Journal of King Saud University - Computer and Information Sciences 36 (2024) 101925

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University - Computer and Information Sciences

journal homepage: www.sciencedirect.com

Full length article

Machine learning-based ransomware classification of Bitcoin transactions

Omar Dib [a,b,*], Zhenghan Nan [c], Jinkua Liu [d]

[a] Department of Computer Science, Wenzhou-Kean University, 88 Daxue Rd, Ouhai, Wenzhou 325060, Zhejiang, China
[b] Department of Computer Science, Kean University, 1000 Morris Avenue, Union 07083, NJ, USA
[c] Computer Science Department at the Courant Institute of Mathematical Sciences, a part of New York University, NY 10012, USA
[d] College of Computing, Georgia Institute of Technology, North Avenue, Atlanta, GA, 30332, USA

ARTICLE INFO

Keywords:
Ransomware detection
Cryptocurrency transactions
BitcoinHeist dataset
Machine learning methods
Anomaly detection

ABSTRACT

Ransomware presents a significant threat to the security and integrity of cryptocurrency transactions. This research paper explores the intricacies of ransomware detection in cryptocurrency transactions using the Bitcoinheist dataset. The dataset encompasses 28 distinct families classified into three ransomware categories: Princeton, Montreal, and Padua, along with a white category representing legitimate transactions. We propose a novel hybrid supervised and semi-supervised multistage machine learning framework to tackle this challenge. Our framework effectively classifies known ransomware families by leveraging ensemble learning techniques such as Decision Tree, Random Forest, XGBoost, and Stacking. Additionally, we introduce a novel semi-supervised approach to accurately identify previously unseen ransomware instances within the dataset. Through rigorous evaluation employing comprehensive classification metrics, including accuracy, precision, recall, F1 score, RoC score, and prediction time, our proposed approach demonstrates promising results in ransomware detection within cryptocurrency transactions.

1. Introduction

Blockchain technology has given rise to various forms of digital currency, with Bitcoin emerging as a decentralized and widely recognized cryptocurrency (Dib et al., 2018). Its unregulated nature has attracted considerable attention from researchers and investors, but it has also become a target for criminal activities. In recent years, the Internet has witnessed a surge in a specific type of malware known as ransomware. This malicious software infiltrates a victim's computer system, encrypts their files and data, and demands a ransom, often in the form of Bitcoin, to restore access to the compromised data (Ruoti et al., 2019; Zhang et al., 2021). Ransomware attacks are orchestrated by distinct hacker groups, each associated with different variants or families of malware. The common characteristic of these attacks is the demand for payment in exchange for releasing the captured data (Kok et al., 2019).

Within the realm of ransomware are various types, with crypto-ransomware being the most virulent and aggressive. Crypto ransomware targets the victim's data by encrypting it, while locker ransomware locks the victim's computer, effectively denying access to the system. Of particular concern is the capability of crypto-ransomware to encrypt not only the user's files but also any files found on mapped and unmapped network drives. Consequently, the infection of a single system can halt an entire department or organization. Crypto ransomware can be categorized based on the employed cryptosystem into three types: Symmetrical (SCR), Asymmetrical (ACR), and Hybrid Cryptosystem Ransomware (HCR) (Al-rimy et al., 2018). SCR employs a symmetrical encryption algorithm such as DES or AES to encrypt the victim's files. The same key is used for encryption and decryption, allowing for the possibility of recovering the secret key through reverse engineering or memory scanning. The above cryptographic techniques can be improved using advanced tools and protocols such as Niasar et al. (2020) and Cintas-Canto et al. (2022a).

In ACR, the ransomware employs a public key embedded within the malware or obtained during communication with the command and control server to encrypt the victim's files. The attacker exclusively holds the corresponding private key, making it inaccessible to the victim without paying the ransom. However, this encryption technique

* Corresponding author at: Department of Computer Science, Kean University, 1000 Morris Avenue, Union 07083, NJ, USA.
E-mail addresses: odib@kean.edu (O. Dib), zn2145@nyu.edu (Z. Nan), jliu3142@gatech.edu (J. Liu).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2024.101925
Received 16 November 2023; Received in revised form 26 December 2023; Accepted 9 January 2024
Available online 11 January 2024

demands more computational resources during file encryption. HCR combines elements of both symmetrical and asymmetrical encryption. In HCR, a dynamically generated symmetric key is used to encrypt the victim's files, providing efficiency and speed in the encryption process. Subsequently, a pre-loaded public key is utilized to encrypt the symmetric key itself, and then it is immediately cleared from memory. This approach allows cybercriminals to leverage the advantages offered by both encryption types. These distinct ransomware cryptosystems exemplify the evolving strategies employed by cybercriminals to maximize the impact of their attacks while ensuring the encryption of victims' data. By utilizing hybrid techniques, ransomware authors seek to enhance their ability to evade detection and improve the efficiency of their encryption processes, making the fight against ransomware an ongoing challenge for cybersecurity professionals.

In addition, there have recently been active discussions among researchers and practitioners on the close intersection between crypto-ransomware and the advent of post-quantum computing, which introduces significant security concerns (Kermani and Azarderakhsh, 2016) across all digital sectors (Mozaffari Kermani et al., 2016). As quantum computing capabilities progress, the cryptographic algorithms safeguarding data may become susceptible to quantum-based decryption methods (Cintas-Canto et al., 2022b). This shift could potentially enable threat actors to compromise existing encryption standards, thereby increasing the efficacy of crypto-ransomware attacks. The underlying concern stems from the prospect of leveraging quantum-powered decryption techniques to breach and manipulate data encrypted using contemporary cryptographic protocols. Hence, urgent attention is needed to transition towards post-quantum cryptographic frameworks, strategically conceptualizing and fortifying against the imminent threats posed by crypto-ransomware in an era dominated by quantum computing advancements. For a comprehensive exploration of the profound implications of post-quantum computing on prevailing security paradigms and to investigate novel methodologies for mitigating resultant challenges, we encourage readers to explore the following papers: Joseph et al. (2022), Canto et al. (2023), Sarker et al. (2022), and Elkhatib et al. (2021).

Identifying and classifying ransomware transactions within the Bitcoin blockchain poses significant challenges due to the anonymity of cryptocurrency payments (Singh et al., 2020). The task becomes even more critical when distinguishing legitimate transactions from malicious ones. Machine learning (ML) models have shown promise in addressing this issue, enabling accurate classification of ransomware transactions based on their distinct features (Zhang et al., 2021; Xu, 2021; Abraham and George, 2019). In this research, we propose an analysis of Bitcoin transactions collected from the blockchain to determine the feasibility of classifying each transaction as belonging to a ransomware family. We utilize the BitcoinHeist ransomware dataset and employ machine-learning techniques for transaction classification. Various transaction features are analyzed, and the classification results' precision, accuracy, and recall are evaluated.

To build an efficient Ransomware Detection System (RDS), this article introduces a novel hybrid RDS framework that leverages multiple ML algorithms. The proposed framework comprises two traditional ML stages, namely data preprocessing and feature engineering, along with four tiers of learning models.

The first tier employs tree-based supervised learners such as decision tree (DT), random forest (RF), and extreme gradient boosting (XGBoost) as multiclass classifiers for detecting known ransomware attacks. The second tier incorporates a stacking ensemble model and employs hyperparameter optimization based on the Optuna method to optimize the performance of the supervised learners.

The third tier utilizes unsupervised learning through labeling (L) k-means algorithm to address the challenge of detecting unknown ransomware attacks. The fourth tier incorporates biased classifiers and employs hyperparameter optimization techniques to optimize the unsupervised learning models. The hybrid RDS framework comprises known and unknown ransomware detection functionalities, with the base learners in tiers 1 and 3 optimized in tiers 2 and 4 to enhance model performance. The proposed framework also includes data preprocessing and feature engineering procedures to improve the quality of the datasets, leading to more accurate attack detection. The performance evaluation of the RDS is conducted on the BitcoinHeist dataset, a publicly available network dataset. Various metrics, including accuracy, precision, recall, F1 scores, ROC score, and model execution time, are employed to assess the proposed model's feasibility, effectiveness, and efficiency. This research presents the pioneering development of a hybrid RDS that optimizes learning models to effectively identify known and previously unseen ransomware attack patterns within the BitcoinHeist dataset (Akcora et al., 2019).

The research presented in this paper offers several key contributions to ransomware detection. These contributions are as follows:

1. **Novel RDS framework:** We propose a novel Hybrid Ransomware Detection System framework capable of accurately detecting various types of surveyed ransomware attacks within the BitcoinHeist dataset. This framework combines the strengths of signature-based and anomaly-based detection approaches to improve the overall detection accuracy.
2. **Engineering new features:** We introduce new features based on Arithmetic operations and K-means algorithms. These features capture important characteristics and patterns in the BitcoinHeist dataset, enhancing the effectiveness of the ransomware detection process.
3. **Feature engineering model:** We propose a feature engineering model based on the Gradient Boosting Classifier. This model leverages the power of gradient boosting to optimize feature selection and construction, resulting in improved detection performance.
4. **Anomaly-based RDS:** We introduce an anomaly-based RDS that utilizes L-kmeans clustering and biased classifiers to detect unknown ransomware threats effectively. This enables the detection of previously unseen ransomware strains by leveraging unsupervised and supervised learning techniques.
5. **Hyperparameter optimization techniques:** We discuss various HPO techniques to automatically tune the parameters of each tier in the proposed RDS. By optimizing these parameters, we ensure the optimal performance of our model and improve its efficiency.
6. **Performance evaluation:** We evaluate the efficiency of the proposed model on the BitcoinHeist dataset for known and unknown ransomware attacks. Through comprehensive experiments and analysis, we demonstrate the effectiveness and robustness of our approach.

By addressing the challenges of ransomware detection in cryptocurrency transactions, this research advances cybersecurity measures in digital currencies, ensuring the integrity and security of Bitcoin transactions. The remainder of this article is organized as follows. Section 2 discusses the related work. Section 4 presents the BitcoinHeist dataset. In Section 3, we explain the deployment process of Ransomware Detection Systems with cryptocurrency networks. Section 4 discusses the BitcoinHeist dataset used in this work. In Section 5, all the systems' components are discussed in detail. Section 6 presents and discusses the experimental results. Section 7 concludes this article.

2. Literature review

2.1. Ransomware detection

The proliferation of ransomware attacks and the use of cryptocurrencies, such as Bitcoin, for ransom payments, have prompted researchers to explore various approaches for detecting and analyzing ransomware activities. Early studies focused on analyzing the Bitcoin

blockchain to track illegal activities and identify suspicious transactions (Androulaki et al., 2013; Di Battista et al., 2015). However, the pseudonymous nature of Bitcoin transactions makes it challenging to trace the flow of funds and identify the recipients (Möser and Böhme, 2017).

Researchers have delved into various facets of cryptocurrency ransomware networks concerning ransomware detection. Certain studies have concentrated on identifying shared hacker behavior and have employed heuristics to reveal patterns related to ransomware payments (Huang et al., 2018). These approaches rely on analyzing known ransomware transactions and applying decision rules based on the amounts and timing of those transactions (Liao et al., 2016). Other researchers have collaborated with blockchain analytics companies to study the trading behavior of different ransomware families (Xu, 2021). They have employed descriptive statistical analysis to identify differences in Bitcoin trading patterns among these families.

Machine learning techniques have also been applied to ransomware detection. For example, decision trees and ensemble learning models have been utilized to classify ransomware families based on transaction data (Al Harrack, 2021). Several researchers have proposed innovative approaches, such as NetConver, which employs decision tree models to analyze ransomware activities within network traffic (Alhawi et al., 2018). The primary emphasis of these studies lies in the early detection of ransomware before it infiltrates a system. However, these approaches do not specifically address the analysis of Bitcoin transactions, which play a significant role in cryptocurrency ransomware operations.

Almashhadani et al. (2019) extensively investigated network-based ransomware detection. They focused on crypto ransomware and used the Locky family as a case study. They created a dedicated testbed environment to analyze the network activities associated with ransomware. From TCP, HTTP, DNS, and NBNS traffic, they extracted and classified valuable network features across multiple categories. They proposed a multi-classifier network-based method that operated at both the packet and flow levels to detect ransomware. Through rigorous experimentation, they achieved remarkable detection accuracy, with 97.92% at the packet level and 97.08% at the flow level, affirming the efficacy of their feature extraction and classification techniques.

While the aforementioned initiatives provide valuable insights into ransomware detection, there is a lack of comprehensive approaches that specifically analyze Bitcoin transactions associated with ransomware. The classification of ransomware transactions within the Bitcoin network requires addressing the challenges posed by the anonymity of payments and identifying known and unknown ransomware attacks. This research aims to fill this gap by proposing a novel hybrid Ransomware Detection System RDS that leverages machine learning algorithms to accurately detect and classify ransomware transactions within the BitcoinHeist dataset (Akcora et al., 2019).

2.2. Bitcoin network security and threats

Apart from ransomware detection, several researchers have investigated the security and privacy concerns of the Bitcoin network. Conti et al. conducted a comprehensive survey on the security challenges and attacks in the Bitcoin network, covering various types of attacks such as double spending, Finney attacks, brute force attacks, and more (Conti et al., 2018). They also discussed countermeasures and mitigation strategies for these attacks. Lim et al. focused on security threats to Bitcoin, including distributed denial-of-service (DDoS) attacks on exchanges, Bitcoin mining malware, and extortion (Lim et al., 2014). They highlighted the vulnerabilities in the Bitcoin ecosystem and potential attack vectors.

Furthermore, researchers have examined the impact of DDoS attacks on Bitcoin exchanges. Feder et al. studied the effects of DDoS attacks on the now-defunct Mt. Gox exchange. They observed a significant decrease in trading volume following such attacks, particularly in large-volume trades (Feder et al., 2018). The study highlights the potential disruption caused by DDoS attacks on cryptocurrency exchanges.

The risks associated with Bitcoin exchanges have also been analyzed by Moore et al. who investigated the factors contributing to the closure of Bitcoin exchanges due to fraud attempts and security breaches (Moore et al., 2018). They explored the relationship between security-related features, such as multiple-factor authentication and bug bounties, and the closure of exchanges. The analysis provides valuable insights into the security challenges faced by cryptocurrency exchanges.

While these studies offer valuable insights into Bitcoin network security and threats, they are distinct from the specific focus of ransomware detection in Bitcoin transactions. The proposed RDS framework addresses the unique challenges ransomware attacks pose within the Bitcoin network. It significantly enhances the comprehension and mitigation of cryptocurrency-associated cybersecurity risks, advancing the field's knowledge and countermeasures.

2.3. Literature comparison

In this subsection, we compare our proposed Hybrid RDS with other recent ransomware detection techniques for the BitcoinHeist dataset (Akcora et al., 2019), as summarized in Table 1. The comparison is based on various criteria, including known ransomware detection, zero-day ransomware detection, data preprocessing and sampling, feature engineering, model optimization, and code reproducibility. Several works in the literature have focused on known ransomware detection, including Alsaif et al. (2023), Al Harrack (2021), jihwankimqd (2019), Cahyani et al. (2021), and Al-Haija and Alsulami (2021). While these works have contributed to detecting known ransomware strains, our proposed Hybrid RDS surpasses them by effectively detecting known ransomware and demonstrating superior performance in detecting zero-day ransomware strains. The ability to detect unknown attacks is a significant advantage of our proposed approach.

Moreover, our Hybrid RDS excels in data preprocessing, sampling, and feature engineering. We have employed advanced techniques to preprocess and sample the data, ensuring high-quality input for our models. Additionally, we have performed comprehensive feature engineering, extracting relevant features and engineering new ones that capture the distinctive characteristics of ransomware attacks in bitcoin networks.

Model optimization is another crucial aspect where our proposed approach demonstrates superiority. Leveraging advanced techniques such as stacking, ensemble learning, and hyperparameter optimization, we have fine-tuned the parameters of our models, resulting in improved performance and accuracy. This optimization process ensures our models are well-suited for ransomware detection in Bitcoin networks.

Lastly, our work excels in terms of code reproducibility. We have made the source code of our project openly available on GitHub (https://github.com/odib/Bitcoin_Heist_Classification), providing researchers and practitioners with access to the code for verification and further development. This transparency promotes the reproducibility of our results and encourages advancements in the field.

3. Bitcoin networks, and RDS deployment

The rise of ransomware attacks in the Bitcoin network has created a need for robust detection systems to secure Bitcoin transactions. A Ransomware Detection System (RDS) can be designed to identify suspicious patterns that indicate ransomware activity. The architecture of such a system is depicted in Fig. 1, and involves several stages:

1. Data Collection: The system collects data from the Bitcoin blockchain, including transaction details such as sender and receiver addresses, amount transferred, and timestamp. This data provides the necessary information for analyzing and detecting ransomware transactions.

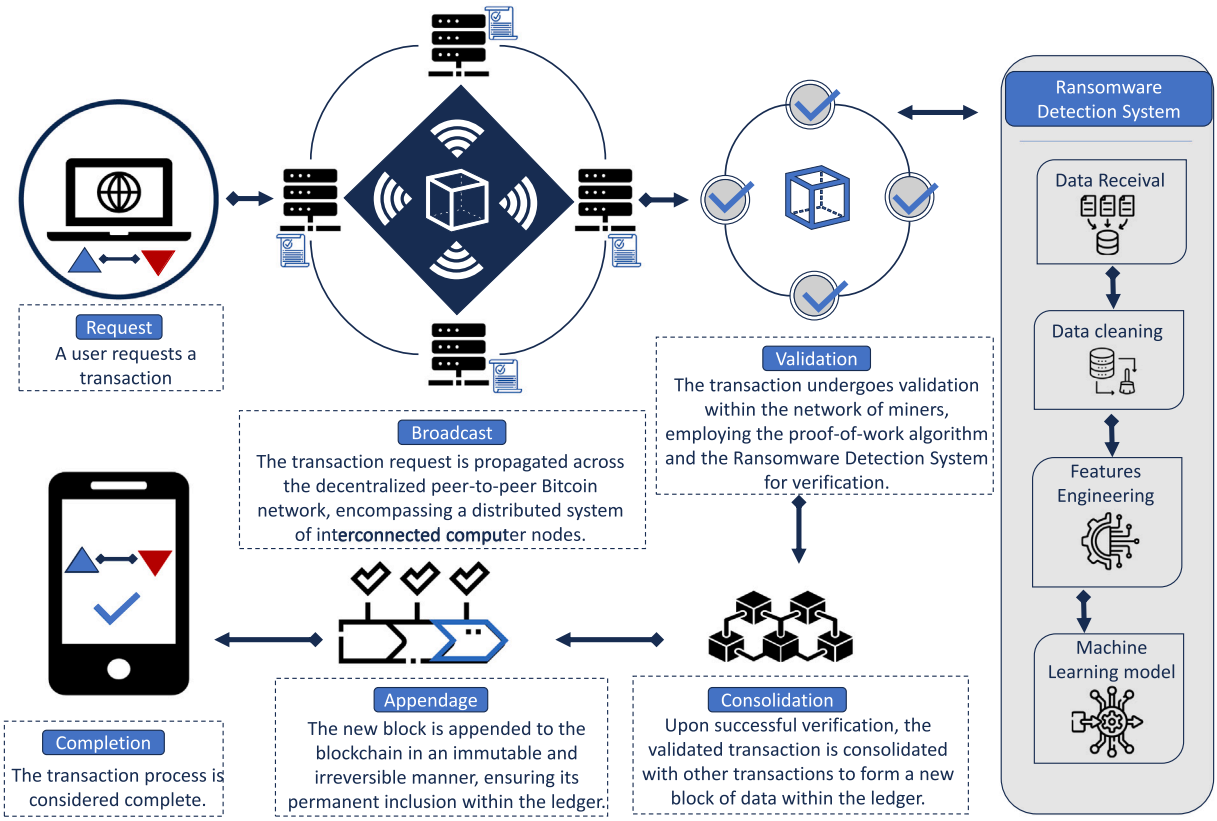


Fig. 1. Architecture of machine learning-based ransomware classification of Bitcoin transactions.

Table 1
Evaluating ransomware detection systems for bitcoin transactions: A comparative study.

Paper	Known ransomware detection	Zero-day ransomware detection	Data preprocessing and sampling	Feature engineering	Model optimization	Code reproducability
Alsaif et al. (2023)	✓		✓			
Al Harrack (2021)	✓		✓			
jihwankimqd (2019)	✓		✓	✓		
Cahyani et al. (2021)	✓		✓		✓	
Al-Haija and Alsulami (2021)	✓		✓		✓	
Proposed RDS	✓	✓	✓	✓	✓	✓

2. Feature Extraction: The collected data is processed to extract relevant features for identifying ransomware transactions. These features could include the frequency of transactions, the amount of Bitcoin transferred, and the number of unique transaction addresses. Extracting meaningful features helps in training the machine learning model effectively.
3. Machine Learning Model Training: The extracted features train a machine learning model, such as a shallow neural network or an optimizable decision tree. The model learns patterns and correlations between features and ransomware activity using labeled data, where known ransomware transactions are identified.
4. Classification of New Transactions: Once the model is trained, it can classify new Bitcoin transaction data based on their likelihood of being involved in ransomware transactions. The trained model can assign a probability or score, indicating the level of suspicion for potential ransomware activity.
5. Transaction Analysis and Alert Generation: To ensure the security of the Bitcoin network, all transactions are analyzed by the RDS before being confirmed in the blockchain. If the system detects a potential ransomware attack based on the classification results, it triggers alerts to notify relevant parties. Depending on the system's capabilities, it may also have the ability to block suspicious transactions from being added to the blockchain.

By implementing such a proactive approach, the RDS helps prevent the Bitcoin network from being exploited by ransomware attacks. It adds a layer of security by identifying and stopping potentially harmful transactions before they can cause significant damage.

4. Bitcoin transaction dataset

The dataset utilized in this study was provided by Akcora et al. (2019) and is currently accessible through the UCI Machine Learning Repository, hosted by the University of California at Irvine (Goldsmith et al., 2020). Each row in the dataset corresponds to a Bitcoin blockchain transaction and comprises the following attributes:

- **Address**: A string representing the transaction's address.
- **Year**: An integer indicating the year of the transaction.
- **Day**: An integer indicating the day of the transaction.
- **Length**: The number of non-starter transactions on the longest chain associated with the address.
- **Count**: The number of transaction starters linked to the address.
- **Neighbors**: The number of transactions with this address as an output.
- **Weight**: The sum of the fraction of Bitcoin coins originating from a starter-start transaction and reaching this address.

Table 2
Bitcoin transactions dataset statistics.

Ransomware family	Ransomware type	# of instances
Montreal (23)	APT	11
	ComradeCircle	1
	CryptConsole	7
	CryptoLocker	9315
	CryptoTorLocker	55
	CryptXXX	2419
	DMALocker	251
	DMALockerv3	354
	EDA2	6
	Flyper	9
	Globe	32
	GlobeImposter	55
	Globev3	34
	JigSaw	4
	NoobCrypt	483
	Razy	13
	Sam	1
	SamSam	62
	VenusLocker	7
	WannaCry	28
	XLocker	1
	XLockerv5.0	7
	XTPLocker	8
	Total	13,163
Padua (3)	CryptoWall	12,390
	Jigsaw	2
	KeRanger	10
	Total	12,402
Princeton(2)	Cerber	9223
	Locky	6625
	Total	15,848
Total no. of ransomware	28	41,413

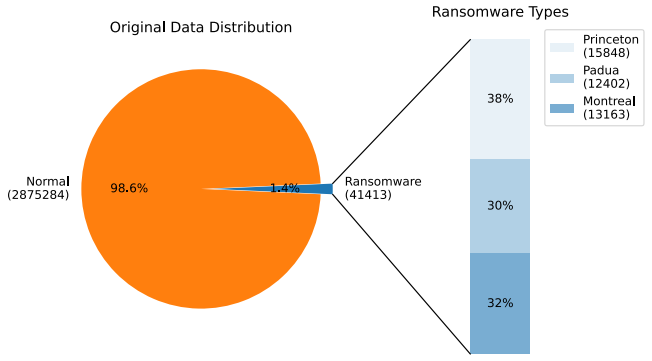


Fig. 2. Distribution of the Bitcoin transactions dataset.

- **Looped**: The number of starter transactions connected to this address by multiple direct paths.
- **Income**: An integer representing the amount of Satoshi. (1 Bitcoin = 10^8 Satoshi. Satoshi is the smallest unit of Bitcoin.)
- **Label**: A string indicating the nature of the transaction, with two categories: "Ransomware" (comprising 29 ransomware families) and "White" (representing legitimate transactions).

The dataset specifications are provided in Table 2. The record distribution between the ransomware families seems almost balanced, with 13,163 records for the first, 12,402 for the second, and 15,848 for the third family (see Fig. 2 to comprehend the distribution of the dataset).

According to Table 2, the transactions within the dataset are categorized into 28 distinct families, which are distributed as follows:

- 3 categories, namely Princeton, Montreal, and Padua, are associated with ransomware. Collectively, these categories encompass 28 families.
- 1 white category representing legitimate transactions.

5. Proposed framework

To develop a predictive model for ransomware detection and classification of Bitcoin transactions, utilizing supervised machine learning algorithms (MLAs) appears essential (Uddin et al., 2019). We aim in this work to design a self-contained classification scheme based on ML, which involves several stages, including data preprocessing, data balancing, instance classification, and model evaluation. The proposed Framework for the incumbent problem is elaborated below.

5.1. System architecture

This study aims to develop an efficient Ransomware Detection System (RDS) capable of combating various common ransomware threats discussed in Section 4. This article proposes a novel hybrid Ransomware Detection System to detect known and unknown ransomware on the Bitcoin network. Fig. 3 illustrates the architecture of the proposed system, which consists of several layers: data preprocessing, a signature-based RDS, and an anomaly-based RDS.

In the data preprocessing layer, the Bitcoin transaction dataset is collected for evaluating the system's performance on different types of ransomware. The data preprocessing involves various procedures, including Label Encoding, Feature Engineering, Outlier Removal, Normalization, Scaling, Class Imbalance Handling, Data Splitting, and Feature Selection for model training. These preprocessing and feature engineering steps significantly enhance the quality of Bitcoin data, leading to more accurate model learning.

Next, the signature-based RDS is developed to detect known ransomware by training four ML algorithms: Decision Trees (DT), Random Forests (RF), XGBoost, and Multilayer Perceptron (MLP). These four algorithms are then utilized to construct a stacking ensemble model. The Hyperparameter Optimization (HPO) method, based on Optuna Framework, is employed to improve ransomware detection accuracy by combining the outputs of the base learners and optimizing their performance.

Subsequently, an anomaly-based RDS is constructed to identify unknown attacks. This approach passes suspicious instances to a clustering-based k-means model to effectively separate ransomware samples from normal samples. The last step of the RDS involves employing the HPO-Optuna method and biased classifiers to optimize the model and reduce detection errors in the k-means. Ultimately, the detection result of each test sample is returned, indicating whether it is a known ransomware with its corresponding type, an unknown ransomware, or a normal transaction. To offer an overview of the algorithmic choices in the proposed Ransomware Detection System (RDS), Table 3 presents a concise description of each component and its performance impact. Detailed explanations of these algorithms can be found in the subsequent subsections.

5.2. Data analysis

To comprehensively understand the current dataset, we performed various statistical analyses. These included examining mean, standard deviation, variance inflation factor (VIF), outliers, attribute scales, and correlation matrix among attributes. Skewness was also studied in detail. VIF was calculated to assess the multicollinearity of each attribute using the following formulas:

$$X_1 = \alpha_0 + \alpha_2 X_2 + \alpha_3 X_3 + \cdots + \alpha_k X_k + e \tag{1}$$

$$VIF_i = \frac{1}{1 - R_i^2} \tag{2}$$

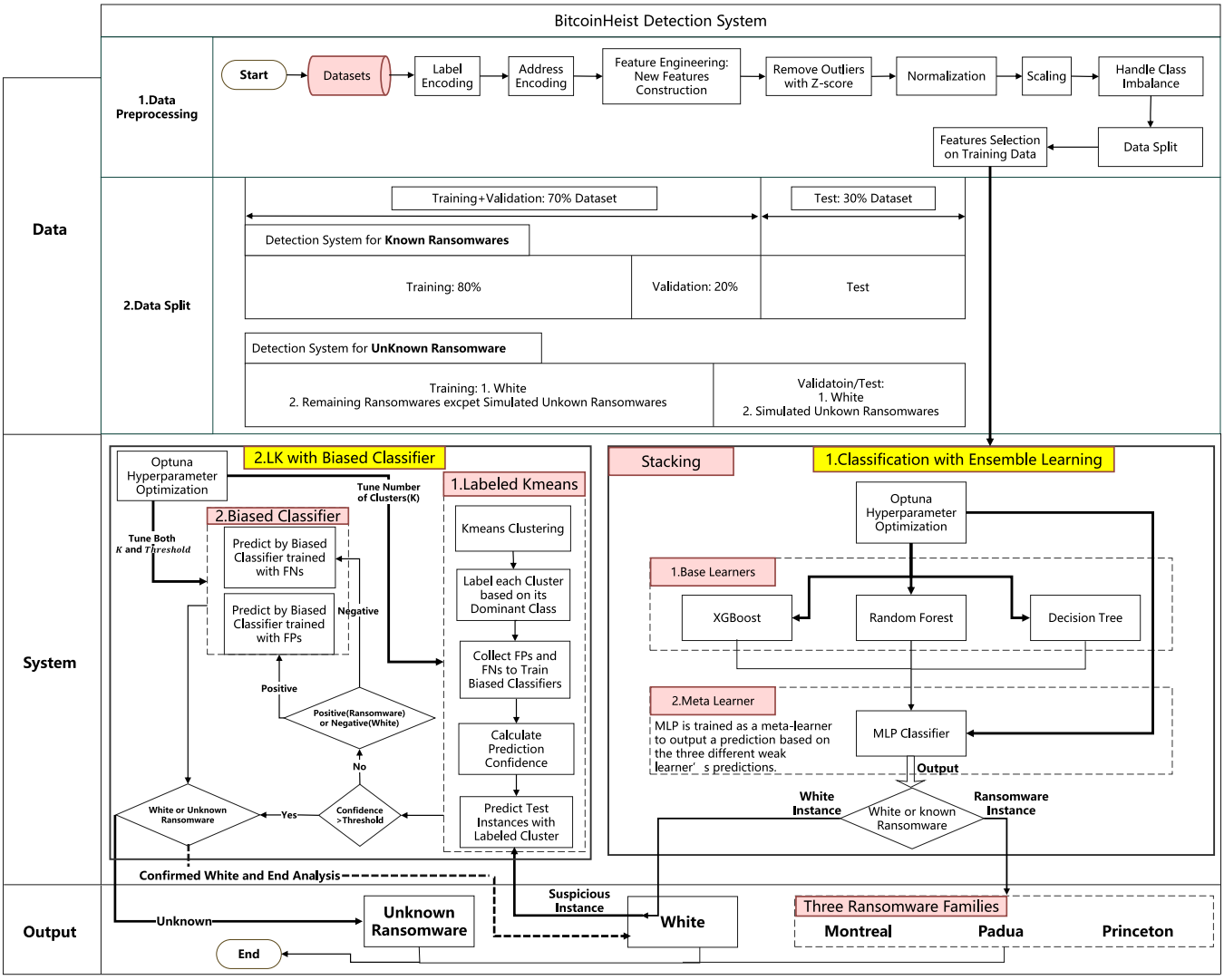


Fig. 3. The hybrid ransomware detection system framework.

In these equations, X_1 represents the target variable, while other variables are denoted by X_i. The constant α_0 and the error term e are also included in the model. This regression model allows for calculating the coefficient of determination, R^2, which is subsequently used in VIF calculations. Results indicate that the original dataset does not exhibit multicollinearity, but is significantly impacted by skewness, outliers, and imbalanced data. Consequently, data preprocessing, as detailed in the following sections, is necessary to ensure the robustness and validity of the final research results. Further discussion on these findings can be found in Section 6.

5.3. Label encoding

To accommodate the requirement of machine learning models for numeric input and output variables, categorical data needs to be encoded before the models can be fitted and evaluated. In this study, the label encoder method, specifically applied to the "Labels" attribute, is utilized to convert categorical data into numerical representations (Pedregosa et al., 2011). This approach enables the model to understand better and generalize the input data, improving its performance. To address potential limitations of machine learning models in capturing relationships between categorical variables, encoding techniques such as label encoding, one-hot encoding, and dummy encoding can be employed (Chowdhury et al., 2021). Our study focused on transforming the "Address" and "Labels" attributes into numerical representations. Specifically, the label encoder method encoded the "Labels" attribute, resulting in the mappings shown in Table 4.

5.4. New attributes construction

Feature construction is a well-established approach in ransomware detection models, known for its significant performance improvements (Davis and Clark, 2011). Building upon this approach, we have developed new attributes outlined in Table 5.

Drawing inspiration from topological data analysis (Akcora et al., 2019) and the work of Chandrasekharuni (2021), we have incorporated five overarching attribute categories into our framework: Clustering-based features, Temporal attributes, Cybercrime related attributes, Transaction volume attributes, and Transaction frequency attributes. Specifically, we have achieved optimal performance for the Clustering attribute by fine-tuning the K-means algorithm based on the sum of squared distances (SSD). Recognizing that not all constructed attributes may be relevant to the model, we will employ feature selection techniques to refine the attribute set. Utilizing a Gradient Boosting

Table 3
Rationale and performance impact of each component of the RDS.

Stage	Components	Algorithms	Rationale
Data pre-processing	Label encoding	Label encoder	Convert categorical labels to numerical form for machine learning algorithms. This conversion allows the algorithms to process the data effectively.
	Address encoding	Label encoder	Encode Bitcoin addresses as numerical values to enable analysis and modeling. This facilitates the inclusion of addresses as features in the ML models.
	New features construction	Arithmetic operation	Perform mathematical operations on features to generate new meaningful ones. This provides additional insights and patterns for improved performance.
		Kmeans elbow	This identifies the optimal number of clusters for feature construction. It helps create informative features based on the clustering structure of the data.
	Remove outliers	Z-score	Normalize features and handle outliers. This ensures a similar scale and reduces the impact of outliers on model training.
	Normalization	Power transformer	Transform feature distribution to a Gaussian-like distribution. This improves the model's ability to capture complex data patterns.
	Scaling	MinMax scaler	Scale features to a specific range using MinMaxScaler. This can prevent bias in the models and improve performance and generalizability.
	Handle class imbalance	SMOTE	Address class imbalance by generating synthetic samples. This balances the representations, improving the detection of rare classes.
		Random underSampler	Balance class distribution by undersampling the majority class. This reduces the dominance of the majority classes and ensures balanced training.
	Data split	Train & test split	Split the dataset into training and testing for model evaluation. This enables performance assessment on unseen data and helps avoid overfitting.
	Features selection	GBoost classifier	Select relevant features using the GBoost, which measures feature importance. This selection identifies the most informative features for improved performance.
		HPO	Optimize hyperparameters of the feature selection algorithm. This can improve feature selection and enhance the model's performance.
Signature-based RDS	Multiclass classification with ensemble learning	Decision tree	Utilize decision trees for known ransomware detection. This is effective for tabular data analysis and can capture complex data patterns.
		Random forest	Ensembling helps in capturing diverse patterns and reducing overfitting. This improves classification accuracy and handles complex patterns.
		XGBoost	XGBoost shows superior performance on complex tabular data. This handles complex interactions among features, improving model accuracy.
		MLP classifier	Apply MLP for nonlinear mapping of features. This captures nonlinear relationships, enabling effective ransomware detection.
		Stacking	Combine base classifiers using stacking ensemble for improved performance. This helps obtain a more accurate and robust classification model.
		HPO	Optimize hyperparameters of the base classifiers. This can improve the performance and utilization of the ensemble models.
Anomaly-based RDS	LK with biased classifier	Kmeans	Use K-means for unknown attack detection based on the similarity of transactions. This can identify clusters of similar bitcoin transactions, helping identify abnormal transactions associated with unknown attacks.
		Labeling Kmeans	Apply CL-k-means to generate normal and attack clusters for identifying zero-day attacks. CL-k-means can help identify zero-day attacks by creating separate clusters for normal and attack transactions.
		Biased classifiers	Train biased classifiers on false positives (FPs) and false negatives (FNs) of CL-k-means to reduce errors in complex unknown attack detection. Biased classifiers can focus on correcting the specific errors made by CL-k-means, thereby improving the accuracy of unknown attack detection.
		HPO	Optimize hyperparameters of the unknown attack detection components using Optuna. Fine-tuning the hyperparameters can improve the performance and effectiveness of the unknown RDS.

Table 4
Labels and numerical representation in the dataset.

Labels	Numerical representation
Montreal	0
Padua	1
Princeton	2
White	3

Classifier, we will identify and retain only the most relevant attributes, enhancing the model's performance.

5.5. *Outliers removal*

Anomaly detection is crucial for identifying data points that deviate from expected patterns or exhibit atypical behavior within the observed probability distribution. However, the challenge lies in the absence of prior knowledge regarding the underlying dataset distribution, necessitating algorithms capable of learning appropriate metrics for anomaly detection. Our research used the Z-score method to identify and eliminate outliers from our dataset. The Z-score method calculates the deviation of each data point from the mean by evaluating its standard deviation. Data points with Z-scores beyond a certain threshold are considered outliers and subsequently removed from the dataset. By employing this approach, we effectively mitigate the influence of outliers on our model, resulting in improved performance and enhanced result accuracy.

The Z-score method is widely employed during the preprocessing phase of machine learning projects to enhance model performance (Pedregosa et al., 2011). Initially, we applied the Z-score method to detect outliers for each attribute. Specifically, we identified 178 instances as

Table 5
Attributes, type, nature, and description of the newly added features.

Attributes	Type	Attribute nature	Description
Number of address Tx	int	Tx frequency	The number of Txs occurring for each address
Quarter number	int	Temporal	The quarter of a year in which the Tx occurs
Is close to holiday	Bool	Temporal	Whether the Tx occurs near holidays (America)
Day of week	int	Temporal	The day of the week on which the Tx occurs (Monday to Sunday)
Average income per Tx	float	Tx volume	The average amount of Satoshi for each address
Tx count	int	Tx frequency	The average count (original feature) for each address
Looped ratio	float	Cybercrime	The ratio of being looped for each address
Merge behavior	float	Cybercrime	The average amount of Bitcoin merged per Tx
Cybercrime-related	int	Cybercrime	Whether the address is related to cybercrime
Length-weight	float	Cybercrime	The relationship between the complexity of a Tx chain and merge behavior
Count looped	int	Cybercrime	The interactions between the number of transactions and the degree to which those transactions are looped
Month	int	Temporal	The month of the transaction
Week	int	Temporal	The week of the transaction
Total volume	float	Tx volume	The total income of transactions in each address
Average volume	float	Tx volume	The average income of transactions in a given address
Tx frequency	int	Tx frequency	The frequency of transactions for each address
Clustering	int	Clustering	The data kinds based on its transaction behavior

outliers within the *income* attribute. This process was repeated for all attributes, leading to the elimination of outliers across the dataset. As a result, our dataset was refined, and we retained 187,079 instances for subsequent stages of our research. By removing outliers, we enhance the integrity and reliability of our dataset, enabling more robust and accurate modeling in further stages of our investigation.

5.6. Power transformation

To address the issues of skewness and normality in the dataset, we have applied the Yeo-Johnson power transformation, which is known to handle various types of data distributions effectively (Weisberg, 2001). Unlike the Box–Cox transformation (Atkinson et al., 2021), which can only be applied to strictly positive numbers, the Yeo-Johnson transformation accommodates positive and negative values. The transformation is defined as follows:

$$\psi(\lambda, y) = \begin{cases} \frac{(y+1)^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0, y \geq 0 \\ \log(y+1) & \text{if } \lambda = 0, y \geq 0 \\ -\left(\frac{(-y+1)^{2-\lambda}-1}{2-\lambda}\right) & \text{if } \lambda \neq 2, y < 0 \\ -\log(-y+1) & \text{if } \lambda = 2, y < 0 \end{cases} \tag{3}$$

When the input values, y, are positive, the Yeo-Johnson transformation is similar to the Box–Cox transformation with $(y+1)$. For negative input values, the Yeo-Johnson transformation is equivalent to the Box–Cox transformation applied to $(-y+1)$ with an exponent of $(2-\lambda)$. By applying the Yeo-Johnson power transformation, we effectively address the skewness issues in the dataset and improve its normality. This preprocessing step plays a crucial role in enhancing modeling performance and facilitates the application of machine learning techniques that assume normality.

5.7. Data standardization

Scale inconsistency across different attributes can indeed affect the performance of classification models. In many research studies, like those by Sinsomboonthong (2022), Min-Max normalization has effectively addressed this issue. The Min-Max normalization formula is straightforward, making it well-suited for handling the scale inconsistency in the Bitcoin Heist dataset. By applying Min-Max normalization, each attribute's values are transformed to fall within 0 to 1. The Min-Max normalization formula is provided below:

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{4}$$

where x_{norm} represents the normalized value of x, and $\min(x)$ and $\max(x)$ denote the minimum and maximum values of x, respectively.

5.8. Data balancing

Several research studies have shown that a balanced dataset enhances the classification performance of various classifiers compared to an imbalanced dataset. Two sampling methods were employed to address the imbalance in the dataset: Synthetic Minority Over-sampling Technique (SMOTE) and random undersampling (Sahni et al., 2021).

SMOTE is a widely used oversampling technique that generates synthetic instances for the minority class. It interpolates between minority class instances and their nearest neighbors in the feature space. The SMOTE algorithm can be described as follows:

1. For each minority instance x_i:
   (a) Randomly select k nearest neighbors x_{nn}.
   (b) For each selected neighbor x_{nn}:
      i. Compute the difference vector $\Delta = x_{\text{nn}} - x_i$.
      ii. Generate a synthetic instance x_{syn} using the formula: $x_{\text{syn}} = x_i + \alpha \cdot \Delta$, where α is a random value between 0 and 1.
      iii. Add the synthetic instance x_{syn} to the dataset.

Random undersampling, on the other hand, involves removing data from the majority class to achieve a balanced distribution. The random undersampling process can be summarized as follows:

1. Randomly select a subset of instances from the majority class.
2. Remove the selected instances from the dataset.

In this research, SMOTE was applied to the minority class (ransomware) to generate synthetic instances and increase the representation of the minority class in the dataset. Random undersampling was performed on the majority class (normal) by randomly selecting a subset of instances to achieve a balanced dataset.

5.9. Feature selection

Feature selection is an essential step in the RDS Framework, where we employ a Gradient Boosting Classifier to identify the most relevant features during the data preprocessing stage. This ensemble-based machine learning model is trained on the training data to extract feature importance scores, quantifying the relevance of each feature for making accurate predictions.

The importance score of a feature is calculated based on its contribution to reducing the loss function (e.g., cross-entropy or Gini impurity) during the boosting process.

Mathematically, the feature importance score in a Gradient Boosting Classifier can be computed as follows:

$$\text{Importance}(f) = \frac{\sum_{t=1}^{T} \text{Gain}(f,t) \cdot \text{Weight}(t)}{\sum_{t=1}^{T} \text{Weight}(t)} \quad (5)$$

where f represents a specific feature, T is the total number of trees in the ensemble, $\text{Gain}(f,t)$ denotes the improvement in the loss function achieved by splitting on feature f at tree t, and $\text{Weight}(t)$ represents the weight of tree t in the ensemble.

We leverage Optuna, an open-source hyperparameter optimization framework, to optimize the feature selection process. We define an objective function to minimize and set a threshold for feature importance. Only features with importance scores above this threshold are selected. Optuna helps us find the optimal threshold value, maximizing the model's accuracy.

After completing the optimization process, we create a mask using the best threshold value found by Optuna. This mask is applied to the training and test datasets, resulting in new datasets comprising only the most relevant features. The detailed results of the data preprocessing can be found in Section 6, showcasing the preprocessing steps and the outcome after applying the optimal feature selection threshold.

By refining the feature set, we anticipate improved performance and accuracy in the RDS model. Selecting the most informative features allows the model to focus on the most discriminative information, enhancing its ability to detect and classify ransomware transactions effectively.

5.10. *Proposed hybrid RDS*

Ransomware Detection Systems (RDSs) can be classified into two main categories: signature-based RDSs and anomaly-based RDSs. Signature-based RDSs rely on supervised machine learning models trained on labeled datasets to detect known ransomware patterns. However, they cannot identify new ransomware patterns not previously stored in their databases (Garg and Maheshwari, 2016).

On the other hand, anomaly-based RDSs leverage unsupervised learning algorithms to differentiate between unknown ransomware data and normal data. These systems assume that new ransomware data are statistically more similar to known ransomware data than normal data. However, they often produce a high number of false alarms.

In this article, we propose a hybrid RDS framework to overcome the limitations of both signature-based and anomaly-based RDSs. The hybrid RDS combines the strengths of both approaches to detect known and zero-day ransomware attacks effectively. By integrating the signature-based and anomaly-based RDS, we aim to achieve improved accuracy and detection performance in identifying various types of ransomware attacks.

5.10.1. *Signature-based RDS*

The signature-based Ransomware Detection System (RDS) is developed by training an ensemble learning model on the preprocessed and engineered labeled data sets. Three tree-based machine learning algorithms—Decision Trees (DT), Random Forest (RF), and XGBoost—are chosen as the base learners.

The DT algorithm, known for its simplicity, interpretability, and ability to handle diverse data types, employs a tree structure to fit data and make predictions. To achieve optimal performance, DT algorithms require careful tuning of hyperparameters, including tree depth, minimum sample split, minimum sample leaf, maximum sample nodes, and minimum weight fraction leaf. In contrast, as an ensemble learning approach, the Random Forest (RF) model combines multiple DT classifiers through majority voting, enhancing predictive accuracy while mitigating overfitting and variance issues. This combination of interpretability and predictive power makes DT and RF valuable tools in various scientific applications and real-world scenarios. At the same time, XGBoost is a gradient-boosted DT-based algorithm that sequentially builds decision trees, learning from the errors of its predecessors. These algorithms are selected for their ability to handle complex, nonlinear data, parallel execution capabilities, feature importance calculation, and the inherent randomness in their construction process, contributing to a robust ensemble model with improved generalizability.

After obtaining the three tree-based ML models, stacking, an ensemble learning method, is employed to enhance model performance further. Stacking combines the predictions from the DT, RF, and XGBoost models as input features for training a Multilayer Perceptron (MLP) classifier, serving as the meta-model for generating the final prediction. MLP is advantageous in capturing complex relationships and patterns that base models might miss, thereby improving the accuracy and reliability of the prediction. It is noteworthy that in this context of ensemble learning, further exploration can involve advanced ensembling techniques, particularly those based on adaptive feature weighting. These techniques encompass ensembling approaches relying on features with attributes like low correlation (Leng and Zhang, 2013), high discrimination (Leng et al., 2017), and high accuracy (Leng et al., 2013).

To optimize the performance of the tree-based ML models, their important hyperparameters are tuned using Optuna, an open-source hyperparameter optimization framework. Optuna automates the search for optimal hyperparameters by efficiently exploring the hyperparameter search space, utilizing statistical modeling and pruning techniques. It dynamically adjusts the search strategy based on a user-defined objective function and search space, maximizing the model's performance. Optuna supports various types of hyperparameters and seamlessly integrates with popular machine-learning libraries, simplifying the hyperparameter optimization process and enhancing model performance.

The runtime complexity of tree-based machine learning algorithms used in signature-based RDS can be summarized as follows. The average time complexity for DT classifiers is approximately $O(n \log n)$, considering their decision-making structure.RF classifiers, which combine multiple DT classifiers, have a time complexity of approximately $O(n^2\sqrt{ft})$, where n is the number of instances, f is the number of features, and t represents the number of DTs in the ensemble model. The gradient-boosted DT-based algorithm XGBoost has a time complexity of $O(nft)$, considering the number of instances (n), features (f), and the number of boosting rounds (t).

Considering the number of instances, features, and the number of DTs in the ensemble models, the overall time complexity of the signature-based RDS is primarily determined by the most computationally demanding component. As a result, the runtime complexity of the first stage can be approximated as follows: $O(n^2 f)$ for DT, $O(n^2\sqrt{ft})$ for RF, and $O(nft)$ for XGBoost.

It is important to note that the runtime complexity of the MLP classifier, which is used as the meta-model in the ensemble, depends on factors such as the input data size, the number of layers, and the neurons per layer. For a single-layer MLP with m neurons, its time complexity can be approximated as $O(nmd)$.

5.10.2. *Anomaly-based RDS*

The proposed signature-based RDS has demonstrated efficacy in detecting many known ransomware variants. Despite this, the system encounters limitations when presented with novel ransomware types that are not yet encompassed within established patterns of known ransomware. Such new types could potentially be misclassified as benign or "White". This classification issue warrants particular attention as these "White" labeled instances may potentially represent ransomware samples yet to be encountered during the system's training phase.

To address this limitation, incorporating a novel anomaly-based RDS architecture is introduced. This advanced system leverages the strengths of both supervised and unsupervised learning paradigms. It is specifically designed to process these suspicious "White" instances, enhancing the system's ability to identify novel ransomware strains accurately.

The proposed RDS leverages anomaly-based algorithms to identify undisclosed ransomware from legitimate or White transactions. The novel RDS was trained and tested on an equally sized dataset comprising both 'White' transactions and two families of known ransomware to validate the system's performance. Simulated data from the unidentified ransomware was then incorporated for testing purposes. The anomaly-based RDS operates in several distinct phases:

1. **Unsupervised Phase:** In this phase, the system employs the K-means algorithm to cluster the training data into K groups based on the characteristics of the feature space. The primary assumption is that legitimate data will form densely populated clusters, while anomalies poorly integrate into any cluster or create sparse clusters. This clustering process facilitates an internal voting mechanism within each cluster, enabling classification determination in the subsequent phase.
   The choice of K-means for distinguishing between attack and normal data is motivated primarily by the real-time requirements of blockchain systems. K-means exhibits computational efficiency, making it faster than many other clustering algorithms, with a linear time complexity of $O(nkt)$, where n is the data size, k is the number of clusters, and t is the number of iterations. Mini-batch K-means are utilized to reduce model training time further, employing randomly sampled subsets as mini-batches during each training iteration. Moreover, K-means guarantees convergence and readily adapts to new samples, making it an advantageous choice for signature-based RDS implementation in the context of cryptocurrency applications.
2. **Labeling Phase:** The Labeling Phase is initiated following the clustering stage. Here, clusters are labeled based on the known classification of most constituent members derived from the training data. Clusters populated primarily by normal instances are labeled as 'normal,' and those with most anomalies are marked as 'anomalous.' Labeling is based on the majority class for clusters with a heterogeneous composition or classified as 'anomalous' if they contain a substantial portion of anomalies. This helps capture areas in the feature space frequently associated with anomalies.
3. **Supervised Phase:** This phase is triggered when a new instance is received. The instance is assigned to the nearest cluster and labeled based on the cluster's assigned label. This stage resembles supervised learning, as the incoming instance is classified in line with the labels attributed to the clusters.
4. **Biased Classifiers Training data and Prediction Confidence Collection Phase:** The subsequent stage, Biased Classifiers Training, and Prediction Confidence Collection Phase, are designed to augment the Labeling Kmeans algorithm procedure by addressing potential false positives and negatives. This is achieved by preparing training data for biased classifiers that aim to minimize such errors. The training dataset produced in the Labeling Phase is used for this purpose. The following terms, central to this project, are defined as:

   - **True Positives (TP):** The model correctly identified instances as positive (the label was 1, indicating ransomware, and predicted as 1).
   - **True Negatives (TN):** The model correctly identified instances as negative (the label was 0, indicating 'White,' and predicted as 0).
   - **False Positives (FP):** Instances incorrectly identified as positive by the model (the label was 0, indicating 'White,' but predicted as 1, indicating ransomware).
   - **False Negatives (FN):** Instances incorrectly identified as negative by the model (the label was 1, indicating ransomware, but predicted as 0, indicating 'White').

   The Labeling Phase also generates prediction confidence, defined as the ratio of the majority class size to the total size of both majority and minority classes within a cluster. This value denotes the confidence with which an instance is classified as belonging to the predominant category in a given cluster. Hence, each cluster has its unique confidence score. For instance, in classification, the Kmeans algorithm initially predicts the cluster to which an incoming instance belongs. Subsequently, this instance is labeled according to the dominant class category within the cluster, along with the cluster's prediction confidence. The inaccurately predicted instances are used as training data for biased classifiers, which aim to reduce false positives and negatives.
5. **Biased Classifiers Evaluation Phase:** This Phase introduces two biased classifiers implemented using XGBClassifier to mitigate false positives and negatives. Suppose B1 reduces false negatives and B2 diminishes false positives. Then, their respective training data are as follows:

   - **B1:** All false negative instances and an equal number of randomly sampled legitimate data from true negatives.
   - **B2:** All false positive instances and an equal number of randomly sampled ransomware data from true positives.

   After training, the biased classifiers are selectively applied based on a confidence threshold optimized by Optuna. If the Kmeans prediction confidence of a test instance surpasses this threshold, indicating high confidence in accurate prediction, the Labeling Kmeans' prediction results become the final output. In contrast, if the prediction confidence is lower, indicating strong uncertainty, the biased classifier is invoked for further analysis. Specifically, if the Labeling Kmeans' prediction is positive, false negatives are likely. Consequently, the B1 classifier is triggered to reassess these instances and replace the prediction results from Labeling Kmeans, ultimately forming the final prediction. The same procedure applies to false positives with classifier B2.
6. **Optuna Tuning Phase:** The Optuna Tuning Phase incorporates parameter tuning. For Labeling Kmeans, the parameter K, representing the number of clusters, is fine-tuned. K and the confidence threshold are optimized for the combined Labeling Kmeans with Biased Classifiers. The resulting tuned versions outperform their untuned counterparts, as confirmed by experimental results.

Lastly, the primary objective of the signature-based RDS is to perform classification tasks across three distinct ransomware families: Montreal, Padua, and Princeton, in addition to one category for non-malicious transactions, termed "White". This totals four classifications. The transactions categorized as "White" by the signature-based RDS are then subjected to further screening through the anomaly-based RDS. This additional detection layer mitigates potential risks associated with undetected, novel, or otherwise unknown ransomware strains, enhancing the system's overall efficacy. Still, distinguishing certain "unseen" ransomware patterns from normal patterns can be challenging, as they may exhibit similarities. Furthermore, legitimate network events could generate samples with patterns that differ significantly from existing normal patterns, leading to potential misclassification as ransomware samples. This highlights the importance of robust and adaptive machine-learning techniques to handle such complexities and improve the overall accuracy of ransomware detection models.

5.10.3. Two established anomaly detection unsupervised learning algorithms

In anomaly detection, two methodologies have gained significant traction due to their effectiveness: Isolation Forest and Autoencoders (Shu et al., 2022). These techniques will be benchmarks against which the proposed LK_BC algorithms will be evaluated.

Isolation Forest represents an unconventional yet potent approach to anomaly detection. Instead of defining "normal" behavior and identifying deviations from this norm, it isolates anomalies. This technique is predicated on the assumption that anomalies are rare and distinctly different from regular data points. As a result, they can be isolated more easily through random partitioning of the dataset. Anomalies are typically isolated in fewer steps. Thus they can be recognized as data points with the shortest average path length in the isolation tree. This innovative methodology allows Isolation Forest to detect anomalies with remarkable efficiency and accuracy (Puggini and McLoone, 2018), (Li et al., 2021).

In contrast, Autoencoders operate based on a principle of data reconstruction. The architecture of an Autoencoder is bifurcated into two primary segments: the encoder and the decoder. During the encoding phase, the input data undergoes transformations initiated by a Dense layer of 128 neurons with a 'relu' activation function. A He Normal initializer sets the initial weight values, and L2 regularization is employed to prevent overfitting. The encoding phase culminates with two LSTM layers, condensing the input data into a latent-space representation.

The decoder segment, essentially a mirror reflection of the encoder, begins with a Repeat Vector layer to replicate the input for the LSTM's expected format. This is succeeded by two LSTM layers, which gradually expand the latent representation. The final reconstruction is achieved via a Dense layer, which reconstructs the original input shape. The Autoencoder is trained using an Adam optimizer, which features a learning rate scheduler for Time-Based Decay and employs the mean squared error as the loss function. The Autoencoder produces a significantly higher reconstruction error when encountering an anomalous data point. This discrepancy can be effectively utilized as an anomaly score (Gong et al., 2019; Xu et al., 2021).

These two methodologies, Isolation Forest and Autoencoders, with their unique strengths and methodological differences, exemplify the vast array of techniques in the field of anomaly detection. Isolation Forest anomalies average the inherent rarity of anomalies, while Autoencoders strategically use reconstruction errors to identify anomalies. Their effectiveness underscores the diverse toolkit available to researchers and practitioners in this field.

5.10.4. Anomaly-based RDS: Runtime complexity

The anomaly-based RDS focuses on refining the classification of the transactions labeled as "White" from the previous stage, aiming to detect novel ransomware strains combination of unsupervised and supervised learning techniques, and follows a six-phase process.

(i) **The Unsupervised Phase:** The system uses the K-means algorithm to cluster the data, whose time complexity is $O(n \cdot K \cdot I \cdot d)$, where K is the number of clusters, and I is the number of iterations.

(ii) **The Labeling Phase:** This stage assigns labels to clusters, primarily using most instances within each cluster. The time complexity here is linear, i.e., $O(n)$.

(iii) **The Supervised Phase:** New instances are assigned to the nearest cluster and labeled based on the cluster's label. The time complexity of this phase is $O(n)$.

(iv) **The Biased Classifiers Training Data and Prediction Confidence Collection Phase:** This phase involves the calculation of true positives, true negatives, false positives, and false negatives. All of these computations have a linear time complexity.

(v) **The Biased Classifiers Phase:** This phase employs two biased classifiers (XG) trained to reduce false positives and negatives. The time complexity of this phase can be approximated as $O(n \cdot d)$.

(vi) **The Optuna Tuning Phase:** This phase fine-tunes the parameters for the Labeling Kmeans and Biased Classifier, whose time complexity is dependent on the number of hyperparameters and the number of trials, making it $O(T)$, where T is the total number of trials.

Consequently, the overall time complexity of the second stage is dominated by the K-means clustering, which can be approximated as $O(n \cdot K \cdot I \cdot d)$.

The time complexity of the entire system is mainly dictated by the most time-consuming stage, which is likely to be the first stage, considering that only "White" labeled instances are forwarded to the second stage. The time complexity, therefore, is $O(n \cdot m \cdot d)$ from the first stage, with an added time complexity from the second stage of $O(n \cdot K \cdot I \cdot d)$ for the label of the instance. as "White". This highlights that the number of instances contributes significantly to the computational complexity of the hybrid RDS.

5.11. Validation metrics

To assess the proposed framework's generalizability and mitigate overfitting issues, cross-validation, and hold-out methods are employed in the known ransomware detection experiments. The train-test-validation split and model evaluation procedures for each dataset are as follows:

1. **Train-Test Split:** A 70%–30% train-test split is employed, with 70% of the data samples allocated to the training set and 30% to the test set. The test set is kept untouched until the final hold-out validation, ensuring it remains independent for unbiased evaluation of the model's performance.
2. **Fivefold Cross-Validation:** Fivefold cross-validation is conducted on the training set to assess the performance of the proposed model. In each cross-validation fold, 80% of the original training set is utilized for model training, while the remaining 20% acts as the validation set for model testing. This process is repeated for ten iterations, ensuring comprehensive evaluation and robustness of the model's results.
3. **Test on Untouched Test Set:** After completing the training and validation steps using the fivefold cross-validation on the training set, the trained model is put to the final test on the untouched test set. This evaluation allows us to assess the model's performance on unseen data, providing valuable insights into its real-world effectiveness and generalization capabilities.

The selection of a 70%–30% train-test split and fivefold cross-validation is motivated by their standard usage and effectiveness in constructing robust validation methods. These approaches help mitigate overfitting and concept drift issues, which are crucial in ensuring reliable model performance. If the proposed method accurately detects rare ransomware instances during cross-validation and hold-out validation, it indicates the absence of underfitting or overfitting problems. This consistency in performance across different validation settings further strengthens the credibility and reliability of the model's ability to identify rare ransomware patterns effectively.

Hold-out validation plays a crucial role in evaluating the proposed system's performance, especially in detecting unknown attacks like zero-day ransomware. To achieve this, a validation set is carefully constructed for each zero-day ransomware type, consisting of all instances of that particular attack type and an equal number of randomly sampled normal data points. The remaining samples are then used to form the training set. The primary objective of this validation process is to assess the system's capability to detect patterns associated with each type of unknown ransomware. The underlying assumption is that new ransomware data is statistically more similar to other known ransomware instances than normal data. By subjecting the system to these unseen zero-day ransomware instances, we can effectively gauge its effectiveness in accurately identifying and distinguishing these previously unseen patterns from normal data. This evaluation approach provides valuable insights into the system's real-world adaptability and robustness, offering a critical assessment of its performance in handling novel and emerging threats.

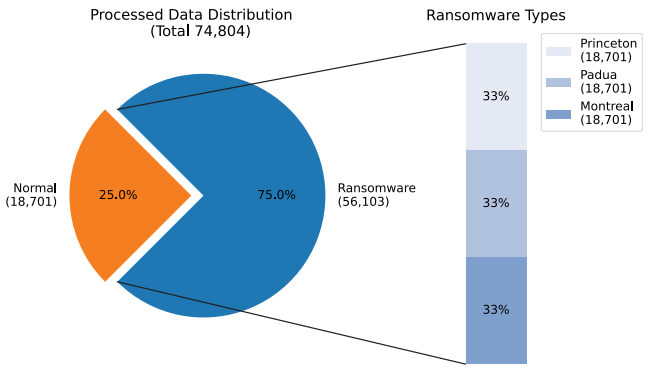


Fig. 4. Distribution of dataset after balancing the classes.

Several metrics are utilized to comprehensively evaluate the performance of the proposed Ransomware Detection System (RDS). These metrics include accuracy (Acc), precision, recall, F1 score, ROC score, and model execution time. They are derived from an analysis of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) within the context of the proposed model.

Accuracy (Acc) quantifies the overall correctness of the RDS predictions by considering the ratio of correctly classified samples (TP and TN) to the total number of samples (TP, TN, FP, and FN):

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

Precision characterizes the ability of the RDS to accurately identify positive instances (TP) relative to the total instances classified as positive (TP and FP):

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

Recall, also known as the true positive rate or sensitivity, gauges the ability of the RDS to correctly detect positive in relation to all actual positive instances (TP and FN):

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

The F1 score provides a balanced assessment of the RDS performance by harmonizing precision and recall. It is computed as the weighted harmonic mean of precision and recall, with higher F1 scores indicating better model performance:

$$\text{F1 score} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{9}$$

The ROC score is derived from the true positive rate (TPR) and false positive rate (FPR). It is an indicator of the RDS's ability to discriminate between positive and negative instances, with higher ROC scores signifying improved discrimination capability:

$$\text{ROC score} = \frac{TPR}{TPR + FPR} \tag{10}$$

Furthermore, the model execution time, which denotes the average duration needed for model training and validation in either fivefold cross-validation or hold-out validation, is a gauge of the RDS's efficiency. Optimal RDS systems aim to achieve high F1 scores (a combined metric of precision and recall) while maintaining low execution times. Balancing performance with computational efficiency is crucial in ensuring a practical and effective ransomware detection system.

6. Experimental study

6.1. Experimental setup

The development of the proposed RDS involved implementing feature engineering and ML algorithms using Python libraries, including Pandas (Team, 2020), Scikit-learn (Pedregosa et al., 2011), and Xgboost (Chen et al., 2015). The hyperparameter optimization (HPO) methods were implemented using the Optuna library (Akiba et al., 2019). The experiments were performed on a tower machine equipped with a 12-core Intel(R) Xeon(R) Platinum 8255C CPU @ 2.50 GHz and a GPU (Graphics Processing Unit) V100-SXM2-32 GB with 43 GB of memory. The machine was running CUDA version 11.7, which facilitated the

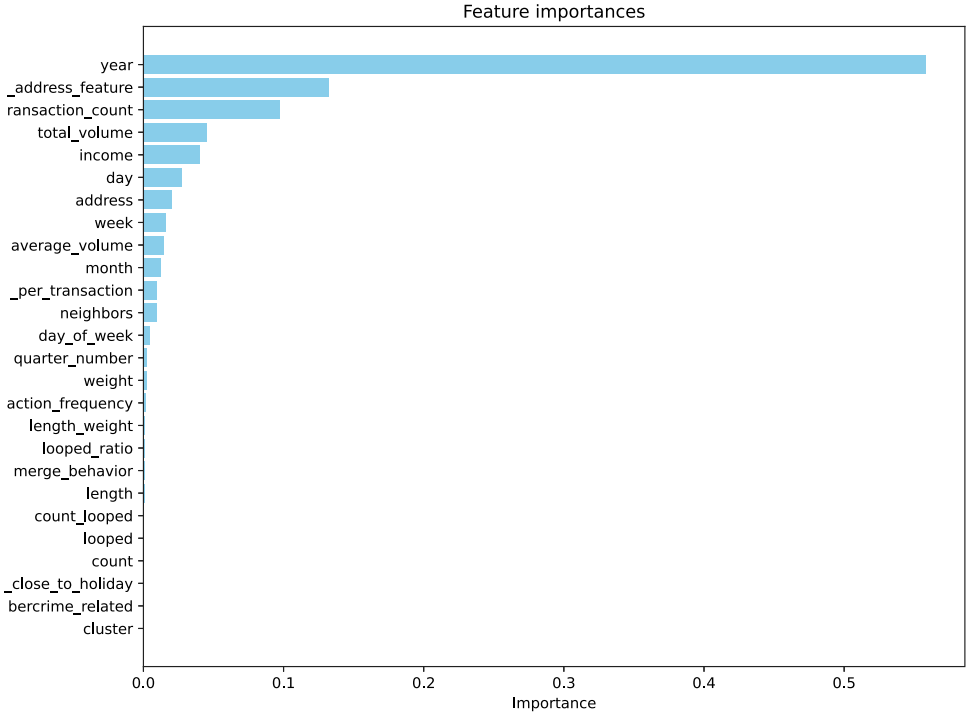


Fig. 5. Features importance after tuning.

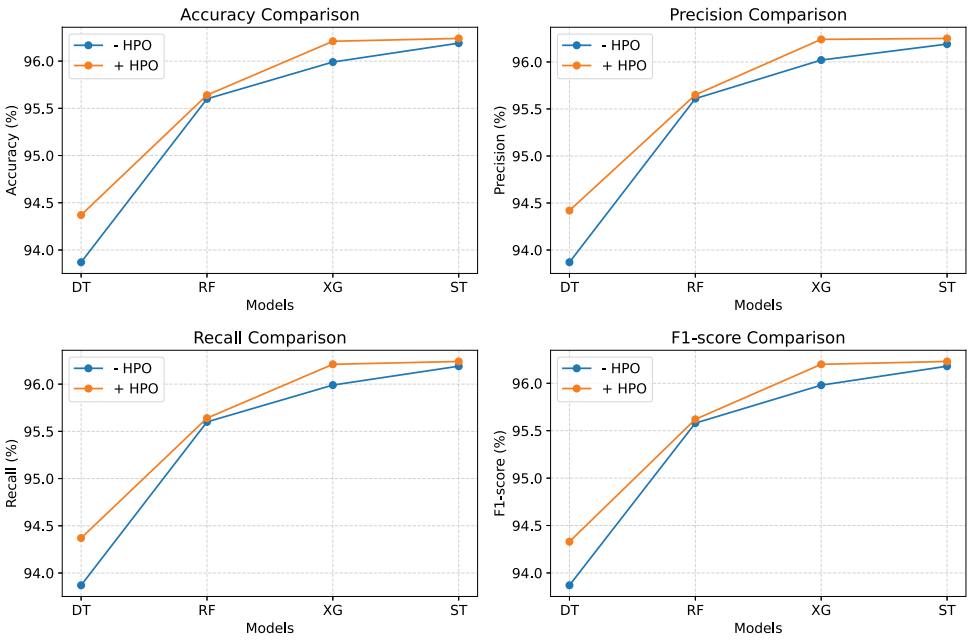


Fig. 6. Comparison of models' classification metrics in stage 1.

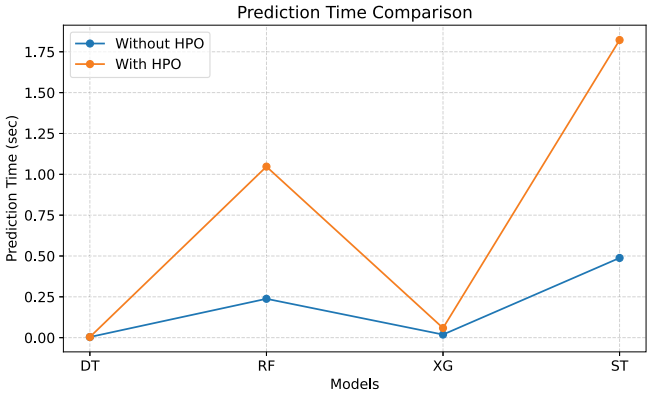


Fig. 7. Comparison of models' prediction time in stage 1.

utilization of GPU acceleration for efficient computations during the experiments.

The experiments were conducted in three distinct parts. In the first part, the emphasis was on data preprocessing, wherein various preprocessing techniques and feature engineering approaches were applied. The second part evaluated the signature-based RDS component using the labeled dataset for known ransomware detection. The system's performance in identifying known ransomware was rigorously assessed and analyzed during this phase. The anomaly-based RDS component was evaluated using unlabeled datasets for unknown ransomware detection in the third and final part. This critical phase aimed to determine the system's efficacy in detecting novel and previously unseen ransomware patterns, ensuring its ability to handle emerging threats.

6.2. Performance analysis of data preprocessing

In this research, SMOTE was applied to the minority class (ransomware) to generate synthetic instances and increase the representation of the minority class in the dataset. Random undersampling was performed on the majority class (normal) by randomly selecting a subset of instances to achieve a balanced dataset. Applying these techniques modified the dataset to have 74804 instances, including 56103 malicious and 18701 legitimate transactions. This balanced dataset enables more effective training of classification models and improves the overall classification performance. The new distribution of the dataset is shown in Fig. 4.

Following the systematic application of feature construction and selection methodologies, 22 attributes were obtained and delineated in Fig. 5. Four specific attributes, namely 'count,' 'is_close,' 'cybercrime_related,' and 'cluster,' were intentionally excluded due to their observed negligible impact on the overall predictive performance. This judicious exclusion ensured that only the most salient attributes were retained for subsequent analysis and model development, enhancing the model's efficiency.

6.3. Performance analysis of known ransomware detection

This section presents a comprehensive performance analysis of our proposed signature-based Ransomware Detection System (RDS) on the known ransomware attacks within the BitcoinHeist dataset. We compare the untuned and tuned versions of four machine learning models: Decision Tree (DT), Random Forest (RF), XGBoost, and Stacking. The evaluation is based on various performance metrics, including accuracy, precision, recall, F1-score, and prediction time.

The results of the untuned models demonstrate their effectiveness in detecting known ransomware attacks (see Table 6 and Fig. 6). The accuracy of the untuned models ranges from 93.87% (DT) to 96.19% (Stacking), indicating high classification performance. The precision and recall metrics exhibit similar values across the models, indicating a good balance between correctly classifying ransomware instances and avoiding false positives and negatives. The prediction times vary significantly, with the Decision Tree model being the fastest (0.004 s) and the Stacking model being the slowest (0.488 s) (see Fig. 7). After applying hyperparameter optimization (HPO) techniques, the tuned models show a slight performance improvement (see Fig. 6). The Stacking model achieves the highest accuracy (96.24%), precision (96.25%), recall (96.24%), and F1-score (96.23%) among all the models. However, the tuned Stacking model's prediction time increases to 1.822 s (see Fig. 7).

Compared to the state-of-the-art algorithms presented in Table 6, our proposed RDS performs better in detecting known ransomware

Table 6
Performance evaluation of the proposed framework on the known ransomware of the Bitcoin transactions dataset.

Framework	Model	Accuracy %	Precision %	Recall %	F1-score %	Prediction time (s)
Proposed algorithms without HPO	Decision tree	93.87	93.87	93.87	93.87	0.004
	Random forest	95.60	95.61	95.60	95.58	0.238
	XGBoost	95.99	96.02	95.99	95.98	0.019
	Stacking	96.19	96.19	96.19	96.18	0.488
Proposed algorithms with Optuna HPO	Decision tree	94.37	94.42	94.37	94.33	0.004
	Random forest	95.64	95.65	95.64	95.62	1.047
	XGBoost	96.21	96.24	96.21	96.20	0.059
	Stacking	**96.24**	**96.25**	**96.24**	**96.23**	1.822
State of the art algorithms	ANN, 2021 (Leef, 2023)	95.75	N/A	N/A	N/A	N/A
	ANN,2021 (Cahyani et al., 2021)	97	N/A	N/A	N/A	N/A
	RF, 2021 (Al Harrack, 2021)	94	94	94	94	N/A
	DT, 2021 (Al Harrack, 2021)	99	99	99	99	N/A
	XGBoost, 2023 (Alsaif et al., 2023)	89.09	71.6	73.88	72.72	N/A
	RF, 2019 (jihwankimqd, 2019)	74	76	69	72	N/A

attacks. Our approach outperforms the Decision Tree, Random Forest, and XGBoost models of the state-of-the-art algorithms in terms of accuracy, precision, recall, and F1-score. Additionally, our Stacking model achieves competitive results in terms of accuracy and precision when compared to state-of-the-art algorithms. It is worth mentioning that some performance metrics are not available for state-of-the-art algorithms; such cases are indicated with N/A in Table 6.

Notably, the DT version proposed by Al Harrack (2021) achieves a high accuracy of 99% for known ransomware detection. However, the RF model proposed by the same author achieves a significantly lower accuracy. This inconsistency suggests that the RF model may not fully leverage the potential of the RF algorithm, which is known for its ability to handle complex data and improve classification performance. Furthermore, in the work of Cahyani et al. (2021), the ANN model achieves a higher accuracy of 97% compared to our models. However, their study's absence of other classification metrics limits the comprehensive performance comparison. It is also worth mentioning that jihwankimqd (2019) provides their source code and reports an accuracy of 74% for known ransomware detection. This allows for a more transparent evaluation and comparison with our models. On the other hand, the lack of available source code in the remaining works hinders a thorough comparison.

To further investigate the behavior of our optimal model, the stacking approach, in detecting known ransomware attacks, we conducted an extensive analysis, presented in Fig. 8. The confusion matrix depicted in Fig. 8(a) provides a comprehensive overview of the model's classification performance and its normalized version. Additionally, Fig. 8(b) focuses on the errors in the confusion matrix.

Our analysis reveals that the stacking model accurately classified instances from different regions. Notably, 97% of instances from Montreal, 99% from Padua, 98% from Princeton, and 92% of normal instances were correctly classified. However, this relatively lower percentage for normal instances (92%) can be attributed to the balanced technique we employed to enhance the training process. A possible solution to improve the classification of normal instances is to include a larger proportion of such instances in the training set, likely boosting the detection rate.

Further scrutinizing the confusion matrix, we observed that among the misclassified normal instances, a significant portion (61%) was incorrectly labeled as Princeton (label 3) instances. This suggests some inherent similarities between normal and Princeton instances, calling for additional efforts to augment the training dataset for the Princeton label attack.

In the case of misclassified instances as Montreal (label 0), 74% of them were actually white. Conversely, instances incorrectly classified as white (label 3) exhibited a diverse composition, comprising 38% Princeton, 22% Padua, and 39% Montreal instances. This finding underscores the existence of certain resemblances among instances across all classes, indicating the necessity for more extensive datasets or models with a higher degree of freedom to address these complexities.

6.4. Performance analysis of unknown ransomware detection

In evaluating our anomaly-based RDS, we juxtapose the performance of three supervised learning classification algorithms, two unsupervised learning anomaly detection algorithms, and two of our proposed hybrid methods. The comparison aims to validate the superior generalizability of unsupervised learning over supervised learning in detecting novel and previously unseen ransomware. Our proposed hybrid methods, which effectively fuse both supervised and unsupervised learning advantages, consistently outperform the other methods in our tests.

In our experimental setup, we scrutinize three supervised learning methods: Support Vector Machine (SVM_HPO), XGBoost (XG_HPO), and Random Forest (RF_HPO). We also evaluate two unsupervised learning algorithms: Autoencoder (AE_HPO) and Isolation Forest 1u (IF_HPO). Our proposed algorithms incorporate supervised and unsupervised learning approaches: Labeling Kmeans (LK_HPO) and Labeling Kmeans with biased classifiers (LK_BC_HPO).

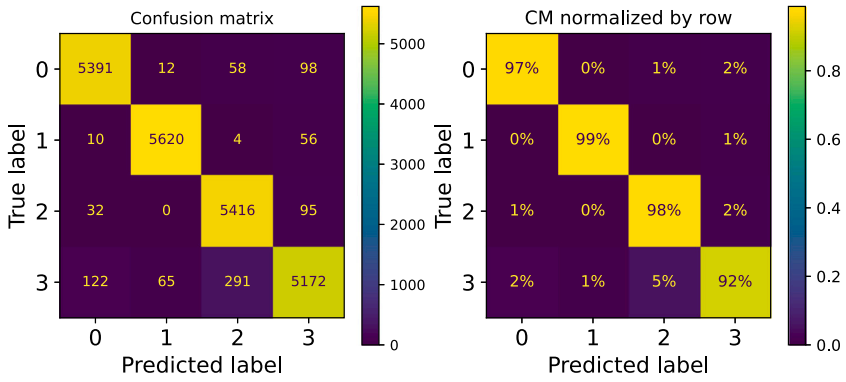
The experimental results presented in Table 7, averaged over three different ransomware families (Montreal, Padua, Princeton), unequivocally demonstrate the superior performance of our proposed LK_BC_HPO method across multiple evaluation criteria, including accuracy/recall, precision, F1-score, ROC-score, and prediction time. Note that the recall value aligns with accuracy in our experiment as it is the weighted recall. The results are further analyzed and illustrated in Figs. 9, and 10.

The LK_BC_HPO method outperforms traditional supervised learning algorithms and unsupervised learning approaches, including the Autoencoder and Isolation Forest. This superior performance is consistently reflected across all metrics. Moreover, the LK_BC_HPO approach exhibits a speed advantage in prediction time, surpassing traditional supervised models like SVM_HPO and RF_HPO.

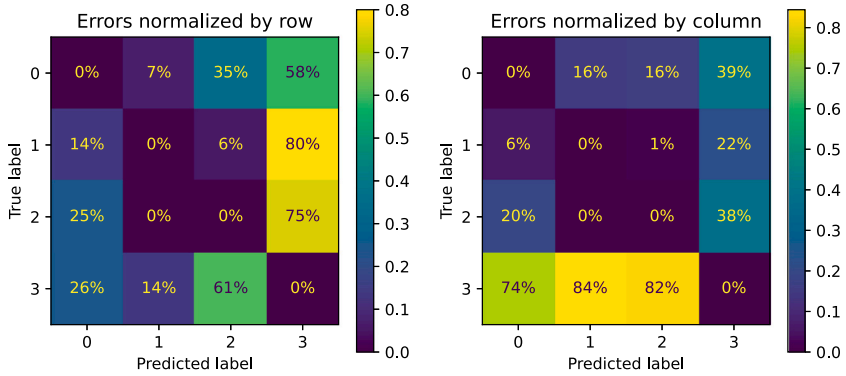
Notably, the high recall rate of the LK_BC_HPO model underscores its proficiency in accurately identifying known and novel ransomware instances. This high recall is crucial in ransomware detection, where the failure to detect a single ransomware instance can precipitate severe damage.

The LK_BC_HPO method also achieves a higher precision rate, indicating its ability to accurately classify actual ransomware instances among all labeled instances. This high precision minimizes false alarms, saving resources for investigating false positives.

Regarding the F1-score, a measure that harmonizes precision and recall, the LK_BC_HPO method further accentuates its dominance. A high F1 score indicates that the model balances identifying as many ransomware instances as possible (recall) and minimizing false positives (precision). Moreover, the LK_BC_HPO method showcases superior performance through its high ROC score, indicating the model's effectiveness in managing the trade-off between the true and false positive rates. The superior ROC score reveals the model's ability to



(a) Confusion Matrix of the Stacking model



(b) Confusion matrix Errors of the Stacking model in Stage 1

Fig. 8. Known attacks: Confusion matrix and errors of the stacking model.

maintain high detection rates while minimizing false alarms, even when the classification threshold varies.

When comparing the prediction times, it was observed that the LK_BC_HPO method outperforms the traditional models, including SVM_HPO and AE_HPO. This speed enables swift response to potential threats, a critical attribute in ransomware detection.

The primary strength of our proposed LK_BC_HPO system lies in its hybrid design, effectively combining supervised and unsupervised learning techniques. Unlike the Autoencoder (AE_HPO) and Isolation Forest (IF_HPO), the LK_BC_HPO approach leverages K-means clustering to identify organic patterns within the data based on feature characteristics rather than pre-defined labels, providing a broader and more flexible base for anomaly detection.

The subsequent labeling phase assigns classifications based on the majority composition of the clusters, ensuring any novel ransomware characteristics identified during the unsupervised phase are accounted for in the labeling. This step is crucial in deciding whether to engage the biased classifiers, and it represents a significant advancement over the unsupervised Autoencoder and Isolation Forest approaches.

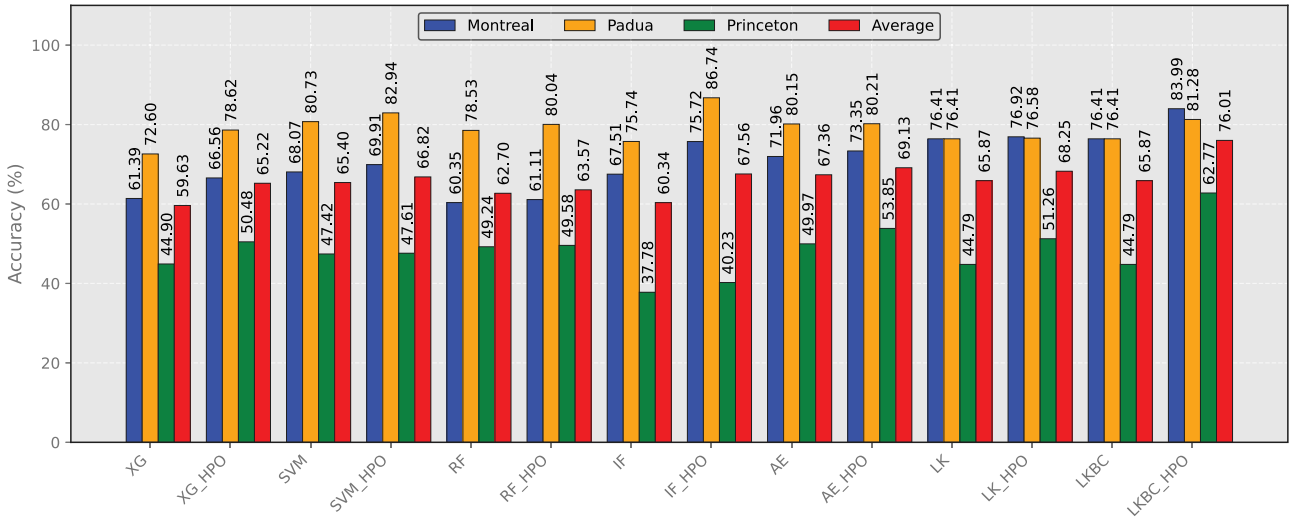
The biased classifiers in the LK_BC_HPO method further refine the model's ability to reduce false positives and negatives, thereby increasing the overall prediction accuracy. Unlike the standard Autoencoder and Isolation Forest methods, these classifiers are activated based on a confidence threshold, allowing the model to discern high-confidence predictions from more uncertain ones requiring further analysis.

The LK_BC_HPO method further enhances its performance with the Optuna Tuning phase. This optimizes critical parameters, such as the number of clusters and the confidence threshold, providing a more balanced and effective model than the Autoencoder and Isolation Forest methods.

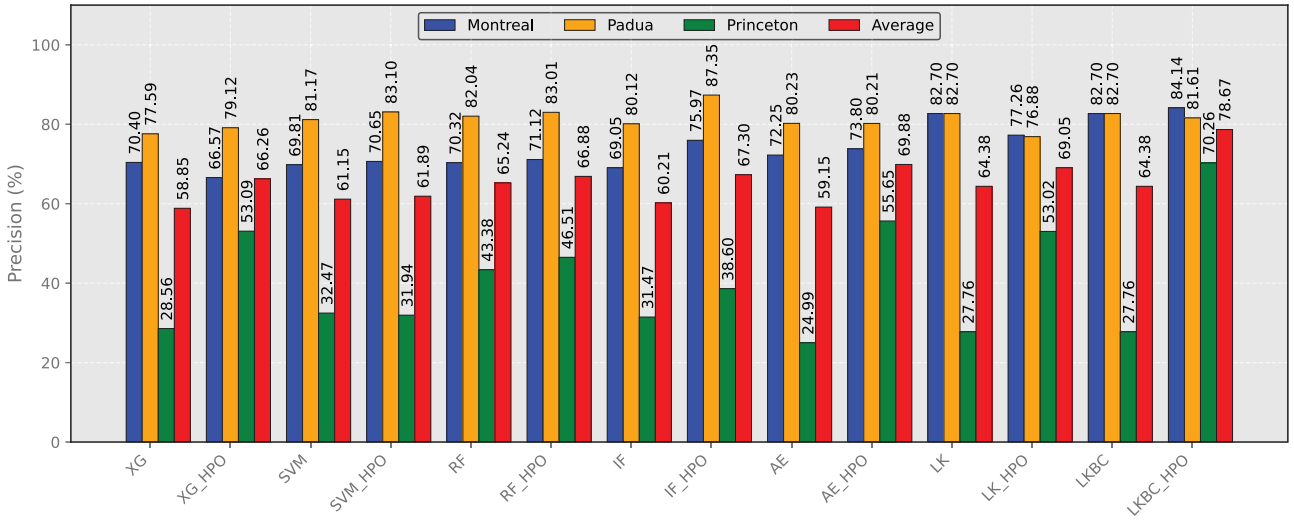
To sum up, our proposed LK_BC_HPO method presents a balanced, efficient, and highly accurate approach to ransomware detection. It harnesses the strengths of both supervised and unsupervised learning, integrates a nuanced approach to handling false positives and negatives, and applies effective parameter tuning. In contrast to methods like the Autoencoder and Isolation Forest, LK_BC_HPO addresses the critical challenge of detecting known and novel ransomware variants, positioning it as a robust and versatile tool in the ongoing battle against ransomware.

Nonetheless, the system under review presents several limitations, requiring further refinement and investigation.

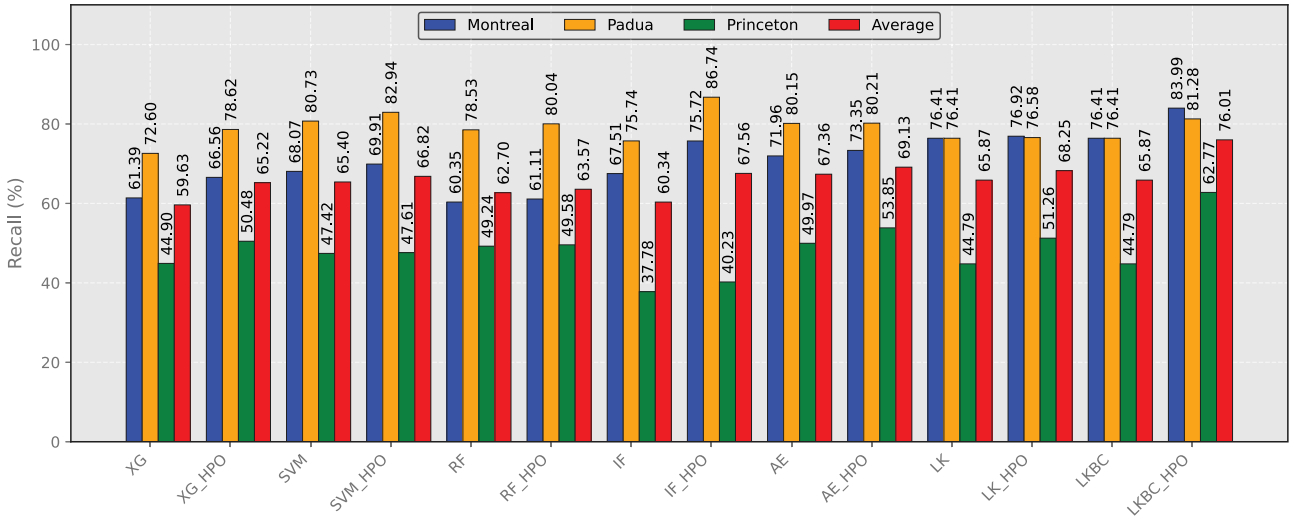
1. **Dependence on Quality and Completeness of Training Data:** The system's efficacy is predominantly contingent on the quality and comprehensiveness of the training data. If the training data fails to encompass the wide gamut of ransomware behaviors, the system could potentially falter in accurately categorizing novel instances. This consideration is especially crucial during the unsupervised phase, wherein the system depends on clustering to discern patterns. Upon the evaluation phase, it was observed that the effectiveness varied according to the ransomware family type, Padua, Montreal, and Princeton, in that order. A particularly concerning finding was the poor performance of most algorithms under the Princeton ransomware family test, often displaying accuracy rates lower than 50%. In our proposed LK_BC_HPO, it only achieved 62.77%. This suggests that certain novel ransomware family structures were nearly indistinguishable from normal or white transactions during simulation, making it challenging for current algorithms to classify them accurately. Consequently, this could pose significant obstacles in real-world tests, resulting in system instability under various conditions.
2. **Potential for Misclassification in the Labeling Phase:** During the labeling phase, clusters are classified according to the majority class of their constituent members, which could lead to potential misclassification of instances belonging to a minority class within a cluster. The system may also encounter difficulties with clusters of an approximately equal mix of different classes.



(a) Accuracy comparison of models

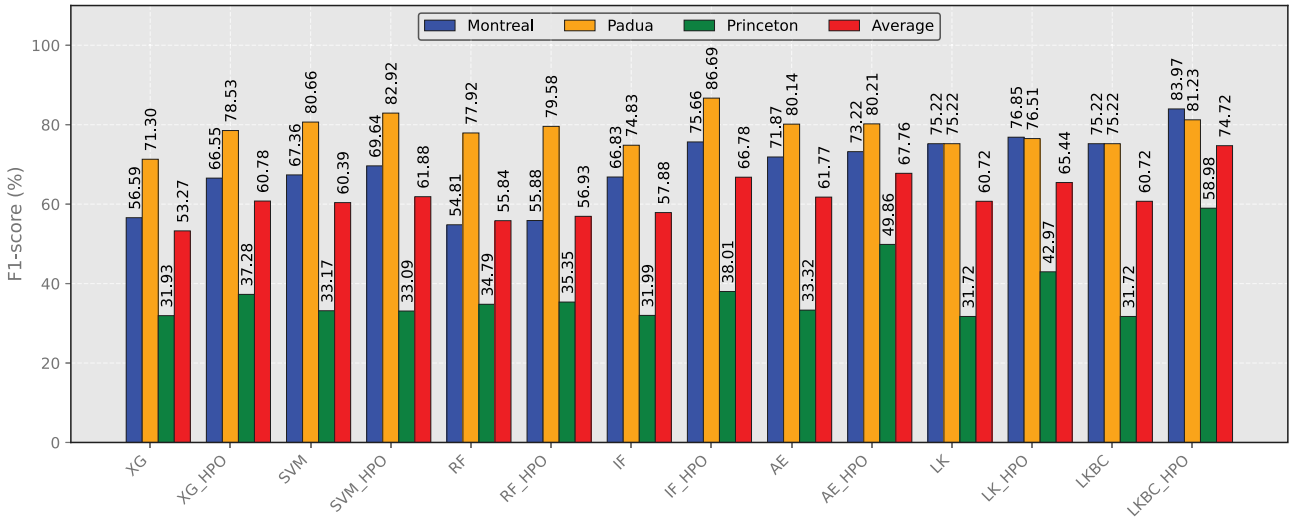


(b) Precision comparison of models

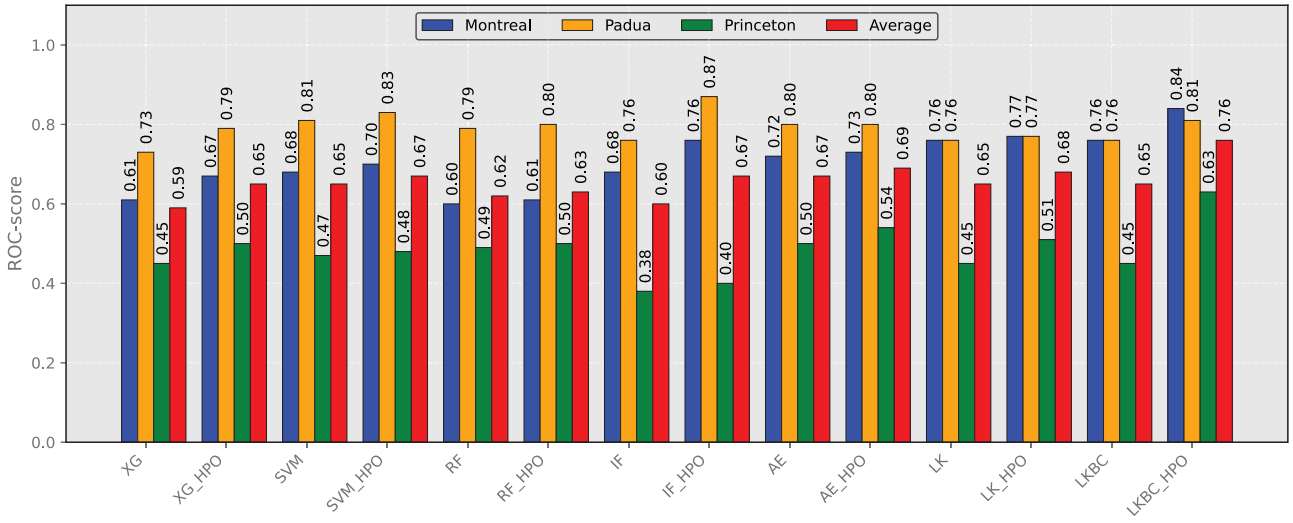


(c) Recall comparison of models

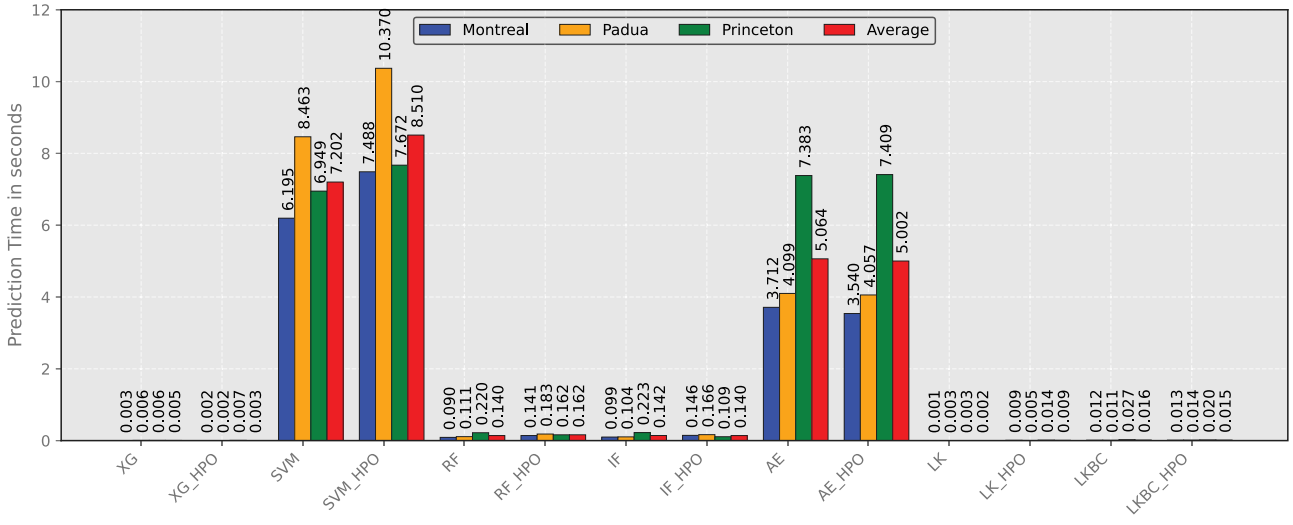
Fig. 9. Unknown ransomware: Performance comparison of models.



(a) F1-score comparison of models



(b) ROC-score comparison of models



(c) Prediction time comparison of models

Fig. 10. Unknown ransomware: Performance comparison of models.

Table 7
Performance evaluation of the proposed framework on the **unknown** ransomware of the Bitcoin transactions dataset.

Algorithm	Ransomware	Accuracy %	Precision %	Recall %	F1-score %	ROC-score	P-time (s)
XGBoost	Montreal	61.39	70.40	61.39	56.59	0.61	0.003
	Padua	72.60	77.59	72.60	71.30	0.73	0.006
	Princeton	44.90	28.56	44.90	31.93	0.45	0.006
	Average	59.63	58.85	59.63	53.27	0.59	0.005
XGBoost_HPO	Montreal	66.56	66.57	66.56	66.55	0.67	0.002
	Padua	78.62	79.12	78.62	78.53	0.79	0.002
	Princeton	50.48	53.09	50.48	37.28	0.50	0.007
	Average	65.22	66.26	65.22	60.78	0.65	0.003
SVM	Montreal	68.07	69.81	68.07	67.36	0.68	6.195
	Padua	80.73	81.17	80.73	80.66	0.81	8.463
	Princeton	47.42	32.47	47.42	33.17	0.47	6.949
	Average	65.40	61.15	65.40	60.39	0.65	7.202
SVM_HPO	Montreal	69.91	70.65	69.91	69.64	0.70	7.488
	Padua	82.94	83.10	82.94	82.92	0.83	10.37
	Princeton	47.61	31.94	47.61	33.09	0.48	7.672
	Average	66.82	61.89	66.82	61.88	0.67	8.51
RF	Montreal	60.35	70.32	60.35	54.81	0.60	0.090
	Padua	78.53	82.04	78.53	77.92	0.79	0.111
	Princeton	49.24	43.38	49.24	34.79	0.49	0.220
	Average	62.70	65.24	62.70	55.84	0.62	0.140
RF_HPO	Montreal	61.11	71.12	61.11	55.88	0.61	0.141
	Padua	80.04	83.01	80.04	79.58	0.80	0.183
	Princeton	49.58	46.51	49.58	35.35	0.50	0.162
	Average	63.57	66.88	63.57	56.93	0.63	0.162
IForest	Montreal	67.51	69.05	67.51	66.83	0.68	0.099
	Padua	75.74	80.12	75.74	74.83	0.76	0.104
	Princeton	37.78	31.47	37.78	31.99	0.38	0.223
	Average	60.34	60.21	60.34	57.88	0.60	0.142
IForest_HPO	Montreal	75.72	75.97	75.72	75.66	0.76	0.146
	Padua	86.74	87.35	86.74	86.69	0.87	0.166
	Princeton	40.23	38.60	40.23	38.01	0.40	0.109
	Average	67.56	67.30	67.56	66.78	0.67	0.140
AE	Montreal	71.96	72.25	71.96	71.87	0.72	3.712
	Padua	80.15	80.23	80.15	80.14	0.8	4.099
	Princeton	49.97	24.99	49.97	33.32	0.50	7.383
	Average	67.36	59.15	67.36	61.77	0.67	5.064
AE_HPO	Montreal	73.35	73.80	73.35	73.22	0.73	3.540
	Padua	80.21	80.21	80.21	80.21	0.8	4.057
	Princeton	53.85	55.65	53.85	49.86	0.54	7.409
	Average	69.13	69.88	69.13	67.76	0.69	5.002
LK	Montreal	76.41	82.70	76.41	75.22	0.76	0.001
	Padua	76.41	82.70	76.41	75.22	0.76	0.003
	Princeton	44.79	27.76	44.79	31.72	0.45	0.003
	Average	65.87	64.38	65.87	60.72	0.65	0.002
LK_HPO	Montreal	76.92	77.26	76.92	76.85	0.77	0.009
	Padua	76.58	76.88	76.58	76.51	0.77	0.005
	Princeton	51.26	53.02	51.26	42.97	0.51	0.014
	Average	68.25	69.05	68.25	65.44	0.68	0.009
LKBC	Montreal	76.41	82.70	76.41	75.22	0.76	0.012
	Padua	76.41	82.70	76.41	75.22	0.76	0.011
	Princeton	44.79	27.76	44.79	31.72	0.45	0.027
	Average	65.87	64.38	65.87	60.72	0.65	0.016
LKBC_HPO	Montreal	83.99	84.14	83.99	83.97	0.84	0.013
	Padua	81.28	81.61	81.28	81.23	0.81	0.014
	Princeton	62.77	70.26	62.77	58.98	0.63	0.020
	Average	76.01	78.67	76.01	74.72	0.76	0.015

3. **Risk of Overfitting with Biased Classifiers:** Although the innovative use of biased classifiers has the potential to decrease false positives and negatives, it could risk overfitting, especially if the training data for these classifiers lacks representative coverage of all potential instances.
4. **Limitations of Anomaly-Based Detection:** Anomaly-based detection systems identify deviating behavioral patterns from the 'normal' as potential threats. Although this can detect new, unseen threats effectively, it also poses a higher risk of false positives than signature-based detection. This is because legitimate behavior, especially in complex systems, can sometimes appear anomalous. For instance, introducing new software could generate unfamiliar network traffic patterns, which the system could erroneously flag as suspicious, leading to disruptive false alarms requiring additional resources for investigation. In contrast, signature-based detection systems, which look for specific malicious behavioral 'signatures,' are less prone to false positives and less effective against novel threats without known signatures.

6.5. Project reproducibility

Ensuring the reproducibility of our project is of paramount importance for scientific integrity. To facilitate this, we have made the complete source code of our project publicly available on GitHub. The repository can be accessed using the following URL: https://github.com/odib/Bitcoin_Heist_Classification. By accessing the repository, readers can inspect the code, explore the project structure, and understand the implementation details comprehensively.

The repository includes a detailed README file that provides instructions on how to set up and utilize the source code effectively. This documentation assists users in reproducing the project environment and replicating the experimental results.

To enhance the readability and transparency of our work, we have utilized Jupyter Notebooks. These notebooks contain the code snippets, data preprocessing steps, ML model implementations, and the corresponding results. The interactive nature of Jupyter Notebooks allows readers to navigate through the code and observe the output easily. By referring to these notebooks, users can gain deeper insights into our methodology and reproduce the experimental outcomes.

By providing open access to the source code and utilizing Jupyter Notebooks, we strive to promote transparency, enable reproducibility, and facilitate the verification of our project by the scientific community.

7. Conclusions and future works

7.1. Conclusions

This research paper has proposed a novel Hybrid Ransomware Detection System framework for accurately detecting ransomware attacks within the BitcoinHeist dataset. The framework combines the strengths of signature-based and anomaly-based detection approaches, leveraging supervised and unsupervised learning techniques.

The key contributions of this research include introducing new features, a feature engineering model based on the Gradient Boosting Classifier, an anomaly-based RDS utilizing L-kmeans and biased classifiers, and applying hyperparameter optimization techniques.

The performance evaluation of the proposed RDS on the BitcoinHeist dataset demonstrates its superiority compared to state-of-the-art algorithms. Our models achieve high accuracy, precision, recall, F1-score, and ROC score, outperforming existing ransomware detection models. The experimental results show that the RDS framework effectively detects known and unknown ransomware attacks within cryptocurrency transactions.

7.2. Future works

Future work encompasses testing the proposed framework under real-time settings to assess its performance in detecting ransomware attacks in real-world scenarios. This will involve integrating the framework into live cryptocurrency transaction systems and evaluating its effectiveness in detecting and preventing real-time ransomware attacks.

Additionally, extending the research to other cryptocurrencies and blockchain technologies is an important direction for future investigations. As different cryptocurrencies and blockchain platforms have unique characteristics and transaction structures, evaluating the proposed framework's applicability and adaptability to diverse cryptocurrency ecosystems will provide valuable insights into its robustness and generalizability.

Furthermore, it would be beneficial to explore the integration of external threat intelligence sources into the RDS framework. The framework can enhance its predictive capabilities and stay ahead of evolving ransomware techniques by incorporating up-to-date information on emerging ransomware threats and attack patterns.

Another avenue for future research is to investigate the scalability and performance of the framework when applied to larger and more complex datasets. Testing the framework on diverse datasets with varying levels of ransomware prevalence and sophistication will provide a comprehensive assessment of its effectiveness and scalability.

Besides, considering the dynamic nature of ransomware attacks, continuous monitoring and updating of the RDS framework with the latest ransomware signatures and attack patterns will be crucial. This includes establishing mechanisms for timely updates and incorporating feedback loops to adapt the model to new and emerging ransomware threats.

Finally, efficient detection of ransomware attacks constitutes a crucial aspect of the decision-making toolkit. Equally vital is the automation of subsequent decision-making steps involving the investigation of ransomware origins and their association with relevant authorities for enhanced automation. To achieve this, our plan involves integrating an additional intelligent module to link ransomware transactions with the authentic identities responsible for such actions.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abraham, J.A., George, S.M., 2019. A survey on preventing crypto ransomware using machine learning. In: 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies. ICICICT, Vol. 1, IEEE, pp. 259–263. http://dx.doi.org/10.1109/icicict46008.2019.8993137.

Akcora, C.G., Li, Y., Gel, Y.R., Kantarcioglu, M., 2019. Bitcoinheist: Topological data analysis for ransomware detection on the bitcoin blockchain. arXiv preprint arXiv:1906.07852, http://dx.doi.org/10.48550/ARXIV.1906.07852, URL: https://arxiv.org/abs/1906.07852.

Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., 2019. Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '19, ACM, pp. 2623–2631. http://dx.doi.org/10.1145/3292500.3330701.

Al-Haija, Q.A., Alsulami, A.A., 2021. High performance classification model to identify ransomware payments for heterogeneous bitcoin networks. Electronics 10 (17), 2113. http://dx.doi.org/10.3390/electronics10172113.

Al Harrack, M., 2021. The BitcoinHeist: Classifications of ransomware crime families. Int. J. Comput. Sci. Inf. Technol. (IJCSIT) 13, 75–81. http://dx.doi.org/10.5121/ijcsit.2021.13506.

Al-rimy, B.A.S., Maarof, M.A., Shaid, S.Z.M., 2018. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. Comput. Secur. 74, 144–166. http://dx.doi.org/10.1016/j.cose.2018.01.001, URL: https://www.sciencedirect.com/science/article/pii/S016740481830004X.

Alhawi, O.M., Baldwin, J., Dehghantanha, A., 2018. Leveraging machine learning techniques for windows ransomware network traffic detection. In: Cyber Threat Intelligence. Springer International Publishing, pp. 93–106. http://dx.doi.org/10.1007/978-3-319-73951-9_5.

Almashhadani, A.O., Kaiiali, M., Sezer, S., O'Kane, P., 2019. A multi-classifier network-based crypto ransomware detection system: A case study of locky ransomware. IEEE Access 7, 47053–47067. http://dx.doi.org/10.1109/access.2019.2907485.

Alsaif, S.A., et al., 2023. Machine learning-based ransomware classification of bitcoin transactions. Appl. Comput. Intell. Soft Comput. 2023, 1–10. http://dx.doi.org/10.1155/2023/6274260.

Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T., Capkun, S., 2013. Evaluating User Privacy in Bitcoin. In: Lecture Notes in Computer Science, Springer Berlin Heidelberg, Springer, pp. 34–51. http://dx.doi.org/10.1007/978-3-642-39884-1_4.

Atkinson, A.C., Riani, M., Corbellini, A., 2021. The box-cox transformation: review and extensions. Statist. Sci. 36 (2), 239–255. http://dx.doi.org/10.1214/20-STS778.

Cahyani, N.D.W., Nuha, H.H., et al., 2021. Ransomware detection on bitcoin transactions using artificial neural network methods. In: 2021 9th International Conference on Information and Communication Technology. ICoICT, IEEE, pp. 1–5. http://dx.doi.org/10.1109/icoict52021.2021.9527414.

Canto, A.C., Kaur, J., Kermani, M.M., Azarderakhsh, R., 2023. Algorithmic security is insufficient: A comprehensive survey on implementation attacks haunting post-quantum security. arXiv preprint arXiv:2305.13544.

Chandrasekharuni, Y., 2021. Using AI to detect Bitcoin addresses involved in ransomware transactions. Retrieved April 19 from https://medium.com/analytics-vidhya/using-ai-to-detect-bitcoin-addresses-involved-in-ransomware-transactions-3beaeccba176.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., et al., 2015. Xgboost: extreme gradient boosting. pp. 1–4, R package version 0.4-2 1.

Chowdhury, M.A.H., Mumenin, N., Taus, M., Yousuf, M.A., 2021. Detection of compatibility, proximity and expectancy of bengali sentences using long short term memory. In: 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques. ICREST, IEEE, pp. 233–237. http://dx.doi.org/10.1109/icrest51555.2021.9331057.

Cintas-Canto, A., Kermani, M.M., Azarderakhsh, R., 2022a. Reliable architectures for finite field multipliers using cyclic codes on FPGA utilized in classic and post-quantum cryptography. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 31 (1), 157–161.

Cintas-Canto, A., Mozaffari-Kermani, M., Azarderakhsh, R., Gaj, K., 2022b. CRC-oriented error detection architectures of post-quantum cryptography niederreiter key generator on FPGA. In: 2022 IEEE Nordic Circuits and Systems Conference. NorCAS, pp. 1–7. http://dx.doi.org/10.1109/NorCAS57515.2022.9934378.

Conti, M., Kumar, E.S., Lal, C., Ruj, S., 2018. A survey on security and privacy issues of bitcoin. IEEE Commun. Surv. Tutor. 20 (4), 3416–3452. http://dx.doi.org/10.1109/comst.2018.2842460.

Davis, J.J., Clark, A.J., 2011. Data preprocessing for anomaly based network intrusion detection: A review. Comput. Secur. 30 (6–7), 353–375. http://dx.doi.org/10.1016/j.cose.2011.05.008, URL: https://www.sciencedirect.com/science/article/pii/S0167404811000691.

Di Battista, G., Di Donato, V., Patrignani, M., Pizzonia, M., Roselli, V., Tamassia, R., 2015. Bitconeview: visualization of flows in the bitcoin transaction graph. In: 2015 IEEE Symposium on Visualization for Cyber Security. VizSec, IEEE, pp. 1–8. http://dx.doi.org/10.1109/vizsec.2015.7312773.

Dib, O., Brousmiche, K.-L., Durand, A., Thea, E., Hamida, E.B., 2018. Consortium blockchains: Overview, applications and challenges. Int. J. Adv. Telecommun. 11 (1), 51–64.

Elkhatib, R., Azarderakhsh, R., Mozaffari-Kermani, M., 2021. Accelerated RISC-V for SIKE. In: 2021 IEEE 28th Symposium on Computer Arithmetic. ARITH, pp. 131–138. http://dx.doi.org/10.1109/ARITH51176.2021.00035.

Feder, A., Gandal, N., Hamrick, J., Moore, T., 2018. The impact of DDoS and other security shocks on Bitcoin currency exchanges: Evidence from Mt. Gox. J. Cybersecur. 3 (2), 137–144. http://dx.doi.org/10.1093/cybsec/tyx012.

Garg, A., Maheshwari, P., 2016. Performance analysis of snort-based intrusion detection system. In: 2016 3rd International Conference on Advanced Computing and Communication Systems. Icaccs, Vol. 1, IEEE, pp. 1–5. http://dx.doi.org/10.1109/icaccs.2016.7586351.

Goldsmith, D., Grauer, K., Shmalo, Y., 2020. Analyzing hack subnetworks in the bitcoin transaction graph. Appl. Netw. Sci. 5 (1), 1–20. http://dx.doi.org/10.1007/s41109-020-00261-7.

Gong, D., Liu, L., Le, V., Saha, B., Mansour, M.R., Venkatesh, S., Hengel, A.v.d., 2019. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1705–1714.

Huang, D.Y., Aliapoulios, M.M., Li, V.G., Invernizzi, L., Bursztein, E., McRoberts, K., Levin, J., Levchenko, K., Snoeren, A.C., McCoy, D., 2018. Tracking ransomware end-to-end. In: 2018 IEEE Symposium on Security and Privacy. SP, IEEE, pp. 618–631. http://dx.doi.org/10.1109/SP.2018.00047.

jihwankimqd, 2019. Bitcoin heist classification project. Retrieved July 15 from https://github.com/jihwankimqd/Bitcoin_Heist_Classification.

Joseph, D., Misoczki, R., Manzano, M., Tricot, J., Pinuaga, F.D., Lacombe, O., Leichenauer, S., Hidary, J., Venables, P., Hansen, R., 2022. Transitioning organizations to post-quantum cryptography. Nature 605 (7909), 237–243.

Kermani, M.M., Azarderakhsh, R., 2016. Lightweight hardware architectures for fault diagnosis schemes of efficiently-maskable cryptographic substitution boxes. In: 2016 IEEE International Conference on Electronics, Circuits and Systems. ICECS, pp. 764–767. http://dx.doi.org/10.1109/ICECS.2016.7841314.

Kok, S., Abdullah, A., Jhanjhi, N., Supramaniam, M., 2019. Prevention of crypto-ransomware using a pre-encryption detection algorithm. Computers 8 (4), 79. http://dx.doi.org/10.3390/computers8040079.

Leef, D., 2023. BitCaught: Accurately identifying ransomware related and generally malicious bitcoin addresses using machine learning and past blockchain activity. URL: https://dleef.github.io/BitCaughtReport.pdf [Accessed 26-December-2023].

Leng, L., Li, M., Kim, C., Bi, X., 2017. Dual-source discrimination power analysis for multi-instance contactless palmprint recognition. Multimedia Tools Appl. 76, 333–354.

Leng, L., Li, M., Teoh, A.B.J., 2013. Conjugate 2DPalmHash code for secure palm-print-vein verification. In: 2013 6th International Congress on Image and Signal Processing. CISP, Vol. 3, IEEE, pp. 1705–1710.

Leng, L., Zhang, J., 2013. Palmhash code vs. palmphasor code. Neurocomputing 108, 1–12.

Li, Z., Zeng, Q., Liu, Y., Liu, J., Li, L., 2021. An improved traffic lights recognition algorithm for autonomous driving in complex scenarios. Int. J. Distrib. Sens. Netw. 17 (5), 15501477211018374. http://dx.doi.org/10.1177/15501477211018374.

Liao, K., Zhao, Z., Doupé, A., Ahn, G.-J., 2016. Behind closed doors: measurement and analysis of CryptoLocker ransoms in Bitcoin. In: 2016 APWG Symposium on Electronic Crime Research. ECrime, IEEE, pp. 1–13. http://dx.doi.org/10.1109/ECRIME.2016.7487938.

Lim, I.-K., Kim, Y.-H., Lee, J.-G., Lee, J.-P., Nam-Gung, H., Lee, J.-K., 2014. The Analysis and Countermeasures on Security Breach of Bitcoin. In: Lecture Notes in Computer Science, Springer International Publishing, Springer, pp. 720–732. http://dx.doi.org/10.1007/978-3-319-09147-1_52.

Moore, T., Christin, N., Szurdi, J., 2018. Revisiting the risks of bitcoin currency exchange closure. ACM Trans. Internet Technol. (TOIT) 18 (4), 1–18. http://dx.doi.org/10.1145/3155808.

Möser, M., Böhme, R., 2017. The price of anonymity: empirical evidence from a market for Bitcoin anonymization. J. Cybersecur. 3 (2), 127–135. http://dx.doi.org/10.1093/cybsec/tyx007.

Mozaffari Kermani, M., Azarderakhsh, R., Mirakhorli, M., Multidisciplinary approaches and challenges in integrating emerging medical devices security research and education. In: 2016 ASEE Annual Conference & Exposition Proceedings, ASEE Conferences, http://dx.doi.org/10.18260/p.25761.

Niasar, M.B., Azarderakhsh, R., Kermani, M.M., 2020. Optimized architectures for elliptic curve cryptography over Curve448. Cryptol. ePrint Arch..

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Puggini, L., McLoone, S., 2018. An enhanced variable selection and Isolation Forest based methodology for anomaly detection with OES data. Eng. Appl. Artif. Intell. 67, 126–135. http://dx.doi.org/10.1016/j.engappai.2017.09.021.

Ruoti, S., Kaiser, B., Yerukhimovich, A., Clark, J., Cunningham, R., 2019. Blockchain technology: what is it good for? Commun. ACM 63 (1), 46–53. http://dx.doi.org/10.1145/3369752.

Sahni, D., Pappu, S.J., Bhatt, N., 2021. Aided selection of sampling methods for imbalanced data classification. In: Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management of Data (8th ACM IKDD CODS & 26th COMAD). In: CODS COMAD 2021, ACM, pp. 198–202. http://dx.doi.org/10.1145/3430984.3431029.

Sarker, A., Mozaffari Kermani, M., Azarderakhsh, R., 2022. Efficient error detection architectures for postquantum signature falcon's sampler and KEM SABER. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 30 (6), 794–802. http://dx.doi.org/10.1109/TVLSI.2022.3156479.

Shu, X., Zhang, S., Li, Y., Chen, M., 2022. An anomaly detection method based on random convolutional kernel and isolation forest for equipment state monitoring. In: Eksploatacja i Niezawodność. Vol. 24, http://dx.doi.org/10.17531/ein.2022.4.16.

Singh, K., Dib, O., Huyart, C., Toumi, K., 2020. A novel credential protocol for protecting personal attributes in blockchain. Comput. Electr. Eng. 83, 106586. http://dx.doi.org/10.1016/j.compeleceng.2020.106586, URL: https://www.sciencedirect.com/science/article/pii/S0045790619315988.

Sinsomboonthong, S., 2022. Performance comparison of new adjusted min-max with decimal scaling and statistical column normalization methods for artificial neural network classification. Int. J. Math. Math. Sci. 2022, 1–9. http://dx.doi.org/10.1155/2022/3584406.

Team, T., 2020. Pandas Development Pandas-Dev/Pandas: Pandas, Vol. 21, Zenodo, pp. 1–9.

Uddin, S., Khan, A., Hossain, M.E., Moni, M.A., 2019. Comparing different supervised machine learning algorithms for disease prediction. BMC Med. Inform. Decis. Mak. 19 (1), 1–16. http://dx.doi.org/10.1186/s12911-019-1004-8.

Weisberg, S., 2001. Yeo-Johnson Power Transformations. Department of Applied Statistics, University of Minnesota, Retrieved June 1, 2003.

Xu, S., 2021. The application of machine learning in bitcoin ransomware family prediction. In: 2021 the 5th International Conference on Information System and Data Mining. ICISDM '21, Association for Computing Machinery, pp. 21–27. http://dx.doi.org/10.1145/3471287.3471300.

Xu, W., Jang-Jaccard, J., Singh, A., Wei, Y., Sabrina, F., 2021. Improving performance of autoencoder-based network anomaly detection on NSL-KDD dataset. IEEE Access 9, 140136–140146. http://dx.doi.org/10.1109/access.2021.3116612.

Zhang, Z., Song, X., Liu, L., Yin, J., Wang, Y., Lan, D., 2021. Recent advances in blockchain and artificial intelligence integration: feasibility analysis, research issues, applications, challenges, and future work. Secur. Commun. Netw. 2021, 1–15. http://dx.doi.org/10.1155/2021/9991535.