Contents lists available at GrowingScience

International Journal of Data and Network Science

homepage: www.GrowingScience.com/ijds

Securing cryptocurrency transactions: Innovations in malware detection using machine learning

Ghassan Samara[a*], Abeer Al-Mohtaseb[a], Hayel Khafajeh[a], Raed Alazaidah[a], Omar Alidmat[a], Ahmad Nasayreh[b], Mazen Alzyoud[c] and Najah Al-shanableh[c]

[a]*Faculty of Information Technology, Zarqa University, Jordan*
[b]*Faculty of information technology, Yarmouk University, Jordan*
[c]*Computer Science Department, Al al-Bayt University, Mafraq, Jordan*

C H R O N I C L E

Article history:
Received: April 25, 2024
Received in revised format: May 20, 2024
Accepted: July 1, 2024
Available online: July 1, 2024

Keywords:
Cryptocurrency
Malware
Machine Learning-Based Malware Detection

A B S T R A C T

Cryptocurrencies are crucial in modern commerce and finance, whether at the national, corporate, or individual level. They serve as fundamental currencies for buying and selling, enabling various business transactions. However, the rise of cybercrime has brought about concerns regarding their operations, potential breaches in encrypted currencies, and the security systems managing them. The frequency of attack tactics and the motivation of attackers seeking financial gain are well-known. Many cryptocurrencies lack the necessary algorithms, techniques, and knowledge to effectively detect and mitigate malware, making them vulnerable targets for hackers. In this study, machine learning techniques are employed to detect malicious code in digital currencies. Additionally, a comparison of these techniques is conducted to determine the most suitable algorithm and technology, Furthermore, this study highlights the importance of effective malware detection in securing cryptocurrencies. Three datasets of different sizes were used, each yielding distinct results based on dataset size. The AdaBoost model demonstrated superior performance when applied to the short dataset, while the decision tree model performed best with the medium-sized dataset. Conversely, the Naive Bayes model consistently produced the worst results, while the large-size KNN model achieved the highest performance.

1. Introduction

Cryptocurrencies have experienced significant growth in recent years and are now competing with traditional economic drivers such as gold and petroleum derivatives. As depicted in Fig. 1, cryptocurrencies are a type of digital currency that utilizes encryption to protect transactions and control the release of new units. Despite its virtual nature, these currencies are notoriously difficult to counterfeit (Hussain et al., 2021). Since their introduction in 2008, cryptocurrencies have gained widespread usage (Abu Al-Haija et al., 2021; Clouston et al., 2019; Yin et al., 2019). In 2017, the market experienced a surge, allowing users to raise funds without dealing with or verifying the owners of the money (Alazaidah et al., 2024). It operates in a decentralized manner, free from the control of any single entity, with circulation and exchange solely driven by its users (Liu et al., 2020). The underlying technology relies on peer-to-peer (P2P) networking, which has the potential to revolutionize the financial landscape (Kulkarni et al., 2020; Samara et al., 2020; Samara, 2020). However, the popularity of cryptocurrencies also makes them an attractive target for hackers and scammers seeking to exploit unsuspecting users for monetary gain. Threats such as forgery, fraud, and attempts to steal information, accounts, and money from clients have become increasingly prevalent (Alazaidah et al., 2023a; Aljaidi et al., 2022; Alazaidah et al., 2023b; Samara et al., 2020). As cryptocurrencies play an ever-growing role in international, national, business, and personal financial transactions, concerns have arisen regarding the evolving nature of attacks and the systems managing them. Recent high-profile hacks have provided hackers with new avenues to profit. Recognizing the need for effective strategies and algorithms to protect against malware and identify

* Corresponding author.
E-mail address gsamara@zu.edu.jo (G. Samara)

ISSN 2561-8156 (Online) - ISSN 2561-8148 (Print)

doi: 10.5267/j.ijdns.2024.7.003

malicious software, researchers have embarked on determining the most suitable approaches for safeguarding computers (Samara, Al Besani et al., 2020; Yazdinejad et al., 2020; Al-Shanableh et al., 2024; Bahrami et al., 2019). This research proposes a novel machine learning (ML) algorithm for detecting bitcoin malware, aiming to identify and analyze the most widespread forms of such malware. The primary objective of this proposed scheme is to establish frameworks for the common ML malware detection approaches currently utilized in the bitcoin industry.

2. Related Work

Cryptocurrency, often referred to as crypto, is a digital currency that relies on encryption keys for its operation. Transactions such as buying, selling, and transferring funds are securely recorded and monitored through a distributed ledger called a blockchain (Abadi et al., 2016). Unlike traditional fiat currency, cryptocurrencies are not issued by governments or other monetary authorities (Bahrami et al., 2019). While there are around 700 cryptocurrencies in existence, only a few are easily tradable, and even fewer have a market value exceeding $100 million. Bitcoin, created by Satoshi Nakamoto (a pseudonym), introduced open-source code in 2009. With the use of a network of communication endpoints called nodes, blockchain technology enables the broadcasting, validation, and storage of data structures known as blocks in public and distributed databases, making various possibilities achievable (Alhawi et al., 2018).

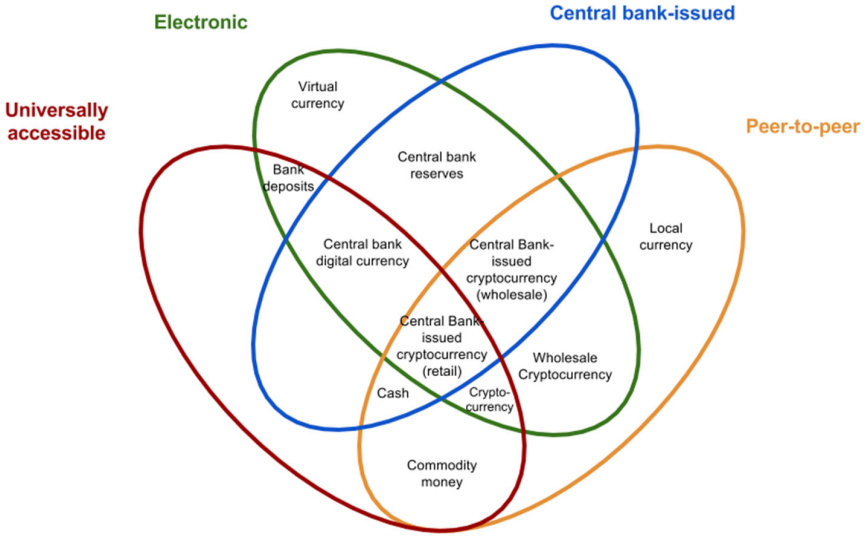

Fig. 1. Digital Currency Forms

Blockchain is a promising and widely adopted technology that offers essential qualities in network communications, including decentralization, transparency, security, and trust in a peer-to-peer manner. It has gained popularity for large-scale implementation as it reduces the risk of cyber-attacks leading to network failures. Privacy and security concerns surrounding Blockchain are crucial in various industries and are integral to the existence of cryptocurrencies (Amy et al., 2018). The KNN (K-Nearest Neighbors) algorithm is a supervised learning tool used for both classification and regression tasks. KNN makes predictions based on the number (k) of closest training samples. If data points have varying scales, it is necessary to normalize the data to improve accuracy, as the algorithm relies on the distance between data points. The KNN algorithm assigns weights to neighbors, with closer neighbors contributing more to the average than those farther away (Adeniyi et al., 2016). KNN is a powerful and straightforward algorithm that requires no complex parameter tuning and is particularly efficient for large training datasets. It is considered a "lazy learner" algorithm since it lacks a dedicated training phase. The decision tree (DT) algorithm is one of the most powerful data mining techniques and is widely used in scenarios involving substantial amounts of data. It excels in its ability to handle both numerical and nominal data (Li et al., 2014; Samara et al., 2021; Al-Fayoumi et al., 2024). The DT algorithm can also handle incorrect or missing information. As previously mentioned, the widespread interest in cryptocurrencies has made them a primary target for systematic attacks by unidentified individuals or groups using malware. These attacks often involve denial of service (DoS) and cryptocurrency mining on targeted servers. Traditional malware types employ common attack tactics and evade detection, highlighting the need for better solutions in malware detection. Code obfuscation, for instance, can mask malware signatures, making them difficult to identify. Machine learning techniques can aid in the identification and combating of such viruses (Darabian et al.,2020; Musa et al., 2023; Alzyoud et al., 2024). In a study by the authors (Yuan et al., 2020; Vasan et al., 2020), grayscale photographs were replaced with Markov images to improve the accuracy of Convolutional Neural Networks (CNN). Before training the CNN, malware samples were converted to Markov images. The classification approach was evaluated using the Drebin and Microsoft datasets. Microsoft's dataset consists of 10,868 malware samples from nine different families, while Drebin's dataset contains 4,020 malware samples from ten families. According to the authors, the suggested technique achieved an average accuracy of 97.364 percent

on the Drebin dataset. Different techniques for processing the dataset while employing the same classifier algorithm resulted in varying classification accuracy. In another study (Danish et al.,2020), CNN was used to classify malware samples, but this time the malware samples were transformed into color images. The authors utilized the Malimg dataset and an IoT-Android smartphone dataset. The classification accuracy for the Malimg dataset was 98.82 percent, and for the CNN's IoT-Android mobile dataset, it was 97.35 percent. A CNN model was used to classify malware samples in this investigation. The Microsoft and Malimg datasets were employed, with Malimg containing 9,339 malware samples from 25 different families, and the Microsoft dataset containing 21,741 malware samples from nine families. CNN's classification performed better on the Microsoft dataset than on the Malimg dataset. In yet another study (Al-rimy et al., 2019), the authors created an ensemble-based crypto-ransomware detection model by combining the semi-random subspace selection (ESRS) technique with incremental bagging. They compared their results with several other classifiers such as AdaBoost, Random Forests (RF), Decision Tree (DT), Linear Regression (LR), k-Nearest Neighbors (kNN), and Support Vector Machine (SVM). When incorporating twenty features into the proposed system, the accuracy was around 0.97. In a study by the authors (Baldwin et al., 2018; Bzdok & Meyer-Lindenberg, 2018), static analysis was employed using SVM and machine learning techniques to categorize five crypto-ransomware families and legitimate software. Opcode features were extracted to assist in the learning process. The findings demonstrated an accuracy of 96.5 percent. Lastly, Li Chen et al. (2018) developed a Generative Adversarial Network (GAN) that automatically extracts dynamic properties from malware samples. These properties were implemented in several classifiers, including Extreme Gradient Boosting (XGB), Linear Discriminant Analysis (LDA), Random Forests, Naive Bayes, and Support Vector Machine (SVM). The accuracy of the suggested approach on a given dataset was 99 percent. In a study by the authors (Takeuchi et al., 2018), dynamic analysis was employed to detect malware by examining the API call history of a Sandbox. API calls were gathered as malware features, and SVM was used to classify a dataset consisting of 312 goodware and 276 malware files. The trials demonstrated an accuracy of approximately 97.48 percent. In another study (Dey, 2018), powerful machine learning algorithms were used to calculate the percentage of cybercriminal enterprises in the Bitcoin ecosystem. A dataset of 854 observations divided across 12 classes and 100,000 unclassified observations was compiled. Thirteen supervised learning classifiers were assessed using the cross-validation method. The classifiers with the highest precision were reported to have accuracy rates of 77.38 percent, 76.47 percent, 78.46 percent, and 80.76 percent. Using machine learning and game theory approaches, the authors (Md et al., 2018) conducted research on securing majority attacks in Blockchain. They developed a method for users to employ intelligent software to monitor the activity of stakeholders in Blockchain networks in order to identify abnormalities such as compromise. Algorithmic game theory and supervised machine learning were utilized to mitigate most attacks. Alhawi et al. (2018) proposed NetConver, an ML technique for analyzing Windows ransomware in network traffic. When using NetConver with the Decision Tree (J48) classifier, a True Positive Detection Rate (TPR) value of 97.1 percent accuracy was achieved. The research experiment consisted of three phases: dataset collection by capturing malicious network traffic, focusing primarily on Windows ransomware in the dataset, and deriving five distinct characteristics from the collected network data using the TShark application. The proposed method was compared against Bayes Network, Random Forest, Multilayer Perceptron, k-nearest neighbors (KNN), and Least Mean Squares Filter (LMS) using TPR and FPR. According to the comparison findings, the proposed algorithm outperformed the other five classifier algorithms. Homayoun et al. (2017) utilized machine learning with sequential pattern mining to discover the ideal sequential patterns (MSP). The dataset consisted of 220 examples of benign software and 1624 cases of malicious software. J48, Random Forest, Bagging, and MLP classifiers were employed in the study. The accuracy of their findings was reported to be 99 percent. In (Yuan et al., 2016), the authors proposed Invincea, a deep neural network-based Android malware detection technique. It achieves a useful detection rate with a low false-positive rate and is scalable to a high number of training cases on commodity hardware. Based on 400,000 software binaries, their method achieved a false-positive rate (FPR) of 0.1 percent and a detection rate of 95 percent.

3. Framework for Malware Detection in Cryptocurrency

This section examines the architecture of typical algorithms used to identify malware infecting cryptocurrency using machine learning and related studies. Fig. 2 shows the workflow used in this article, which comprises the following steps: network traffic flow and data collection, feature extraction, malware classification using machine learning, and malware detection decisions in cryptocurrency.

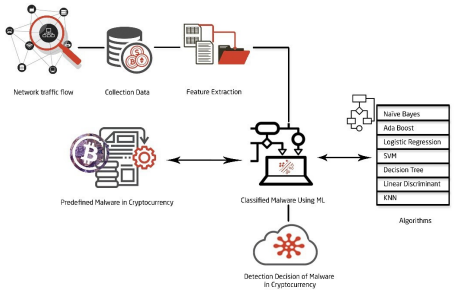

Fig. 2. Network Traffic Flow and Collection Data

In this work, standard machine learning (ML) models for detecting malware using static and dynamic features were applied. It is crucial to train ML algorithms on data that adequately simulates real-world model settings, and therefore, a collection of representative datasets was chosen. These datasets consist of both malicious and benign data. Importantly, the use of these datasets does not compromise users' privacy as they are publicly available on GitHub and Kaggle. However, it is worth noting that the malware dataset is secondary in comparison to the non-malicious data collection. To determine the outcomes for each group, three datasets were utilized. The first dataset comprises 5,184 benign malwares with 69 features, the second dataset includes 19,611 benign malwares with 79 features, and the third dataset consists of 47,547 benign malwares with 1,002 features. Numeric characteristics were employed and evaluated using the standard scaling approach. These characteristics, along with the target variable (category 0 for benign and 1 for malware), were used to develop the classification model.

4. Experimental results

To assess the classification potential of the proposed ensemble learning model, its performance was evaluated using the Leave-One-Out Cross-Validation (LOOCV) approach with 10-fold cross-validation. By comparing the actual labels to the predicted labels, the following values were determined: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). The classification metrics considered were “recall”, “precision”, “accuracy”, “error rate”, and “F-score”. In this context, the positive class is denoted by “+1”, while the negative class is denoted by “0”.

4.1 Performance Metrics

Several performance metrics have been utilized to evaluate the performance of the models used in the proposed system. These metrics include the confusion matrix, accuracy, precision, recall, F-measure, receiver operating characteristic (ROC), and root mean square error (RMSE).

4.1.1 Confusion Matrix

The confusion matrix is a (N × N) matrix used to evaluate and display the performance of a classification method, where N represents the number of classes in a dataset. The confusion matrix provides a visual representation of the actual and predicted values. It consists of four values that determine the measuring metrics of the classifier (Ajay et al., 2020). Table 1 presents an example of a confusion matrix:

Table 1
Confusion Matrix

	Actual: positive	Actual: Negative
Predicted: positive	TP	FP
Predicted: negative	Naïve Bayes	0.699

where TP (True Positive) represents the number of positive examples that have been correctly classified by the classifier, TN (True Negative) indicates the number of negative examples that have been correctly classified by the classifier, FP (False Positive) represents the number of negative examples that have been incorrectly classified as positive by the classifier, and FN (False Negative) indicates the number of positive examples that have been incorrectly classified as negative by the classifier.

4.1.2 Accuracy

Accuracy is used to measure the effectiveness of a classifier. It represents the percentage of correctly identified instances relative to the total sample size. Accuracy is particularly useful in symmetric datasets where the number of false negatives (FN) and false positives (FP) are equal (Paquet-Clouston et al., 2019). Eq. (1) demonstrates the calculation of accuracy:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

4.1.3 Precision

Precision reflects the ratio of positive instances that are correctly classified to the total number of positive examples classified (Maxmen, 2018). Eq. (2) demonstrates the calculation of precision:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

4.1.4 Recall

Recall, also known as sensitivity or true positive rate, represents the proportion of correctly identified positive cases relative to the total number of actual positive examples (Alam et al., 2018). The recall is calculated using Eq. (3):

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

4.1.5 F-measure

The F-measure, also known as the F1-score, is a metric that combines the accuracy and recall of a model using a weighted harmonic mean (Hand et al., 2021). Eq. (4) demonstrates the calculation of the F-measure:

$$f\_measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

4.1.6 Receiver Operating Characteristic (ROC)

The ROC (Receiver Operating Characteristic) is a graph that illustrates the trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR). It is commonly used to evaluate the performance of a classifier at different thresholds. By plotting the TPR against the FPR, the resulting ROC curve provides insights into the classifier's accuracy. The area under the ROC curve (AUC) represents the region beneath the curve and serves as a measure of the classifier's performance (Bowers & Zhou, 2019). Eq. (5) depicts the calculation of TPR, and Eq. (6) represents the calculation of FPR.

$$TRP = \frac{TP}{TP + FN} \tag{5}$$

$$FPR = \frac{FP}{FP + TN} \tag{6}$$

4.1.7 Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) is a commonly used metric for evaluating numerical predictions. It is calculated as the square root of the prediction error's standard deviation (Al-Daoud et al., 332007; Samara et al., 2023). Eq. (7) illustrates the calculation of RMSE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(S_i - 0_i)^2} \tag{7}$$

4.2 Classes Distribution

Utilizing three multi-file datasets, the number of malware and benign software instances were estimated for each dataset. Table 2 displays the total count of malicious and benign files in each dataset, calculated using the Python programming language. The findings reveal that the first dataset consists of 5,184 malicious files and 2,722 benign files, the second dataset includes 14,599 malicious files and 5,012 benign files, and the third dataset comprises 45,651 malicious files and 1,896 benign files, as depicted in Fig. 3.

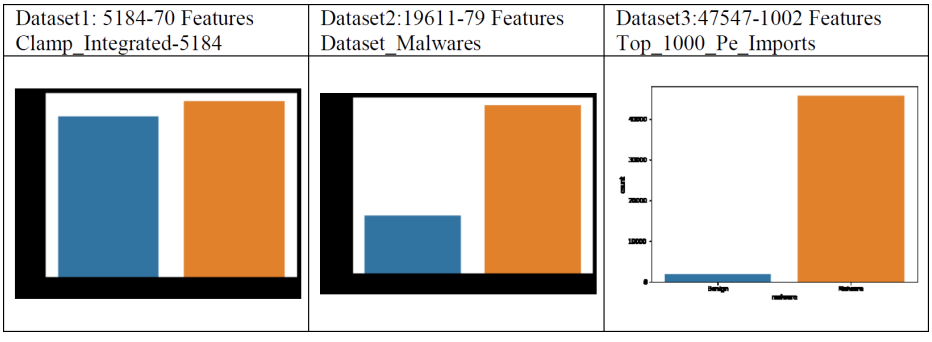

Fig. 3. Calculation of the Benign and Malware for Each Dataset

4.3.1 Dataset 1 Experiment

The ML algorithms were trained on the first dataset, as shown in Table 2. Figure 4 displays the results graph, indicating that the AdaBoost method achieved the highest accuracy of 98% and the lowest error rate of 13%. On the other hand, Naive Bayes yielded the poorest results.

Table 2
Algorithms Comparison on Dataset 1

Classifier Algorithms	Accuracy	F1_core	Recall	precision	RM E
Naïve Bayes	0.699	0.673	0.699	0.798	0.548
Ada boost	0.980	0.980	0.980	0.980	0.138
Logistic Regression	0.955	0.955	0.955	0.955	0.210
VM	0.960	0.960	0.960	0.960	0.199
Decision Tree	0.971	0.971	0.971	0.971	0.168
Linear Discriminant	0.923	0.923	0.923	0.924	0.277
KNN	0.971	0.971	0.971	0.971	0.168
CNN	0.97	0.97	0.97	0.97	0.158

4.3.2 Dataset 2 Experiment

The ML algorithms were trained on the second dataset, as shown in Table 3. Fig. 5 depicts the graph of the results. The decision tree method gave the best results, with an accuracy of 98.4% and an error rate of 12%. The worst results were achieved using Naive Bayes.

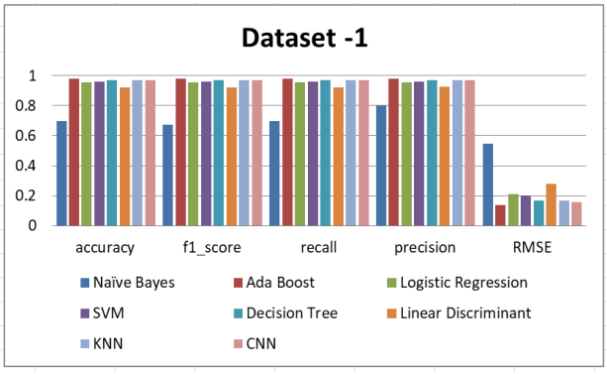

Fig. 4. Algorithms Comparison on dataset 1

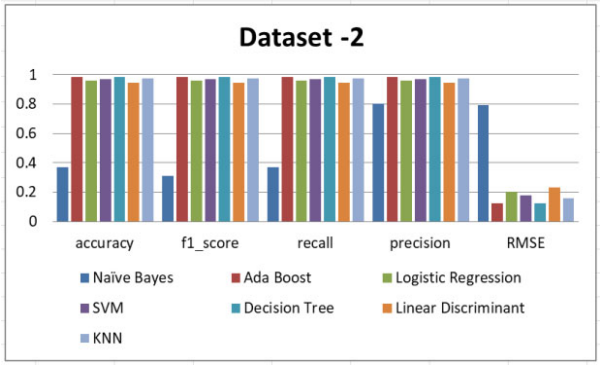

Fig. 5. Algorithms Comparison on dataset 2

Table 3
Algorithms Comparison on dataset 2

Classifier Algorithms	Accuracy	F1_core	Recall	Precision	RM E
Naïve Bayes	0.372	0.313	0.372	0.800	0.792
Ada Boost	0.9840	0.984	0.984	0.984	0.126
Logistic Regression	0.958	0.958	0.958	0.958	0.203
VM	0.967	0.967	0.967	0.967	0.179
Decision Tree	0.9842	0.984	0.984	0.984	0.125
Linear Discriminant	0.945	0.944	0.945	0.944	0.233
KNN	0.975	0.975	0.975	0.975	0.157

4.3.3 Dataset 3 Experiment

The ML algorithms were trained on the second dataset, as indicated in Table 4. Fig. 6 depicts the graph of the results obtained, with the KNN method achieving the highest accuracy of 97.8% and the lowest error rate of 14.4%. The worst results were achieved using Naive Bayes.

Table 4
Algorithms Comparison on dataset 3

Classifier Model	Accuracy	F1_core	Recall	Precision	RM E
Naïve Bayes	0.559	0.681	0.559	0.956	0.663
Ada Boost	0.972	0.969	0.972	0.969	0.165
Logistic Regression	0.976	0.974	0.976	0.974	0.153
VM	0.977	0.975	0.977	0.975	0.149
Decision Tree	0.9780	0.976	0.978	0.976	0.148
Linear Discriminant	0.975	0.974	0.975	0.973	0.156
KNN	0.9789	0.977	0.978	0.977	0.144

4.4 Results Analysis

ML models were trained using Python libraries and features extracted from several datasets. All experiments were conducted on a 64-bit laptop running Windows 10. In this manner, the outcomes of employing traditional ML models based on the

dataset's features are displayed. The expected data results for each algorithm across all datasets are as follows: a value of 1 indicates a malicious program, and a value of 0 indicates a benign program.

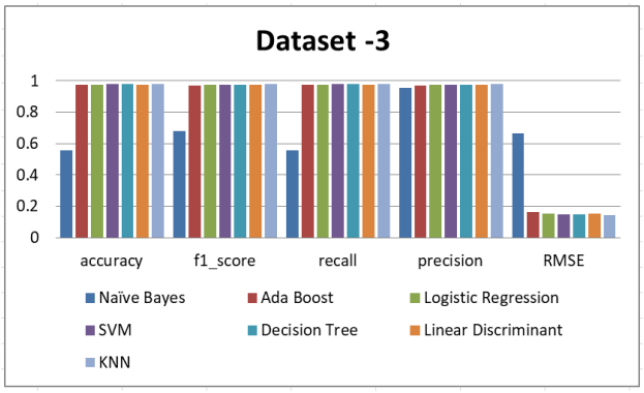

Fig. 6. Algorithms Comparison on dataset 3

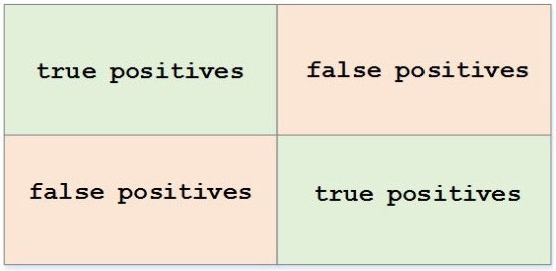

Fig. 7. Confusion Matrix

After the data has been cleaned and processed, the first stage is to feed it into an outstanding model and obtain probability outcomes. The Confusion Matrix is then implemented as a performance measure for ML classification.

The Confusion Matrix, shown in Fig. 7, is used to analyze and explain the results of the classification algorithm. The figure below depicts confusion matrices for each algorithm deployed on the selected datasets, where benign software is referred to as "healthy" and malicious software as "malicious".

The confusion matrix comprises four attributes with numeric values that determine the classifier's measurements. These four digits represent:

TP reflects the number of programs that the classifier has accurately classified as malware, indicating that they are malware.

TN, or true negative, is the number of correctly classified non-threatening programs.

FP shows the number of programs misclassified as malicious when they are benign. FP is referred to as a type I error.

The number of malicious applications incorrectly classified as benign is reflected in False Negative (FN). FN is also known as a type II error.

The F1 score is the algorithm's score, determined by its performance in previous tests. Algorithms are evaluated based on how well they categorize programs (TP + TN) and all programs (TP + TN + FP + FN). The algorithm's accuracy is measured as the ratio of programs accurately identified as malware (TP) to the total number of programs estimated to be hazardous (TP + FP).

The recall metric measures the proportion of correctly identified malicious software (TP) divided by the total amount of malicious software. It is also known as the true positive rate. Precision is based on the number of correct malicious software classifications. The F1 score represents the balance between recall and precision.

Overall, the confusion matrix and associated metrics are used to evaluate the performance of classification algorithms in detecting malware.

4.4.1 Results Analysis for Dataset1

In Fig. 8, the accuracy percentage was computed for each algorithm used on the initial dataset, and the results were as follows:

For the anticipated result of 641, 632 were correctly identified as positive, and 9 were mistakenly labeled as positive.

For the predicted result of -662, 383 were incorrectly categorized, and 279 were correctly classified, resulting in the following accuracy ratio: Ac = (632+279)/(632+9+383+279) = 0.699.

AdaBoost:

The predicted outcome is 641, with 627 correctly classified as positive and 14 incorrectly classified as positive. The accuracy ratio was as follows:

For the predicted result of -662, 11 were incorrectly categorized, and 651 were correctly categorized. Ac = (627+651)/(627+14+11+651) = 0.980.

Logistic Regression:

The predicted positive outcome is 641, of which 610 were correctly classified as positive and 31 were incorrectly classified as positive.

The accuracy ratio was as follows:

For the predicted result of -662, 27 were incorrectly categorized, and 635 were successfully categorized. Ac = (610+635)/(610+31+27+635) = 0.955.

Linear Discriminant:

The predicted result is 641, with 570 correctly classified and 71 incorrectly classified as positive. The accuracy ratio was as follows:

For the predicted result of -662, 29 were incorrectly categorized, and 633 were correctly categorized. Ac = (570+633)/(570+71+29+633) = 0.920.

Support Vector Machine (SVM):

The predicted result is 641, with 609 correctly classified as positive and 32 incorrectly classified as positive. The predicted result of -662 had 20 incorrectly classified and 642 correctly classified, yielding the following accuracy ratio:

Ac = (609+642)/(609+32+20+642) = 0.960.

The AdaBoost method resulted in the highest accuracy for the first dataset.

The results show the accuracy of different machine learning algorithms on an initial dataset. The accuracy ratio (Ac) was computed for each algorithm for both the anticipated and predicted results. The AdaBoost algorithm achieved the highest accuracy of 0.980 for the anticipated result, while for the predicted result, the logistic regression algorithm achieved the highest accuracy of .0.955. The linear discriminant algorithm achieved an accuracy of 0.920, and the SVM algorithm achieved an accuracy of 0.960 for the predicted result.

It is important to note that the accuracy of results may vary depending on the dataset and the specific problem being solved. Therefore, it is recommended to test multiple algorithms and evaluate their performance to determine the most effective one for a given problem. In this case, the AdaBoost algorithm was the most accurate for the initial dataset.

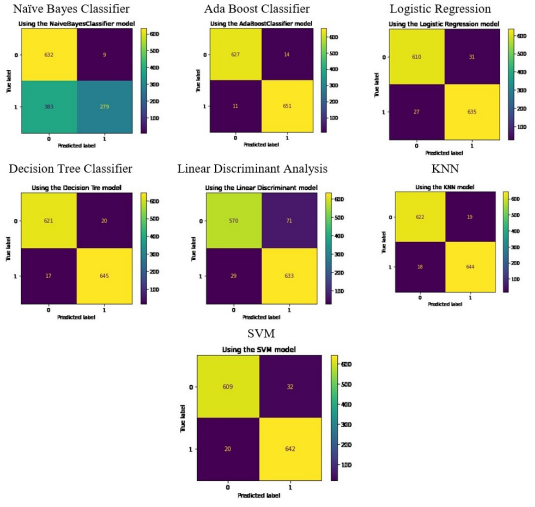

Fig. 8. Confusion Matrix for Dataset1

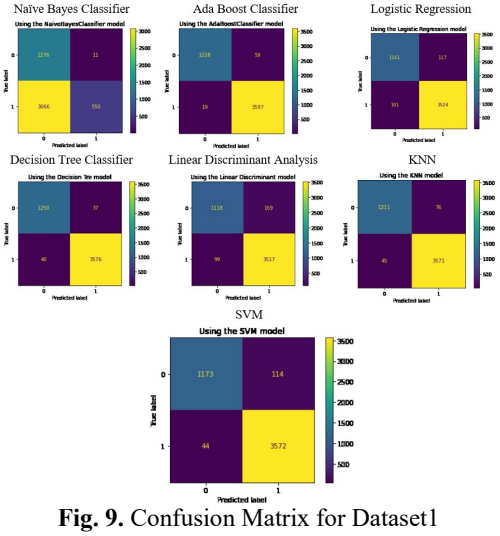

Fig. 9. Confusion Matrix for Dataset1

In Fig. 9, the accuracy percentage for each algorithm used on the second dataset was determined and listed as follows:

Random Forest: The anticipated result is positive 1287, of which 1276 were correctly classified as positive, and 11 were incorrectly classified as positive. The accuracy ratio was as follows: The expected result was a negative 3616, of which 3066 were incorrectly categorized, and 550 were correctly categorized. Ac = (1276+550)/(1276+11+3066+550) = 0.372.

Ada Boost: The expected result is positive 1287, of which 1228 were classified correctly, and 59 were incorrectly classified as positive. The accuracy ratio was as follows: The expected result was negative 3616, of which 19 were incorrectly categorized, and 3597 were correctly categorized. Ac = (1228+3597)/(1228+59+19+3597) = 0.984.

Logistic Regression: The expected result is 1287 positives, of which 1161 were correctly identified, and 117 were incorrectly classified as positives. The accuracy ratio was as follows: The expected result was negative 3616, of which 101 were incorrectly categorized, and 3524 were correctly categorized. Ac = (1161+3524)/(1161+117+101+3524) = 0.958.

Linear Discriminant: The expected result is positive 1287, with 1250 instances correctly classified and 37 instances incorrectly classified as positive. Ac = (1250 + 3576) / (1250 + 37 + 40 + 3576) = 0.984%. The expected result is positive 1287, of which 1118 were correctly identified, and 169 were incorrectly classified as positive. The accuracy ratio was as follows: The expected result was negative 3616, of which 99 were incorrectly categorized, and 35517 were correctly categorized. Ac = (1118+3517)/(1118+169+99+3517) = 0.945.

KNN: The expected outcome is 1287 positive results, of which 1211 were correctly identified, and 76 were incorrectly classified as positive. The accuracy ratio was as follows: The predicted result was negative 3616, of which 45 were incorrectly categorized, and 3571 were correctly categorized. Ac = (1211+3571)/(1211+76+45+3571) = 0.975%.

Using the Random Forest approach gave the highest accuracy for the first dataset.

The results in Figure 9 show the accuracy of each algorithm when applied to the second dataset. Accuracy is determined by comparing the expected results with the actual results obtained by each algorithm.

Based on the accuracy ratios, it can be seen that Ada Boost had the highest accuracy (0.984), followed by Linear Discriminant (0.984%) and KNN (0.975%). Logistic Regression had an accuracy of 0.958, while the Random Forest approach had the highest accuracy for the first dataset.

It is important to note that accuracy alone may not be sufficient to evaluate the performance of an algorithm. Other metrics, such as precision, recall, and F1 score, should also be considered to have a more comprehensive understanding of the algorithm's performance. Additionally, the size and quality of the dataset, the complexity of the problem, and other factors can also impact the performance of the algorithm.

4.4.3 Results Analysis for Dataset 3

In Fig. 10, the accuracy percentages for each algorithm used in the previous dataset are presented:

Random Forest: The expected positive outcome is 478, of which 437 were correctly classified, and 41 were incorrectly classified as positive. The predicted negative outcome was 11,417, of which 6,222 were correctly categorized, and 5,195 were incorrectly categorized, resulting in the following accuracy ratio: Ac = (437+6,222)/(437+41+5,195+6,222) = 0.559.

Ada Boost: For Ada Boost, the expected positive result is 478, of which 229 were correctly classified, and 249 were incorrectly classified as positive. The predicted negative outcome was 11,417, of which 11,341 were correctly categorized, and 76 were incorrectly categorized. The accuracy ratio was: Ac = (229+11,341)/(229+249+76+11,341) = 0.972.

Logistic Regression: For Logistic Regression, the expected positive result is 478, of which 271 were correctly classified, and 207 were incorrectly classified as positive. The predicted negative outcome was 11,417, of which 11,342 were correctly categorized, and 75 were incorrectly categorized. The accuracy ratio was: Ac = (271+11,342)/(271+207+75+11,342) = 0.976.

Linear Discriminant: The expected positive outcome is 478, of which 301 were correctly classified, and 117 were incorrectly classified as positive. The predicted negative outcome was 11,417, of which 11,333 were correctly categorized, and 84 were incorrectly categorized. The accuracy ratio was: Ac = (301+11,333)/(301+117+84+11,333) = 0.978.

KNN: For KNN, the expected positive outcome is 478, of which 297 were correctly classified, and 199 were incorrectly classified as positive. The predicted negative outcome was 11,417, of which 11,326 were correctly categorized, and 91 were incorrectly categorized. The accuracy ratio was: Ac = (297+11,326)/(297+199+91+11,326) = 0.975.

SVM: For SVM, the expected positive outcome is 478, of which 262 were correctly classified, and 216 were incorrectly classified as positive. The predicted negative outcome was 11,417, of which 11,366 were correctly categorized, and 51 were incorrectly categorized. The accuracy ratio was: Ac = (262+11,366)/(262+216+51+11,366) = 0.97.

Using the Linear Discriminant algorithm gave the highest accuracy for the third dataset.

The results show the accuracy percentages for each algorithm used in the previous dataset. The evaluated algorithms were Random Forest, Ada Boost, Logistic Regression, Linear Discriminant, KNN, and SVM. The highest accuracy percentage was achieved by the Linear Discriminant algorithm with an accuracy ratio of 0.978, followed by Logistic Regression and KNN with accuracy ratios of 0.976 and 0.975, respectively. SVM had the lowest accuracy ratio of 0.97.

The results also show that the expected positive outcomes were classified correctly in all algorithms, with the exception of KNN, which had 199 false positives. In terms of false negatives, Ada Boost had the lowest number (249), followed by Logistic Regression (207) and KNN (199). SVM had the highest number of false negatives with 216.

Overall, the results indicate that Linear Discriminant is the most accurate algorithm for this dataset, followed closely by Logistic Regression and KNN. However, it is important to note that the choice of algorithm should be based on the specific needs and characteristics of the dataset, and not solely on accuracy.

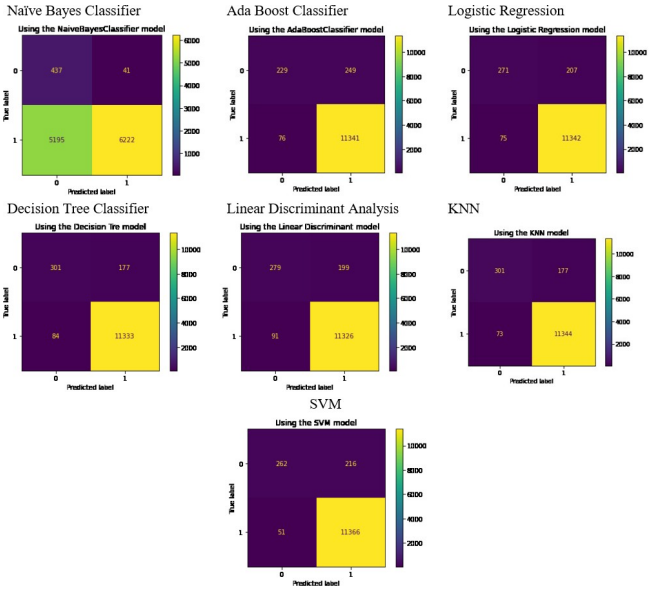

Fig. 10. Results Analysis for Dataset 3

5. Conclusion and Future Works

Due to the contemporary significance of cryptocurrencies, there is a growing interest in protecting them and averting attacks. This study presents a method based on ML algorithms to discover and classify malicious programs in cryptocurrencies with high accuracy. These programs are classified as "benign" or "malicious." Naive Bayes, AdaBoost, Logistic Regression, SVM, Decision Tree, and Linear Discriminant algorithms are used for this purpose. The investigations made use of three datasets: top_1000_pe_imports, malware, and ClaMP Integrated-5184.

The results demonstrated that the proposed framework achieved good results on all three datasets. Specifically, on the first dataset, the AdaBoost approach outperformed other classifiers. While the decision tree method outperformed other methods in the second dataset, the KNN method generated the best results in the third dataset. When comparing the findings of the proposed algorithm, it was found that the results were dependent on the size of the dataset. The methods on the small dataset produced the greatest results for the AdaBoost model, while the decision tree model produced the best results on the medium-sized dataset. The KNN model produced the best results for the largest dataset. On the other hand, the Naive Bayes experiments consistently produced the worst outcomes.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016, October). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 308-318).

Adeniyi, D. A., Wei, Z., & Yongquan, Y. (2016). Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method. *Applied Computing and Informatics*, *12*(1), 90-108.

Alam, M. S., Husain, D., Naqvi, S., & Kumar, P. (2018). IOT security through Machine Learning and homographic encryption technique. In *International conference on new trends in engineering & technology (ICNTET), Chennai*.

Alazaidah, R., Al-Shaikh, A., Al-Mousa, M. R., Khafajah, H., Samara, G., Alzyoud, M., & Almatarneh, S. (2024). Website phishing detection using machine learning techniques. *Journal of Statistics Applications & Probability*, *13*(1), 119-129.

Alazaidah, R., Samara, G., Aljaidi, M., Haj Qasem, M., Alsarhan, A., & Alshammari, M. (2023a). Potential of Machine Learning for Predicting Sleep Disorders: A Comprehensive Analysis of Regression and Classification Models. *Diagnostics*, *14*(1), 27.

Alazaidah, R., Samara, G., Almatarneh, S., Hassan, M., Aljaidi, M., & Mansur, H. (2023b). Multi-label classification based on associations. *Applied Sciences*, *13*(8), 5081.

Al-Daoud, E. (2007). Quantum Computing for Solving a System of Nonlinear Equations over GF (q). *Int. Arab J. Inf. Technol.*, *4*(3), 201-205.

Al-Fayoumi, M., Al-Haija, Q. A., Armoush, R., & Amareen, C. (2024). XAI-PDF: a robust framework for malicious PDF detection leveraging SHAP-based feature engineering. *Int. Arab J. Inf. Technol.*, *21*(1), 128-146.

Al-Haija, Q. A., & Alsulami, A. A. (2021). High performance classification model to identify ransomware payments for heterogeneous bitcoin networks. *Electronics*, *10*(17), 2113.

Alhawi, O. M., Baldwin, J., & Dehghantanha, A. (2018). Leveraging machine learning techniques for windows ransomware network traffic detection. *Cyber threat intelligence*, 93-106.

Aljaidi, M., Alsarhan, A., Samara, G., Alazaidah, R., Almatarneh, S., Khalid, M., & Al-Gumaei, Y. A. (2022, November). NHS WannaCry ransomware attack: technical explanation of the vulnerability, exploitation, and countermeasures. In *2022 International Engineering Conference on Electrical, Energy, and Artificial Intelligence (EICEEAI)* (pp. 1-6). IEEE.

Al-rimy, B. A. S., Maarof, M. A., & Shaid, S. Z. M. (2019). Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. *Future Generation Computer Systems*, *101*, 476-491.

Al-Shanableh, N., Al-Zyoud, M., Al-Husban, R. Y., Al-Shdayfat, N., Alkhawaldeh, J. F. M., Alajarmeh, N. S., & Al-Hawary, S. I. S. (2024). Data Mining to Reveal Factors Associated with Quality of life among Jordanian Women with Breast Cancer. *Applied Mathematics*, *18*(2), 403-408.

Alzyoud, M., Al-Shanableh, N., Alomar, S., AsadAlnaser, A., Mustafad, A., Al-Momani, A., & Al-Hawary, S. (2024). Artificial intelligence in Jordanian education: Assessing acceptance via perceived cybersecurity, novelty value, and perceived trust. *International Journal of Data and Network Science*, *8*(2), 823-834.

Bahrami, P. N., Dehghantanha, A., Dargahi, T., Parizi, R. M., Choo, K. K. R., & Javadi, H. H. (2019). Cyber kill chain-based taxonomy of advanced persistent threat actors: Analogy of tactics, techniques, and procedures. *Journal of information processing systems*, *15*(4), 865-889.

Baldwin, J., & Dehghantanha, A. (2018). Leveraging support vector machine for opcode density based detection of crypto-ransomware. *Cyber threat intelligence*, 107-136.

Bowers, A. J., & Zhou, X. (2019). Receiver operating characteristic (ROC) area under the curve (AUC): A diagnostic measure for evaluating the accuracy of predictors of education outcomes. *Journal of Education for Students Placed at Risk (JESPAR)*, *24*(1), 20-46.

Bzdok, D., & Meyer-Lindenberg, A. (2018). Machine learning for precision psychiatry: opportunities and challenges. *Biological Psychiatry: Cognitive Neuroscience and Neuroimaging*, *3*(3), 223-230.

Chen, L., Yang, C. Y., Paul, A., & Sahita, R. (2018). Towards resilient machine learning for ransomware detection. *arXiv preprint arXiv:1812.09400*.

Darabian, H., Homayounoot, S., Dehghantanha, A., Hashemi, S., Karimipour, H., Parizi, R. M., & Choo, K. K. R. (2020). Detecting cryptomining malware: a deep learning approach for static and dynamic analysis. *Journal of Grid Computing*, *18*, 293-303.

Dey, S. (2018, September). Securing majority-attack in blockchain using machine learning and algorithmic game theory: A proof of work. In *2018 10th computer science and electronic engineering (CEEC)* (pp. 7-10). IEEE.

Hand, D. J., Christen, P., & Kirielle, N. (2021). F*: an interpretable transformation of the F-measure. *Machine Learning*, *110*(3), 451-456.

Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., & Khayami, R. (2017). Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence. *IEEE transactions on emerging topics in computing*, *8*(2), 341-351.

Hussain, I., Samara, G., Ullah, I., & Khan, N. (2021, December). Encryption for end-user privacy: A cyber-secure smart energy management system. In *2021 22nd International Arab Conference on Information Technology (ACIT)* (pp. 1-6). IEEE.

Kulkarni, A., Chong, D., & Batarseh, F. A. (2020). Foundations of data imbalance and solutions for a data democracy. In *Data democracy* (pp. 83-106). Academic Press.

Li, L., Ma, S., & Zhang, Y. (2014, December). Optimization algorithm based on genetic support vector machine model. In *2014 Seventh International Symposium on Computational Intelligence and Design* (Vol. 1, pp. 307-310). IEEE.

Liu, W., Liang, X., & Cui, G. (2020). Common risk factors in the returns on cryptocurrencies. *Economic Modelling*, *86*, 299-305.

Maxmen, A. (2018). AI researchers embrace Bitcoin technology to share medical data. *Nature*, *555*(7697), 293-295.

Musa, N. S., Mirza, N. M., & Ali, A. (2023). Navigating the Complex Landscape of IoT Forensics: Challenges and Emerging Solutions.

Paquet-Clouston, M., Haslhofer, B., & Dupont, B. (2019). Ransomware payments in the bitcoin ecosystem. *Journal of Cybersecurity*, *5*(1), tyz003.

Samara, G. (2020, November). Wireless sensor network MAC energy-efficiency protocols: a survey. In *2020 21st International Arab Conference on Information Technology (ACIT)* (pp. 1-5). IEEE.

Samara, G., & Al-okour, M. (2020). Optimal number of cluster heads in wireless sensors networks based on LEACH. *arXiv preprint arXiv:2003.13765*.

Samara, G., Al-Daoud, E., Swerki, N., & Alzu'bi, D. (2023). The recognition of holy qur'an reciters using the mfccs' technique and deep learning. *Advances in Multimedia*, *2023*(1), 2642558.

Samara, G., Besani, G. A., Alauthman, M., & Khaldy, M. A. (2020). Energy-efficiency routing algorithms in wireless sensor networks: A survey. *arXiv preprint arXiv:2002.07178*.

Samara, G., Elhilo, A., Asassfeh, M. R., Injadat, M., Qasem, M. H., Alazaidah, R., & Hnaif, A. A. (2023, December). Optimizing Road Safety through Intelligent Congestion Management. In *2023 24th International Arab Conference on Information Technology (ACIT)* (pp. 1-7). IEEE.

Takeuchi, Y., Sakai, K., & Fukumoto, S. (2018, August). Detecting ransomware using support vector machines. In *Workshop Proceedings of the 47th International Conference on Parallel Processing* (pp. 1-6).

Vasan, D., Alazab, M., Wassan, S., Naeem, H., Safaei, B., & Zheng, Q. (2020). IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks*, *171*, 107138.

Yazdinejad, A., HaddadPajouh, H., Dehghantanha, A., Parizi, R. M., Srivastava, G., & Chen, M. Y. (2020). Cryptocurrency malware hunting: A deep recurrent neural network approach. *Applied Soft Computing*, *96*, 106630.

Yin, M., Wortman Vaughan, J., & Wallach, H. (2019, May). Understanding the effect of accuracy on trust in machine learning models. In *Proceedings of the 2019 chi conference on human factors in computing systems* (pp. 1-12).

Yuan, B., Wang, J., Liu, D., Guo, W., Wu, P., & Bao, X. (2020). Byte-level malware classification based on markov images and deep learning. *Computers & Security*, *92*, 101740.

Yuan, Z., Lu, Y., & Xue, Y. (2016). Droiddetector: android malware characterization and detection using deep learning. *Tsinghua Science and Technology*, *21*(1), 114-123.