by creating a matrix that represents the combinations of all possible predefined hyperparameters values. In addition, it applies the cross-validation technique to the training set and split it into training and validation sets to avoid overfitting and underfitting issues. By the end of the training and validation process, the GridSearchCV keeps the best combination of hyperparameters that has achieved the best detection results. The predefined values of hyperparameters are shown in Table 3 for each machine learning algorithm.

Table 3. The predefined values of hyperparameters for the used supervised machine learning algorithms.

Machine Learning Algorithm	Hyperparameter	Values
SVM	Kernel	'Linear'
	C	(1, 0.25, 0.5, and 0.75)
	gamma	1, 2, 3, and 'auto'
DT	max_features	'auto', 'sqrt', and 'log2'
	ccp_alpha	(0.1, 0.01, and 0.001)
	max_depth	(5, 6, 7, 8, and 9)
	Criterion	'Gini' and 'entropy'
LG	C	log(−3,3, and 7)
	Penalty	l1 and l2
KNN	K	[1–30]
RF	n_estimators	(10, 100, 1000, 2000)
	Criterion	'Gini' and 'entropy'

Several experiments were performed based on both datasets to assess the robustness of the old and new ransomware features in detecting each other. Furthermore, the final experiment was conducted using the mixed dataset, which contains all ransomware samples besides the benign samples for the ransomware detection task. Figure 5 shows the distribution of the training and testing sets in each part of the experiments.

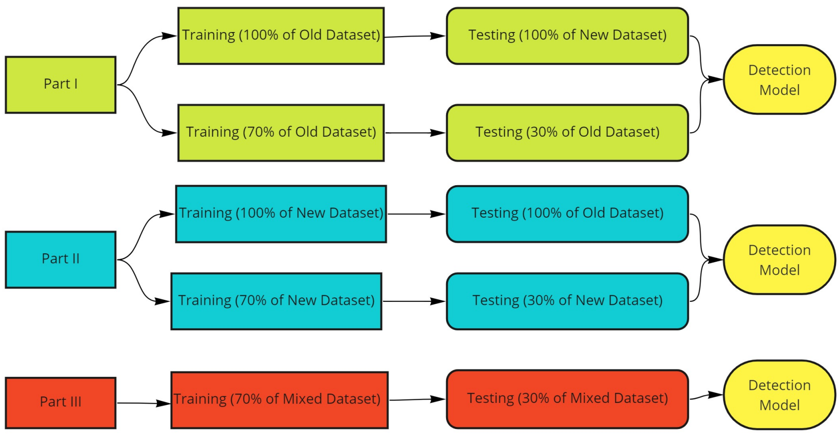
Part I
Training (100% of Old Dataset)
Testing (100% of New Dataset)
Training (70% of Old Dataset)
Testing (30% of Old Dataset)
Detection Model
Part II
Training (100% of New Dataset)
Testing (100% of Old Dataset)
Training (70% of New Dataset)
Testing (30% of New Dataset)
Detection Model
Part III
Training (70% of Mixed Dataset)
Testing (30% of Mixed Dataset)
Detection Model

Figure 5. The distribution of the training and testing datasets in each part of experiments.

The aforementioned algorithms, SVM, DT, LG, KNN, and RF, were trained and tested based on the explained sets. Moreover, the performances of each detection model were evaluated by calculating the appropriate metrics, namely the accuracy, precision, recall, and f1-score. These metrics are evaluated based on the results of the confusion matrix, which compute the true positive, false positive, true negative, and false negative rates for classes of binary classification. The evaluation metrics are computed as follows.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{6}$$

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

$$f1\_score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{9}$$

where *TP* denotes true positive, *TN* denotes true negative, *FP* denotes false positive, and *FN* is denotes false negative rates.

For a better understanding of the results, comparative analytics were conducted based on the evaluation results. The most efficient detection model will be presented to clarify the power of the proposed model in outperforming the previous studies. Moreover, the model with the best performance was trained on the old and new ransomware datasets, which were used to detect each other. Finally, the findings will be well explained to come up with a competitive stand in terms of Android ransomware detection, besides explaining the efficiency of having old and new ransomware features in our models for detecting ransomware files.

The experiments were divided into two parts. The first part aims at showing the effect of the age of ransomware dataset on the detection accuracy. Moreover, it aims at showing that supervised machine learning models developed based on old ransomware only or new ransomware only are not efficient in detecting ransomware. The second part aims to develop a supervised machine learning model that is trained using features extracted and selected from old and new ransomware to efficiently detect ransomware.

4.1. The Effect of the Ransomware Age on the Detection Accuracy

The collected dataset consists of 6340 ransomware and 2300 benign APK files. In this part, the dataset was divided into two datasets: old and new ransomware datasets, as shown in Table 4. The first one is the old ransomware dataset which consists of ransomware samples that existed between 2008 and 2015, and the second one is the new ransomware dataset which consists of ransomware samples which existed between 2016 and 2020. The benign samples were added to each dataset for classification purposes, and the permission features were extracted for each dataset separately.

Table 4. The distribution of ransomware APKs based on dex date.

Dataset	Year	Num. of Samples
Old Ransomware	2008	212
	2009	3
	2010	3
	2011	110
	2012	288
	2013	201
	2014	1412
	2015	2222
New Ransomware	2016	1001
	2017	490
	2018	232
	2019	325
	2020	46

The features used for classification in this work are the permissions. Table 5 shows the total number of permission features extracted from both datasets. As shown in the table, a huge number of features of approximately 4604 features were extracted. To reduce the number of features, chi-square technique was used. After the feature selection process, the number of features selected from the old ransomware was 161 out of 2521 permission features. Similarly, the number of selected features from the new ransomware dataset was 274 out of 2083 permission features. Hence, these features include the benign features in each dataset.

Table 5. The distribution of extracted and selected features for old and new ransomware APKs.

Dataset	Num. of Extracted Features	Num. of Selected Features
Old Ransomware	2521	161
New Ransomware	2083	274

The contribution of this paper is based on the assumption that there is a difference in the permissions requested by old ransomware and new ransomware. Therefore, before introducing the results of the experiments that support this assumption, we show in Table 6 the top-10 frequent permission features that distinguish each of the old and new Android applications. As shown in the table, the most frequent permissions that are requested by new ransomware are totally different from those requested by old ransomware. Therefore, training classifiers using old ransomware should affect their accuracy in detecting new ransomware and vice versa. The following subsections demonstrate the results of the experiments conducted on each dataset.

Table 6. Top-10 frequent permission features for old and new ransomware APKs.

Old Ransomware	New Ransomware
INSTALL_PACKAGES	MMOAUTH_CALLBACK
READ_GSERVICES	READ_COARSE_LOCATION
ACCESSORY_FRAMEWORK	WRITE_APN_STORAGE
ENABLE_NOTIFICATION	MAPS_RECEIVE
CHANGE_CONFIREAD_PHONE_STATEGURATION	RECEIVE_KEAT
RECEIVE_ADM_MESSAGE	GET_DETAILED_TASKS
UA_DATA	LOCAL_MAC_ADDRESS
SYSTEM_ALERT_WINDOW	HKBC_SEND
REMOVE_TASKS	UNINSTALL_SHORTCUT
QUICKBOOT_POWERON	INSTALL_SHORTCUT

4.1.1. Part I: Training Models Using Old Ransomware Dataset

The first part of the experiments was performed to assess the robustness of the old ransomware, where the algorithms were trained on the features extracted from the old ransomware dataset and benign samples only. Table 7 shows the datasets used for training and testing in this part. In the first experiment, the algorithms were trained using 100% of the old ransomware dataset and tested using 100% of the new ransomware dataset. While, in the second experiment, the algorithms were trained using 70% of the old ransomware dataset and tested using the remaining 30% of the old ransomware dataset.

Table 7. Dataset Distribution for the Experiments of Part I.

Experiment	Training Set	Testing Set
Experiment I	Old dataset (100%)	New dataset (100%)
Experiment II	Old dataset (70%)	Old Dataset (30%)

Figure 6 shows the results of the first experiment. As shown in the figure, the detection accuracy after training the algorithms using the old ransomware dataset ranges from 68.72% to 83.36%. The RF algorithm outperformed the remaining algorithms by achieving an accuracy of 83.36%, and it was able to correctly predict 85.86% of the testing samples. Meanwhile, the worst detection accuracy, which is 68.72%, was achieved by DT. Obviously, the detection accuracy is poor for all algorithms tested in this work. This means that old supervised machine learning models or supervised machine learning models that are trained using old datasets are not effective in detecting new ransomware applications. This means that countermeasures that employ such models are vulnerable, and companies that use such systems must address these vulnerabilities and patch their systems with new or updated countermeasures.

Figure 7 shows the results of the second experiment, which was conducted to enhance the findings of the first experiment and show that the tested algorithms are effective in ransomware detection when trained using the proper dataset. In this experiment, the models were trained using 70% of the old dataset and benign apps only, and tested using 30% of the old dataset and benign apps only. As shown in the figure, all algorithms except DT achieved very high accuracy using the old ransomware for training and testing, where they achieved 99.99% accuracy, and they were able to predict 99.99% of the testing samples correctly. The lowest detection accuracy (95.91%) was achieved by the DT, which is much higher than the best accuracy achieved in the first experiment. This experiment enhances the findings of the first experiment and assures that training models using the old dataset are not effective in detecting new ransomware.

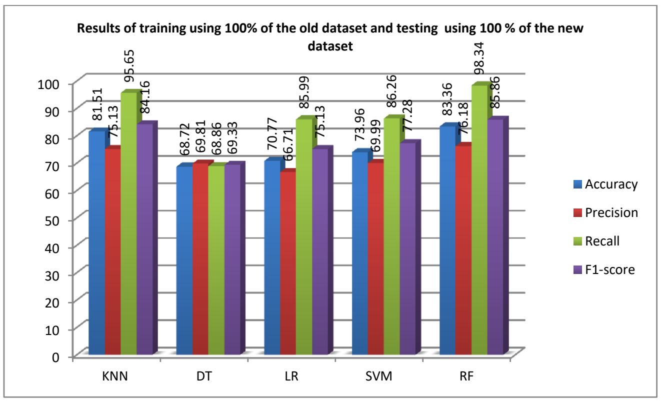

Figure 6. The performance of supervised machine learning algorithms trained on 100% of the old dataset and tested on 100% of the new dataset.

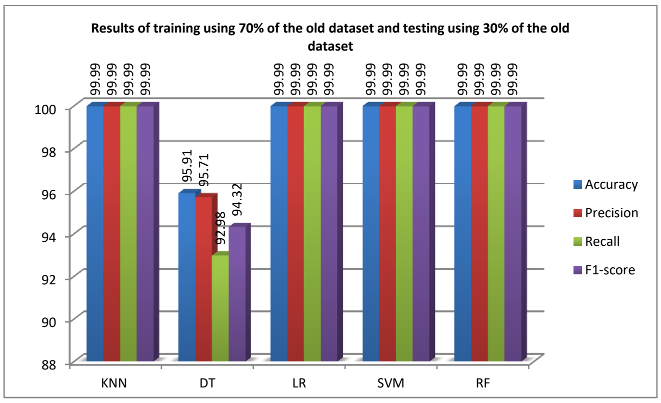

Figure 7. The performance of the supervised machine learning algorithms trained on 70% of the old dataset and tested on 30% of the old dataset.

The results reveal the fact that training algorithms using the features of old ransomware samples does not help in predicting new ransomware samples. The detection model of the highest performance predicted 14.14% of the new ransomware samples incorrectly.

4.1.2. Part II: Training Models Using New Ransomware Dataset

This part of the experiments was performed to assess the robustness of the new ransomware dataset, where the algorithms were trained on the features extracted from a new ransomware dataset and benign samples. This experiment was conducted to answer the question: can the training of supervised machine learning models using a new ransomware dataset lead to the efficient differentiation between new and old ransomware? Table 8 shows the datasets used for training and testing in this part. In the first experiment, the algorithms were trained using 100% of the new ransomware dataset and tested using 100% of the old ransomware dataset. In the second experiment, however, the algorithms were trained using 70% of the new ransomware dataset and tested using the remaining 30% of the new ransomware dataset.

Table 8. Datasets' Distribution for Experiments of Part II.

Experiment	Training Set	Testing Set
Experiment I	New dataset (100%)	Old dataset (100%)
Experiment II	New dataset (70%)	New dataset (30%)

Figure 8 shows the results of the first experiment. As shown in the figure, the detection accuracy after training the algorithms using the new ransomware dataset ranged from

58.13% to 70.96%, which was even worse than the accuracy achieved in part I when the model was trained using the old dataset and tested using the new dataset. The LR algorithm outperformed the remaining algorithms by achieving an accuracy of 70.96%, and it was able to predict 70.33% of the testing samples correctly. Meanwhile, the worst detection accuracy, which is 58.13%, was achieved by RF. Obviously, the models in this experiment learned new features which are not sufficient to detect old ransomware with old features. This was because the learning process was not correct. This may explain the unusual behavior of the models in this paper, where RF has the best accuracy.

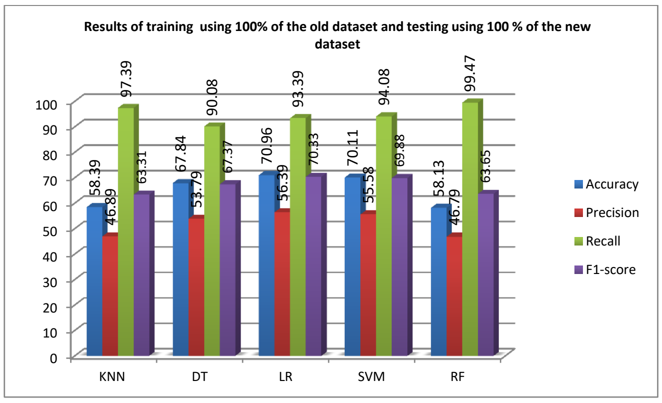

Figure 8. The performance of supervised machine learning algorithms trained on 100% of the new dataset and tested on 100% of the old dataset.

The results in this experiment assure that the results obtained in the first part by showing that training the supervised machine learning model using either only the new ransomware dataset only or only the old dataset is not enough to build a robust and efficient model for detecting ransomware.

Figure 9 shows the results of the second experiment, which was conducted to verify the findings of the first experiment and show that the tested algorithms are effective in ransomware detection when trained and tested using the right dataset. In this experiment, the models were trained using 70% of the new dataset and benign apps only, and tested using 30% of the new dataset and benign apps only. That is, the new ransomware dataset and benign apps were used for training and testing. As shown in the figure, the detection accuracy ranges from 77.34% to 92.52%, which is significantly better than the accuracy range achieved in the first experiment in this part. The highest detection accuracy, which is 92.52%, was achieved by RF, while the lowest detection accuracy, of 77.34%, was achieved by the DT.

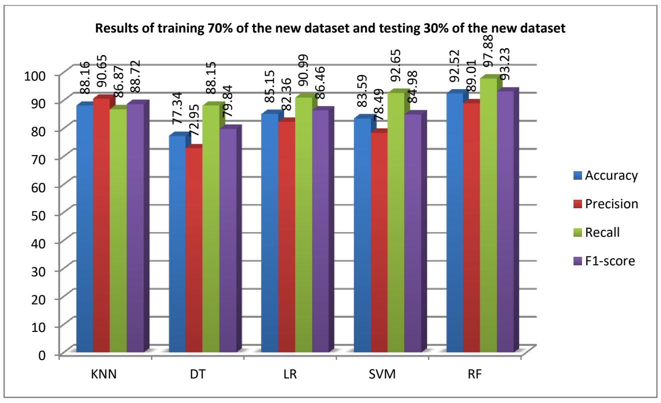

Figure 9. The performance of supervised machine learning algorithms trained on 70% of the new dataset and tested on 30% of the new dataset.

Two important findings can be inferred from the results in this experiment. The first one assures the previous findings from the first experiment, which is that using a new ransomware dataset for training is not enough to build an accurate supervised machine learning model to detect various ransomware. The second one can be inferred by comparing the results of Figure 9 and the results of Figure 7, where the detection accuracy by training and testing the models only using the old ransomware dataset is much higher than the accuracy obtained by training and testing the models only using the new ransomware dataset. This may explain the variance in permission features in both datasets. This means that the permission features requested by ransomware that existed from 2008 to 2015 are close or have low variance, while there may be a higher variance in permission features that are requested by ransomware applications which existed from 2016 to 2020.

4.2. Building an Effective Model Using Old and New Ransomware Datasets

This section aims to build a robust model that can efficiently detect ransomware regardless of the age of the ransomware. In addition, this section discusses the accuracy of the state-of-the-art considered in this work and how the proposed model outperforms the state-of-the-art. Therefore, to achieve this task, both datasets, old and new ransomware datasets, were merged to train and test the model.

The experiments were performed to assess the robustness of the combination of new and old ransomware together, where the algorithms trained on both features were extracted from new and old ransomware datasets as well as benign samples. The dataset was divided into two main datasets: the training dataset, which represents 70% of the dataset, and the testing dataset, which represents the remaining 30% of the dataset.

Figure 10 shows the results of the experiment. As shown in the figure, the detection accuracy ranged from 89.08% to 97.48%. The highest detection accuracy, which was 97.48%, was achieved by RF, while the lowest detection accuracy, which was 89.08%, was achieved by the DT. Obviously, the accuracy result (for all classifiers) achieved by this experiment is better than the accuracy result achieved by training the models only using the old ransomware dataset and testing them only using the new ransomware dataset (see Figure 6), or by training the classifiers only using the new dataset and testing the classifiers only using the old ransomware dataset (see Figure 8). Moreover, the accuracy results (for all classifiers) achieved by training the models using both new and old datasets is better than the accuracy results when training and testing the classifiers only using the new ransomware dataset, as shown in Figure 9.

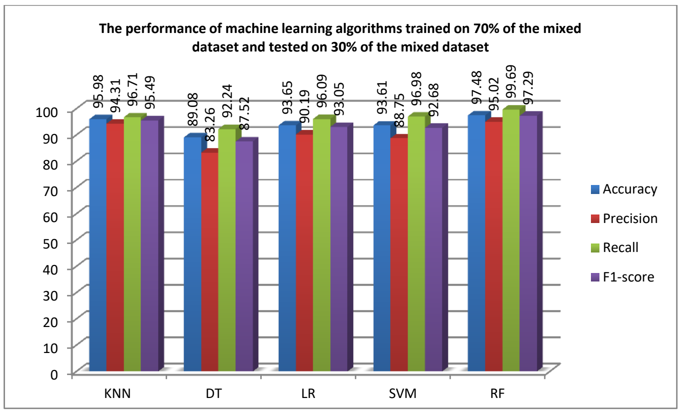

Figure 10. The performance of the supervised machine learning algorithms trained on 70% of the mixed dataset and tested on 30% of the mixed dataset.

4.3. Comparing the Proposed Approach with the State of Art Models

This paper considers recent and important state-of-the-art models in Android malware detection. It aims to show the significance of this work and raise awareness of the effects of omitting of the ransomware dataset age used in previous models. Table 9 shows a comparison between the performance of the proposed model and the state-of-the-art models seen in Sharma et al. [56], Alsoghyer et al. [24] and Alsoghyer et al. [57]. As shown in the table, the proposed approach used a large ransomware dataset, comprising 6340 ransomware, while the other approaches used a relatively small ransomware dataset of approximately 500 ransomware [24], 500 ransomware [57], and 2721 ransomware [56]. The proposed approach outperformed the state-of-the-art models [24,57] in terms of ransomware detection by 1% and 0.6%, respectively. However, the proposed approach used a larger number of features. The state of art model [56], outperformed our approach in terms of accuracy. However, the authors in [56] used a very large number of features—1045—in their comparison with the proposed model which only used 241 features. Moreover, the authors in [56] used a small ransomware dataset of 2721 ransomware in the comparison with that used in the proposed approach, which used 2721 ransomware. Furthermore, their work included only two types of Android ransomware, namely locker and crypto. Similarly, the state-of-the-art model [25] outperformed the proposed model in terms of accuracy. However, the authors used a relatively small dataset of only 1787 ransomware compared to the 6340 ransomware used in the proposed work. Furthermore, they used a very large number of opcode sequences as features, while the proposed work only used 241 features. In addition, the type of features used in the proposed approach, which is permissions, is different from the type of features used by the state-of-the-art model, which used opcode sequences. Moreover, the age of the used ransomware dataset used by the state-of-the-art model spans over five years only—from 2012 to 2017. Meanwhile, the used ransomware dataset used in the proposed work is from 2008 to 2020.

For a fair comparison, none of the previous state-of-the-art models or the discussed related work considered the age factor in ransomware acquisition. Although the state of art models mentioned the collection dates of some of their datasets, however, they did not extract the build date of the ransomware applications to check the ransomware age. That is, their models may use features that are inefficient in detecting the ransomware of age different than those used in the testing dataset. Therefore, the accuracy they obtained needs to be reviewed.

4.4. Recommendations and Work Limitations

The results of this work prove that there is a need to consider a diverse dataset that has features from ransomware of different ages when training and testing supervised machine learning models. In addition, it shed light on the inaccuracy of the previous work that used features from specific and not diverse ransomware datasets.

Section 4.2 presented the accuracy of the classifiers that learned using the features extracted from both new and old ransomware. The mixed dataset consists of an approximately equal number of features from both new and old ransomware. However, there is a trade-off between the weight given to the old features and new features. Classifiers that learn using a high number of new features in comparison to a smaller number of old features may have degraded accuracy in terms of detecting old ransomware and vice versa. This is an optimization problem that needs to be considered when building successful classifiers or in continual learning classification. The work in this direction is left to future work.

Another limitation of this work is the point of split between old and new ransomware datasets. We chose half of the period 2008–2020, which is 2015, to differentiate between old ransomware and new ransomware. However, there is a need to choose different split points to evaluate the effect of the split point on the accuracy. This is also left to future work.

Table 9. A comparison between the recent state-of-the-art works and the proposed study.

Work	Year	Ransomware Samples	Benign Samples	Features	Number of Selected Features	Classification Algorithm	Accuracy	Dataset Age
[24]	2019	500	2959	API calls	173	Random forest	96.5%	HelDroid: collected 2010–2014; VirusTotal: submitted 2017–2018; Koodous: unknown
[25]	2019	1787	NA	Opcode sequences	695,945	Random forest	99.3%	VirusTotal: 2012–2017
[57]	2020	500	500	Permissions	115	Random forest	96.9%	HelDroid: collected 2010–2014; VirusTotal: submitted 2017–2018; Koodous: unknown
[56]	2020	2721	2000	Permissions	1045	Logistic regression	99.5%	RansomProber: collected from HelDroid 2010–2014
The Proposed Work	2022	6340	2300	Permissions	241	Random forest	97.5%	AndroZoo: 2008–2020

5. Conclusions

There has been significant development in the detection of Android ransomware applications. Moreover, there is growing research in using machine learning algorithms to build intelligent detection models. However, there is a discrepancy in the performance of these models due to the abundance of data, preprocessing techniques, type of analysis, features extraction methods, and classification approaches. As a result of this huge effort, supervised machine learning models are able to obtain a high level of accuracy in detecting Android ransomware. However, there are still shortcomings in the different aspects in these models. This paper has addressed one important issue of these shortcomings, which is the age of Android ransomware dataset. The paper has proved that not considering this issue when training and testing supervised machine learning models may produce incorrect accuracy. For this purpose, the paper collected a large and diverse ransomware dataset with different ransomware ages from 2008 to 2020. This dataset was been split into two datasets, comprising the new dataset of ransomware that were developed between 2015 and 2020, and an old dataset of Android ransomware that were built between 2008 and 2015. These datasets were interchangeably used for training and testing. The experiments have shown that training supervised machine learning classifiers using one of these datasets and using the other for testing achieved poor detection accuracy. However, training the classifiers using a mixed dataset of new and old ransomware achieved a high accuracy of approximately 97.48% in detecting new or old ransomware. The interesting results was achieved using random forest classifier. This successful model was compared with interesting state-of-the-art models and showed that the proposed methodology achieved a competitive accuracy compared to the state-of-the-art models, although the state-of-the-art model did not use diverse datasets for either training or testing. Therefore, based on this work and results, the paper recommends that the accuracy of supervised machine learning models should be reviewed using the diverse datasets of different ages to achieve more authenticity.

The proposed work may be extendable to cover more cases. For example, in addition to supervised machine learning, neural networks may be used to test the contribution of the paper. Moreover, finding the optimal split date between old and new ransomware may be conducted. Furthermore, exploring different training dataset sizes may enhance the quality of the work. These ideas shape the future work stemming from this paper.

Funding: This research was funded by Ajman University, UAE under Grant 2021-IRG-ENIT-19.

Data Availability Statement: AndroZoo [16].

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

References

1. Becker, J. Standards for Automotive Operating Systems. *ATZelectron. Worldw.* **2022**, *17*, 58. [CrossRef]
2. Almahmoud, M.; Alzu'bi, D.; Yaseen, Q. ReDroidDet: Android Malware Detection Based on Recurrent Neural Network. *Procedia Comput. Sci.* **2021**, *184*, 841–846. [CrossRef]
3. Sharma, S.; Kumar, R.; Rama Krishna, C. A survey on analysis and detection of Android ransomware. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e6272. [CrossRef]
4. Shishkova, T. The Mobile Malware Threat Landscape in 2022. 2022. Available online: https://securelist.com/mobile-threat-report-2022/108844/ (accessed on 4 March 2022).
5. O'Kane, P.; Sezer, S.; Carlin, D. Evolution of ransomware. *IET Netw.* **2018**, *7*, 321–327. [CrossRef]
6. Al-Asli, M.; Ghaleb, T.A. Review of Signature-based Techniques in Antivirus Products. In Proceedings of the 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 3–4 April 2019; pp. 1–6. [CrossRef]
7. Shatnawi, A.S.; Yaseen, Q.; Yateem, A. An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms. *Procedia Comput. Sci.* **2022**, *201*, 653–658. [CrossRef]
8. Odat, E.; Alazzam, B.; Yaseen, Q.M. Detecting Malware Families and Subfamilies Using Machine Learning Algorithms: An Empirical Study. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 761–765. [CrossRef]
9. Almomani, I.M.; Khayer, A.A. A Comprehensive Analysis of the Android Permissions System. *IEEE Access* **2020**, *8*, 216671–216688. [CrossRef]

10. Singh, J.; Thakur, D.; Gera, T.; Shah, B.; Abuhmed, T.; Ali, F. Classification and Analysis of Android Malware Images Using Feature Fusion Technique. *IEEE Access* **2021**, *9*, 90102–90117. [CrossRef]
11. Li, L.; Bissyandé, T.F.; Papadakis, M.; Rasthofer, S.; Bartel, A.; Octeau, D.; Klein, J.; Traon, L. Static analysis of android apps: A systematic literature review. *Inf. Softw. Technol.* **2017**, *88*, 67–95. [CrossRef]
12. Andersson, K.; Shim, J.; Lim, K.; je Cho, S.; Han, S.; Park, M. Static and Dynamic Analysis of Android Malware and Goodware Written with Unity Framework. *Secur. Commun. Netw.* **2018**, *2018*. [CrossRef]
13. Aung, Z.; Zaw, W. Permission-Based Android Malware Detection. *Int. J. Sci. Technol. Res.* **2013**, *2*, 228–234.
14. Ceschin, F.; Botacin, M.; Gomes, H.M.; Pinagé, F.; Oliveira, L.S.; Grégio, A. Fast & Furious: On the modelling of malware detection as an evolving data stream. *Expert Syst. Appl.* **2023**, *212*, 118590.
15. Kumar, D.; Radhamani, G.; Vinod, P.; Shojafar, M.; Kumar, N.; Conti, M. Identification of Android malware using refined system calls. *Concurr. Comput. Pract. Exp.* **2019**, *31*, e5311. [CrossRef]
16. androzoo. Available online: https://androzoo.uni.lu/ (accessed on 30 July 2022).
17. Kouliaridis, V.; Kambourakis, G.; Geneiatakis, D.; Potha, N. Two Anatomists Are Better than One—Dual-Level Android Malware Detection. *Symmetry* **2020**, *12*, 1128. [CrossRef]
18. Virusshare. Available online: https://virusshare.com/ (accessed on 30 July 2022).
19. Gong, D. Top 6 Machine Learning Algorithms for Classification. 2022. Available online: https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501 (accessed on 4 December 2022).
20. AlJarrah, M.N.; Yaseen, Q.M.; Mustafa, A.M. A Context-Aware Android Malware Detection Approach Using Machine Learning. *Information* **2022**, *13*, 563. [CrossRef]
21. Massarelli, L.; Aniello, L.; Ciccotelli, C.; Querzoni, L.; Ucci, D.; Baldoni, R. AndroDFA: Android Malware Classification Based on Resource Consumption. *Information* **2020**, *11*, 326. [CrossRef]
22. Berman, D.S. DGA CapsNet: 1D Application of Capsule Networks to DGA Detection. *Information* **2019**, *10*, 157. [CrossRef]
23. Andronio, N.; Zanero, S.; Maggi, F. HelDroid: Dissecting and Detecting Mobile Ransomware. In *RAID 2015: Research in Attacks, Intrusions, and Defenses, Proceedings of the International Symposium on Recent Advances in Intrusion Detection, Kyoto, Japan, 2–4 November 2015*; Bos, H., Monrose, F., Blanc, G., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 382–404.
24. Alsoghyer, S.; Almomani, I. Ransomware Detection System for Android Applications. *Electronics* **2019**, *8*, 868. [CrossRef]
25. Zhang, H.; Xiao, X.; Mercaldo, F.; Ni, S.; Martinelli, F.; Sangaiah, A.K. Classification of ransomware families with machine learning based onN-gram of opcodes. *Future Gener. Comput. Syst.* **2019**, *90*, 211–221. [CrossRef]
26. Scalas, M.; Maiorca, D.; Mercaldo, F.; Visaggio, C.A.; Martinelli, F.; Giacinto, G. On the effectiveness of system API-related information for Android ransomware detection. *Comput. Secur.* **2019**, *86*, 168–182. [CrossRef]
27. Gaur, K.; Kumar, N.; Handa, A.; Shukla, S.K. Static Ransomware Analysis Using Machine Learning and Deep Learning Models. In *ACeS 2020: Advances in Cyber Security, Proceedings of the International Conference on Advances in Cyber Security, Penang, Malaysia, 8–9 December 2020*; Anbar, M., Abdullah, N., Manickam, S., Eds.; Springer: Singapore, 2021; pp. 450–467.
28. VirusTotal. Available online: https://www.virustotal.com/ (accessed on 30 July 2022).
29. Chen, J.; Wang, C.; Zhao, Z.; Chen, K.; Du, R.; Ahn, G.J. Uncovering the Face of Android Ransomware: Characterization and Real-Time Detection. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1286–1300. [CrossRef]
30. Koodous: Collective Intelligence against Android Malware. Available online: https://koodous.com/ (accessed on 30 July 2022).
31. Su, D.; Liu, J.; Wang, X.; Wang, W. Detecting Android Locker-Ransomware on Chinese Social Networks. *IEEE Access* **2019**, *7*, 20381–20393. [CrossRef]
32. Anzhi Market. Available online: http://www.anzhi.com/ (accessed on 30 July 2022).
33. HaboMalHunter. Available online: https://github.com/Tencent/HaboMalHunter (accessed on 30 July 2022).
34. Zakaria, W.Z.A.; Abdollah, M.F.; Mohd, O.; Yassin, S.M.W.M.S.M.M.; Ariffin, A. RENTAKA: A Novel Machine Learning Framework for Crypto-Ransomware Pre-encryption Detection. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 378–385. [CrossRef]
35. Kok, S.; Abdullah, A.; Zaman, N.; Supramaniam, M. Prevention of Crypto-Ransomware Using a Pre-Encryption Detection Algorithm. *Computers* **2019**, *8*, 79. [CrossRef]
36. Abdullah, Z.; Muhadi, F.W.; Saudi, M.M.; Hamid, I.R.A.; Foozy, C.F.M. Android Ransomware Detection Based on Dynamic Obtained Features. In *Recent Advances on Soft Computing and Data Mining, Proceedings of the International Conference on Soft Computing and Data Mining, Melaka, Malaysia, 22–23 January 2020*; Ghazali, R., Nawi, N.M., Deris, M.M., Abawajy, J.H., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 121–129.
37. Bibi, I.; Akhunzada, A.; Malik, J.; Ahmed, G.; Raza, M. An Effective Android Ransomware Detection through Multi-Factor Feature Filtration and Recurrent Neural Network. In Proceedings of the 2019 UK/ China Emerging Technologies (UCET), Glasgow, UK, 21–22 August 2019; pp. 1–4. [CrossRef]
38. Lashkari, A.H.; Kadir, A.F.A.; Taheri, L.; Ghorbani, A.A. Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification. In Proceedings of the 2018 International Carnahan Conference on Security Technology (ICCST), Montreal, QC, Canada, 22–25 October 2018; pp. 1–7. [CrossRef]
39. Kim, T.; Kang, B.; Rho, M.; Sezer, S.; Im, E.G. A Multimodal Deep Learning Method for Android Malware Detection Using Various Features. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 773–788. [CrossRef]
40. MalGenome Project. Available online: http://www.malgenomeproject.org (accessed on 30 July 2022).

41. Alqahtani, A.; Gazzan, M.; Sheldon, F.T. A proposed Crypto-Ransomware Early Detection(CRED) Model using an Integrated Deep Learning and Vector Space Model Approach. In Proceedings of the 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 6–8 January 2020; pp. 0275–0279. [CrossRef]
42. Aurangzeb, S.; Wang, C.; Anwar, H.; Naeem, M.A.; Aleem, M. BigRC-EML: Big-data based ransomware classification using ensemble machine learning. *Clust. Comput.* **2022**, *25*, 3405–3422. [CrossRef]
43. Sgandurra, D.; Muñoz-González, L.; Mohsen, R.; Lupu, E.C. Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection. *arXiv* **2016**, arXiv:1609.03020.
44. Arp, D.; Spreitzenbarth, M.; Hübner, M.; Gascon, H.; Rieck, K. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In Proceedings of the 2014 Network and Distributed System Security (NDSS) Symposium, San Diego, CA, USA, 23–26 February 2014. [CrossRef]
45. Gharib, A.; Ghorbani, A. DNA-Droid: A Real-Time Android Ransomware Detection Framework. In *NSS 2017: Network and System Security, Proceedings of the International Conference on Network and System Security, Helsinki, Finland, 21–23 August 2017*; Yan, Z., Molva, R., Mazurczyk, W., Kantola, R., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 184–198.
46. R-PackDroid Dataset. Available online: https://goo.gl/RVxfxL (accessed on 30 July 2022).
47. Parkour, M. Contagio Mini-Dump. Available online: http://contagiominidump.blogspot.it/ (accessed on 30 July 2022).
48. Ferrante, A.; Malek, M.; Martinelli, F.; Mercaldo, F.; Milosevic, J. Extinguishing Ransomware—A Hybrid Approach to Android Ransomware Detection. In *FPS 2017: Foundations and Practice of Security, Proceedings of the International Symposium on Foundations and Practice of Security, Nancy, France, 23–25 October 2017*; Imine, A., Fernandez, J.M., Marion, J.Y., Logrippo, L., Garcia-Alfaro, J., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 242–258.
49. Sun, M.; Li, X.; Lui, J.C.S.; Ma, R.T.B.; Liang, Z. Monet: A User-Oriented Behavior-Based Malware Variants Detection System for Android. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 1103–1112. [CrossRef]
50. Huang, H.; Zheng, C.; Zeng, J.; Zhou, W.; Zhu, S.; Liu, P.; Chari, S.; Zhang, C. Android malware development on public malware scanning platforms: A large-scale data-driven study. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 1090–1099. [CrossRef]
51. Martín García, A.; Lara-Cabrera, R.; Camacho, D. A new tool for static and dynamic Android malware analysis. In Proceedings of the FLINS 2018: The 13th International FLINS Conference on Data Science and Knowledge Engineering for Sensing Decision Support, Belfast, UK, 21–24 August 2018; pp. 509–516. [CrossRef]
52. Khalid, S.; Khalil, T.; Nasreen, S. A survey of feature selection and feature extraction techniques in machine learning. In Proceedings of the 2014 Science and Information Conference, London, UK, 27–29 August 2014; pp. 372–378. [CrossRef]
53. Jović, A.; Brkić, K.; Bogunović, N. A review of feature selection methods with applications. In Proceedings of the 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 25–29 May 2015; pp. 1200–1205. [CrossRef]
54. Rege, M. Machine Learning for Cyber Defense and Attack. In Proceedings of the DATA ANALYTICS 2018: The Seventh International Conference on Data Analytics, Porto, Portugal, 26–28 July 2018.
55. Pirjatullah; Kartini, D.; Nugrahadi, D.T.; Muliadi; Farmadi, A. Hyperparameter Tuning using GridsearchCV on The Comparison of The Activation Function of The ELM Method to The Classification of Pneumonia in Toddlers. In Proceedings of the 2021 4th International Conference of Computer and Informatics Engineering (IC2IE), Depok, Indonesia, 14–15 September 2021; pp. 390–395. [CrossRef]
56. Sharma, S.; Krishna, C.R.; Kumar, R. Android Ransomware Detection using Machine Learning Techniques: A Comparative Analysis on GPU and CPU. In Proceedings of the 2020 21st International Arab Conference on Information Technology (ACIT), Giza, Egypt, 28–30 November 2020; pp. 1–6. [CrossRef]
57. Alsoghyer, S.; Almomani, I. On the Effectiveness of Application Permissions for Android Ransomware Detection. In Proceedings of the 2020 6th Conference on Data Science and Machine Learning Applications (CDMA), Riyadh, Saudi Arabia, 4–5 March 2020; pp. 94–99. [CrossRef]