Knowledge Discovery in the Bitcoin Market: Study on a High-Volume Data Set.

by

José Antonio Blanco Mellado

A thesis submitted in conformity with the requirements
for the MSc in Economics, Finance and Computer Science

University of Huelva & International University of Andalusia

December 2021

Knowledge Discovery in the Bitcoin Market: Study on a High-Volume Data Set

José Antonio Blanco Mellado

Máster en Economía, Finanzas y Computación

Supervisor: Dr. Antonio J. Tallón Ballesteros
Universidad de Huelva y Universidad Internacional de Andalucía

2021

Abstract

Knowledge Discovery in Databases makes possible analysing the cybercriminal activity on Bitcoin Market. Ransomware attacks are one of the most dangerous related crimes in the coins market; this type of malware forces victims to pay after encryption of the victim's file. This Master's dissertation aims to analyse this context using machine learning techniques through bitcoin transactions data set. The baseline data set containing all the attributes has been compared with two reduced subsets applying feature selection methods in an independent way: Extended Correlation-based Feature Selection (eCFS) and Correlation-based Feature Selection with Exhaustive Search (XCFS). CART from the trees' family, Random Forest and XGBoost, as ensemble methods, and kNN, which is lazy algorithm, have been selected to achieve our results. We focus on kappa and accuracy metrics to evaluate on unseen data the trained classification models, and precision, recall and f1-score to study the detection performance on ransomware classes studied.

JEL classification: D47, E42, G10, K42.

Key words: KDD, Bitcoin Market, Supervised Machine Learning, Data Mining, Big Data, CFS, Feature Selection, Classification.

Resumen

La metodología *Knowledge Discovery in Databases* posibilita el análisis del cibercrimen en el mercado de Bitcoin. Los ataques de *ransomware* son uno de los delitos más peligrosos en el mercado de las criptomonedas; este tipo de *malware* obliga a las víctimas a pagar después de cifrar el archivo de la víctima. Este TFM tiene como objetivo detectar *ransomware* utilizando técnicas de aprendizaje automático utilizando un conjunto de datos de transacciones de bitcoin. El conjunto de datos base se ha comparado con dos subconjuntos reducidos que aplican métodos de selección de atributos: correlación extendida (eCFS) y correlación con búsqueda exhaustiva (XCFS). Los modelos CART, Random Forest, XGBoost, y *k*NN, han sido seleccionados para llevar a cabo la experimentación. La precisión de los modelos ha sido comparada con las métricas AUC y kappa de Cohen. La especificidad, la sensibilidad y f1-score muestran el rendimiento en la detención de las clases de ransomware estudiadas.

Acknowledgments

I would like to thank Antonio Tallón for his suggestions, helpful advice, and guidance in this project.

I am also grateful to my family for their unconditional support and love.

I cannot begin to express my thanks to Natalia for being my partner in all aspects of my life.

Table of Contents

List of Tables

List of Figures

List of Appendices

1 Introduction

Cryptocurrencies are getting a prominent place on global monetary transactions (Liu, Jiang, Liu, & Tse, 2021). Cryptocurrencies are digital currencies whose transactions are maintained and verified through a decentralized system with the use of cryptography and not by a centralized system. The bitcoin market has been the top cryptocurrency by market capitalization since the blockchain transaction started. The bitcoin network is a public and distributed ledger where all transactions append between two different addresses and is recorded on bitcoin blockchain built and secured by bitcoin miners, who solve complex cryptographic problems through their own computing power and verify blocks and transactions (Goldsmith, Grauer, & Shmalo, 2020). This process establishes a huge record of data which is quickly expanding and allows the chance to use it to analyse the human behaviour in the bitcoin market, this analysis is often called blockchain analysis (Liu et al., 2021) and is a powerful tool against cybercriminal activity. The European Commission defined cybercrime as: “criminal acts committed using electronic communication networks and information systems or against such networks and systems” (Koops, 2010). Most popular cybercrime involves phishing scams, scamming activity, and hacking of wallets or transactions. In this context, Ransomware is a type of malware which prevents users from accessing their personal online accounts or personal files to require a ransom payment to regain their access. Ransomware can lock access to resources and encrypt their content and is able to infect mobile devices and IoT devices. In most cases, cryptocurrency payments have been selected by ransomware owners as the established payment. Recent computer hacks such as CryptoLocker, CryptoWall, DMA Locker and WannaCry use bitcoin network to ransom (Paquet-Clouston, Haslhofer, & Dupont, 2019a). Transactions stored in the public ledger that Blockchain constitutes do not require the presence of a central authority. The transactions based on blockchain technology have a pseudo-anonymous based on this situation, and criminals can take advantage of it to extort money from their victims without leaving any trace. But the collection of huge data sets from cryptocurrency transactions can become the first step to find the trace of ransomware activity.

Knowledge Discovery in Databases (KDD) is a process which consists of analysing useful information from relevant and important sources with any pre-processing or transformation of

the database to extract knowledge according to the specification of measures and thresholds (Montalvo-Garcia, Quintero, & Manrique-Losada, 2020). This process starts with the creation of a target data set, or making data samples or subset of variables, where discovery will be done. After that, target data could need being cleaned or pre-processed to obtain consistent data. Transformation methods may be used to reduce the dimensionality of data. There is an important phase during KDD process which consists of the searching for patterns of interest, this stage is called Data Mining (Fayyad, Piatetsky-Shapiro, & Smyth, 1996). Finally, the patterns are evaluated and interpretated in order to reach conclusions of the extraction of knowledge. Machine learning (ML) (Sarker, 2021) is becoming the optimal technology to analyse patterns on the data from the current age of the Fourth Industrial Revolution (Schwab, 2017) (4IR or Industry 4.0). The knowledge of artificial intelligence has achieved the development of data analysis techniques that automates the making of analytics models to reach better results. Machine learning techniques can be classified into two different categories, supervised and unsupervised learning models. On the one hand, supervised models (Sathya & Abraham, 2013) are based on evaluating a training dataset from data with an assigned correct classification. On the other hand, unsupervised models (Sathya & Abraham, 2013) are those where the learning is used to find hidden patterns in non-classified input data. The unsupervised context means that the algorithm may organize information and learn to achieve the potential solution.

This Master's dissertation aims to develop a KDD process through the use of a high volume dataset from bitcoin transactions (Akcora, Cuneyt G., Li, Gel, & Kantarcioglu, 2020) applying Data Mining techniques and making data samples, pre-processing data and reducing the dimensionality of data and searching patterns of interest in dataset in order to evaluating results with different metrics through machine learning methods. Specifically, the objective is ransomware (Brewer, 2016) detection through supervised learning classifier algorithms applying feature selection techniques and comparing with the base scenario where no data selection takes place.

2 State-of-the-art

The first study based on the cryptocurrency transaction history was conducted by Reid and Harrigan (F. Reid & M. Harrigan, 2011) in 2011, where the emerging structure from the Bitcoin network was revealed and they demonstrated the forensic capabilities of bitcoin transaction data. Since them, many studies through data mining techniques have been successfully applied with the use of cryptocurrency transactions data. Although it is easier to find more literature in the field of supervised learning, the related work has been applied through unsupervised and supervised learning models. While supervised learning has been used in price prediction, detention of fraud activities or cybercrime and other purposes such as de-anonymizing cryptocurrencies transactions, unsupervised learning studies have been concentrated in the detection of anomalies through the cryptocurrency networks.

Phan and Lee (Pham & Lee, 2016) used three unsupervised learning methods, k-means clustering, Unsupervised Support Vector Machines (modified SVM) and Mahalanobis distance based method through two graph of Bitcoin Network, one graph has used as nodes and in the other the nodes are transactions. Their work focused on the detection of anomalies in Bitcoin transaction network, involving a more general searching than the fraud activities detection. Chen et al. (Chen, Ting et al., 2020) wrote the first systematic investigation on Ethereum transactions using graph analysis, a new way to collect the transaction data is proposed to make money flow graph, smart contract creation graph and smart contract invocation graph to analyse more activities on Ethereum Network and analysing attack forensics and anomaly detection via cross-graph analysis.

In the line of price prediction, Jang and Lee (Jang & Lee, 2017) presented empirical studies able to show the effect of Bayesian networks in predicting Bitcoin price time series, and interpreting the high volatility of Bitcoin price on the date of their paper was written, by relevant features from Blockchain information of Bitcoin network. The results reached are compared with other linear and non-linear benchmark models. Jay et al. (Jay et al., 2020) trained LSTM (Long Short-Term Memory) models and MLP (Multi-Layer Perceptron) models for Bitcoin, Ethereum and Litecoin price prediction. The proposed approach is based on random walk theory from financial market used for modelling stock prices. The results achieved are superior to results from the

deterministic models applied in previous papers. Saad et al. (Saad, Choi, Nyang, Kim, & Mohaisen, 2019) reached an accuracy over to 99% for Bitcoin and Ethereum prediction with their work. Linear regression, Random Forest (RF), and Gradient Boosting (GB) have been used as regression models to predict the prizes. They also applied LSTM networks, based on recurrent neural networks keeping a continuous set of data for a long time, and reaching high accuracy than previous works.

The literature where classification learning is applied can be grouped in two categories, binary classification, and multi-class classification. Ranshous et al. (Ranshous et al., 2017) distinguished from previous works representing the multi-way relation between transactions and addresses in the Bitcoin network as directed hypergraph model. They searched a potential laundering pattern through binary classification models, testing Random Forest, AdaBoost, Linear SVM, Perceptron and Logistic Regression with features from different characteristics of exchange and non-exchange addresses, and reaching the best perform with Random Forest and AdaBoost attaining high accuracy labelling addresses as being owned by exchanges or not. Bartolleti, Pes and Serusi (Bartoletti, Pes, & Serusi, 2018) study Ponzi schemes on the Bitcoin blockchain via imbalanced data set from Bitcoin transactions. They selected 5 different features from 32 features based on Bitcoin addressed characteristic to applied data mining and feature selection techniques with binary classification through RIPPER, Bayes Network and Random Forest classifiers. They consider Random Forest with cost sensitive learning the most effective model for detecting Ponzi schemes on Bitcoin Blockchain, which achieved a low number of false positives. Weber et al. (Weber et al., 2019) contributed to literature with a binary classification task predicting illicit transaction using a time series graph of over 200K Bitcoin transactions, 234K directed payment flows and 166 features. They tested a variation of Logistic Regression, Random Forest, MLP and Graph Convolutional Networks (GCN), and they found special interest on GCN as a new method for capturing relational information. Their results achieved showed the superiority of Random Forest. Chen et al. (Chen, Weili, Zheng, Ngai, Zheng, & Zhou, 2019) also study Ponzi schemes which are hired as smart contracts, they called these Ponzi schemes as *smart Ponzi schemes* and obtained examples by manually checking more than 3000 open source smart contracts on the Ethereum network. Two types of features have been extracted to perform a binary classification model to detect *smart Ponzi schemes.* Their proposed model performs

better results than many traditional classification models, they achieved estimating more than 500 *smart Ponzi schemes* through Ethereum network.

Most of multi-class learnings applied in the previous researches have been focused on solving the issues derived from Blockchain pseudo-anonymity. Jourdan et al. (Jourdan, Blandin, Wynter, & Deshpande, 2018) defined novel features for entity classification from a graph neighbourhood perspective based on Bitcoin exchange transactions modelling the Bitcoin Blockchain such as directed weighted bipartite graph. They analysed the anonymity properties of Bitcoin Networks as a classification problem by a set of categories of Bitcoin users represented by 5 entity categories, *exchange, service, gambling, mining pool* and *DarkNet Marketplace.* Decision Tree (LightGBM) and Logistic Regression performances have been analysed on their work, adding features to the generic model based on their ease of access. They achieved an accuracy of 41% in the identification of entity type, the model choice has not been significance in the classification performance, but the use of more sophisticated features provides a drastic improvement. Toyoda et al. (Toyoda, Ohtsuki, & Mathiopoulos, 2018) proposes the first multi-class identification of Bitcoin addresses to identify among seven classes via supervised learning, *exchange, faucet, gambling, investment scam, marketplace, mining pool* and *mixer.* Extreme Gradient Boosting (XGBoost), Random Forest, SVM and neural network have been trained via 26000 Bitcoin addresses that have been used from January 2009 to February 2017. The best performance achieved 72% of accuracy in the identification. Zolal et al. (Zola, Eguimendia, Bruse, & Urrutia, 2019) present a method to attack Bitcoin anonymity by leveraging a cascading machine learning approach which can work with only few features directly extracted from Bitcoin blockchain data. The data set is composed of 311 different samples, and similar to other works, which is divided into six classes. GB, Adaboost and Random Forest are the models selected on their work to predict each class and improve the results obtained in the previous works. Sun Yin et al. (Sun Yin, Langenheldt, Harlev, Mukkamala, & Vatrapu, 2019) present an approach for de-anonymizing the Bitcoin Blockchain transactions through multi-class supervised learning with 22 different imbalanced uncategorized clusters, 14 of them have been labelled such as ransomware or darknet-market. They utilized a sample of 957 entities with around 385 million transactions. They applied KNN, Decision Trees, Adaboost, GB, Random Forest, Extra trees, and Bagging classifiers. GB with default parameters reached a F1-score of 79.64% which provides a list of suspects who can belong to cybercriminal activities and may be used for further

investigation. Linoy et al. (Linoy, Stakhanova, & Ray, 2021) present an approach which finding affiliation between Ethereum addresses to undermine pseudo-anonymity of Ethereum transactions by grouping addresses were used to deploy smart contracts from the same authors by the use of stylometry techniques. The approach is validated on real-word scammers' data and Ponzi scheme-related contracts using supervised learning techniques.

There are more than 500 known ransomware families, almost all of them demand payments in Bitcoin, and ransomware detection has become a specific field from cybercriminal activities studied in the related works where our research can fit. In this context, Cabaj, Gregorczyk, and Mazurczyk (Cabaj, Gregorczyk, & Mazurczyk, 2018) contributes with a Software-Defined Networking (SDN) based on the observation of network between communication of two families of ransomware, CryptoWall and Locky. The paper defends that HTTP messages' sequences and their respective content sizes is enough to detect both families. The experimentation confirms that SDN analysis is efficient and feasible in the ransomware detection, which can reach a detection rate of 97-98% with 1-2% or 4-5% false positives. WannaCry ransomware uses similar encryption techniques and attack methodologies as other ransomware families. Zimba, Simukonda, and Chishimba (Zimba, Simukonda, & Chishimba, 2017) discovered these characteristics analysing WannaCry ransomware samples based on malware-free infection vectors. The ransomware code is dissected by the perform of reverse-enginnering for further analysis. When a new paper of anti-malware software that improves detection accuracy is published, malware developers usually upgrade their attack methods to reduce the detection rates. Egunjobi et al. (Egunjobi, Parkinson, & Crampton, 2019) present a possibility of increasing the detection rates and classification by the use of static and dynamic features of ransomware through the application of supervised learning models such as instance-based, Random Forest, Naïve Bayes and SVM to classify Locker Ransomware and Crypto Locker Ransomware, and using a data set from a ransomware repository and goodware from Portable Apps. Paquet-Clouston et al. (Paquet-Clouston, Haslhofer, & Dupont, 2019b) propose a data-driven method based on known clustering heuristics for identifying ransomware information from Bitcoin transactions and providing a better picture of global impact of ransomware activities. The method has been applied on a sample of 35 different ransomware families finding new addresses related to each family. The rising of ransom payments over time and the direct effect of each ransomware family have been analysed in this work. Almashhadani et al.

(Almashhadani, Kaiiali, Sezer, & O'Kane, 2019) present a multi-classifier intrusion detection system to detect ransomware network activities in two independent binary classifiers, packet-based classifier, and flow-based classifier. Random Forest, Random Tree, LibSVM and Bayes Net were selected to build each classifier, and the algorithm which provides better accuracy is selected by each classifier autonomously. Locky family is selected as a case studied to analysis the behaviour of crypto ransomware activities. There is a complex problem in ransomware detection related to these addresses that belong to same criminal actors which does not present common patterns associated. Dalal, Wang and Sabharwal (Dalal, Wang, & Sabharwal, 2021) introduce new algorithms for local clustering and supervised graph machine learning which achieve a 85% accuracy to differentiate between random, ransomware and gambling actor in local subgraphs of the known criminal actors.

3 Description of the Data

The studied data set (Akcora, Cuneyt G. et al., 2020) contains information about daily transactions of bitcoin network through 2916697 instances and 10 variables. Table 1 collects the total of missing and different values for each variable. Missing values have not been found in the data set. In this way, it is not necessary to do any additional preprocessing. Two variables of measure of time have been collected in the data set, year, and day. The period of analysis data starts in 2011 and finishes in 2018, variable day has 365 different values. The data set contains a nominal attribute called *Address* with a lot of different values (a ninety per cent of the number of instances in the sample) which has not enabled to achieve any classification model after more than 120 hours in our preliminary experiments. This feature represents the address of transactions, and it has been removed to proceed with the analysis. The rest of variables are numeric variables and have been listed with their statistics characteristic in Table 1.

Table 1. Values of variables in data set.

Variable	Mean	Std. Dev.	Min	Max	Different values	Missing Values
address	-	-	-	-	*2631095*	*0*
year	-	-	-	-	*8*	*0*
day	-	-	-	-	*365*	*0*
length	*45.009*	*58.982*	*0*	*144*	*73*	*0*
weight	*0.546*	*3.674*	*0*	*1943.749*	*785669*	*0*
count	*721.645*	*1689.676*	*1*	*14497*	*11572*	*0*
looped	*238.507*	*966.322*	*0*	*14496*	*10168*	*0*

neighbors	*2.207*	*17.919*	*1*	*12920*	*814*	*0*
income	*4464889007.186*	*162685960669.345*	*30000000*	*49964398238996*	*1866365*	*0*
label	-	-	-	-	*29*	*0*

Length quantifies mixing rounds on Bitcoin which means where transactions are received and distributed with similar amounts of coins in many rounds with new addresses for hiding the origin of coin. *Weigh*t is designed to quantify the merge behaviour, where coins in multiple addresses have been passed through merging transactions and collected in a final address. In the same way, *Count* feature represents information on the number of transactions. The count feature contains information of the number of transactions, and weight feature contains information of the quantity of transactions. *Looped* aims at counting how many transactions move coins on the network by using different paths, split their coins, or merge them in a single address. Finally, *Neighbor* is an integer variable, and it is the number of transactions which have common addresses as one of its output addresses and *Income* represents the total amount of coin output and it has been measured in Satoshi (Nakamoto, 2008) amount (1 bitcoin = 100 million satoshis).

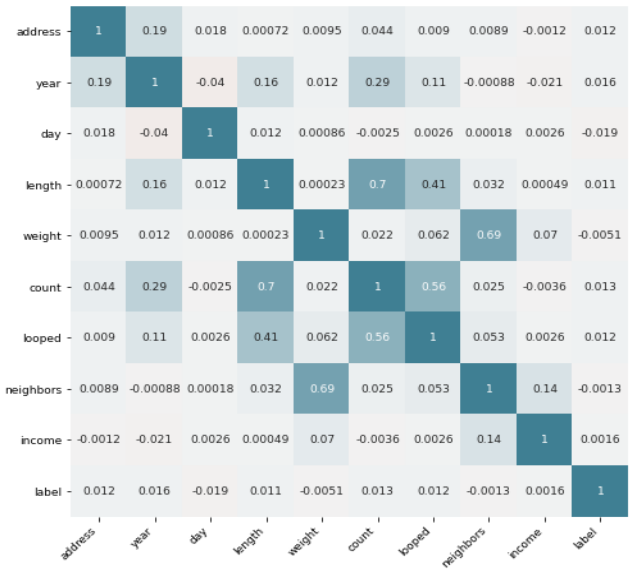


Figure 1. Correlation between variables.

Figure 1 represents the relationship between the variables in the data set through the correlation relations. Each column and row represent a variable. The values inside the matrix are the Pearson correlation coefficient (Wang, 2013) between variables.

$$r_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (1)$$

Where n is the size of the sample; x_i, y_i are the individual sample points and $\bar{x}, \bar{y}$ are the samples means.

The nominal attributes have been converted in categorical variables to process with this representation. The highest values of correlation have been found between length and count, weight and neighbours, and count and looped.

Table 2. Ransomware labels.

ID	*Label*	*Count*
29	white	2875284
5	paduaCryptoWall	12390
3	montrealCryptoLocker	9315
1	princetonCerber	9223
2	princetonLocky	6625
4	montrealCryptXXX	2419
11	montrealNoobCrypt	483
7	montrealDMALockerv3	354
12	montrealDMALocker	251
9	montrealSamSam	62
8	montrealCryptoTorLocker2015	55
25	montrealGlobeImposter	55
19	montrealGlobev3	34
13	montrealGlobe	32
6	montrealWannaCry	28
23	montrealRazy	13
28	montrealAPT	11
15	paduaKeRanger	10
10	montrealFlyper	9
17	montrealXTPLocker	8
16	montrealVenusLocker	7
21	montrealXLockerv5.0	7
24	montrealCryptConsole	7
14	montrealEDA2	6
20	montrealJigSaw	4
18	paduaJigsaw	2
22	montrealXLocker	1
26	montrealSam	1
27	montrealComradeCircle	1

Ransomware labels, which are listed in Table 2, have been obtained using Ransomware addresses taking three different studies: Montreal, Princeton, and Padua (Akcora, Cuneyt Gurcan, Li, Gel, & Kantarcioglu, 2019) through clustering process which analysed Ransomware behaviour similarity. The white label represents transactions that are free of ransom and stages the 98.58% of analysed transactions. It is relevant to stress that we are tackling with a multi-class classification problem with more than 25 classes and with an unbalanced data set. There are many labels with minuscule counts, and this situation represents an issue in a multi-class classification problem, there is not enough information which make possible a good performance in the classification of this labels. The data set has been subsampled to test our experiments using the top 6 labels listed in Table 2 to improve the detection of most common ransomware labelled, hence, the number of instances to conduct the experiments is 2915256.

4 Experimentation

4.1 Methodology

The knowledge discovery process in the studied data set has been tedious because of the high volume of data and complexity and nature of data. The procedure of building classifier models for imbalanced data set such as our problem is a demanding task. Classic classifier models often cannot achieve good performance with high imbalance ratio in data set. Even if a high level of accuracy has been reached, this level can only represent the results achieved of classification on majority class, but it can be close to zero of the rest whole data set. This situation becomes more difficult when we are finding solution to an imbalanced multi-class problem. On one hand, resampling techniques can be applied in order to solve this issue and there are a huge record of different techniques, such SMOTE (Chawla, Bowyer, Hall, & Kegelmeyer, 2002), a method which creates artificial samples belonging to the minority class, where new samples are called synthetic samples. Another method, ADASYN (He, Bai, Garcia, & Li, 2008) creates new instances depending on the relative weight of minority instances defined by proportion of majority class samples in the neighbourhood to determine the number of synthetic examples that will be created. Also, under-sampling approaches (Yen & Lee, 2006) can be applied to improve the classification accuracy for minority class reducing the number of majority class instances on training data. On the other hand, cost sensitive learning (Elkan, 2001) assigns a high cost to non-true positive on classification of the minority class and reduces the overall cost through the development of a Cost Matrix (Ling & Sheng, 2008). Oversampling such as SMOTE or ADASYN makes more copies of existing instances which can have same effect than overfitting on learning process. Also, the new copies make the problem bigger and increasing the learning time. Undersampling approaches do not use potentially useful data and contributing to the loss of information on learning process. These techniques could not consider neighbour examples from other classes, and this can introduce additional noise to training data. The problem with cost sensitive learning is that there are not many implementations of all learning algorithms and could be hard finding a good Cost Matrix. Considering these disadvantages and due to the size of data set used oversampling and undersampling techniques have not been applied in this work. Instead of these techniques, *k*NN have been tested without any strategy mentioned, because *k*NN can be independent of imbalance data issues; Decision Tree, Random Forest have been tested using

different weights in all the classes misclassifying the white class to obtain better results on the rest of classes; and XGBoost have been tested by the optimization of *max_delta_step* parameter.

The experimentation has been conducted by the equipment with the following settings: Windows 10 as operative system (OS), CPU AMD Ryzen 4800 h 2.9 Ghz, 16 Gb of RAM memory and GPU RTX Nvidia 2060. Concerning the specific software for data two different tools have been used: the Weka (Waikato Environment for Knowledge Analysis) framework (Hall, Mark et al., 2009) and Python language (VanderPlas, 2016) with the libraries Pandas and Numpy (McKinney, 2011), Scikit Learn (Pedregosa et al., 2011), and XGBoost implementation through Jupyter Notebook (Perkel, 2018) from Anaconda distribution platform. The data set has been divided into two subsets with a stratified hold-out cross validation (Ojala & Garriga, 2010) to ensure that samples on training and test sets have been organized in same proportion as in original data set. One quarter of instances (728814 instances) have been collected in testing sets, and the rest have been selected as training set (2186442 instances). All classifiers have been applied using the same training and testing sets. The files are available in github[1] in order to make possible the reproduction of experiments by any interested person.

4.2 Feature selection

Feature Selection (FS) (Cai, Luo, Wang, & Yang, 2018) is a process to reduce the dimension of data set and makes more effectively and faster the learning of ML models, it consists in identifying the irrelevant and redundant features and remove as much of these feature as possible. Feature subset selection is the most common strategy within the FS area. In general, FS algorithms have two parts: an algorithm is used to generate proposed subsets of features selection and to find an optimal subset, and other part that evaluates the results of proposed feature subset and returning measure of goodness to the selection algorithm. FS have normally been grouped into two groups, filter, and wrapper. The difference depends on how the FS method works against inductive algorithm that will apply the selected subset. This paper focuses on the use of filter methods, but many of them have been applied. ReliefF is a method that selects the relevant features using a randomly picks of a sample of instances and assigns a weight for each feature

[1] https://github.com/josanb12/KDD-Bitcoin

based on the Euclidean distance measure. The features are chosen after exhausting all instances in the sample using the order assigns of the weight feature. Neural-network feature selector (NNFS), Boosted Decision Stump FS (BDSFS), LVF, LVS, Sequential forward floating selection (SFFS) and sequential backward floating selection (SBFS) are other methods. However, methods based on a correlation measure (Hall, 1999) have been applied in this work. The feature subset is selected following a hypothesis that suggests that "a good feature subset is one that collects features which are highly correlated with the independent variable but are uncorrelated with the rest of variables". Equation 2 (Kavipriya & Karthikeyan, 2017) is used to filter out the extremely correlated and unrelated features.

$$Fs = \frac{N\overline{r_{ci}}}{N + N(N-1)r_{ii}} \tag{2}$$

Where N is the number of features in data set, r_{ci} is the average feature correlation with the class and r_{ii} is the mean feature inter-correlation. The advantages (Kavipriya & Karthikeyan, 2017) of this method are that it needs less computational complexity compared to other method, and it scale well to high dimensional data set. However, this method depends on the model and can fail to fit well the data, but it does not ignore the communication with classifier such as other methods. This works is an extension of the results obtained on (Blanco & Tallón-Ballesteros, 2021) using similar FS techniques.

Extended Correlation-based Feature Selection (eCFS) (Tallón-Ballesteros, Cavique, & Fong, 2019) and Correlation-based Feature Selection (XCFS) with Exhaustive Search (Mnich & Rudnicki, 2020) have been considered as feature subset selection methods. They have been applied using the Weka (Waikato Environment for Knowledge Analysis) framework (Hall, Mark et al., 2009).

Table 3 summarizes the settings used in both methods and the feature selection obtained. The experiments have been conducted through three different cases, no feature selection applied called Base, eCFS where eFSS method applied, and XCFS where XFSS method applied.

Table 3. Feature selection procedures used in the experimentation.

Feature selection method	Type	Parameter/Property	Value	Features selected
eCFS (Tallón-Ballesteros et al., 2019)	*eFSS*	*Attribute evaluation measure* *Search method* *Consecutive expanded nodes without improving* *Search direction*	*Correlation* *Best First* *5* *Forward*	*year, day, neighbor, income, weight, count, looped*
XCFS (Mnich & Rudnicki, 2020)	*XFSS*	*Attribute evaluation measure* *Search method* *Consecutive expanded nodes without improving* *Search direction*	*Correlation* *Exhaustive search* *5* *Forward*	*year, day, neighbor, income*

4.3 Metrics

The overall classification accuracy may not be an appropriate metric in imbalanced data problems. A common classifier can predict every case as the majority class and still achieve very high level on accuracy metric. It is necessary to calculate alternative metrics such Cohen's Kappa (Cohen, 1960) and ROC AUC score (Fawcett, 2006). In multi-class classification with learning extremely imbalanced data, the study of overall metrics is not enough, the difference among metric of each class is much relevant than binary classifications. For this purpose, precision, recall, and f1 measures of each class have been collected from each model applied. These metrics depend on confusion matrix results. Confusion matrix (Provost & Kohavi, 1998) contains information between actual and predicted classifications obtained by a classification algorithm. The size of a confusion matrix is (N, N) where N is the number of different classes in the problem. Figure 2 represents a confusion Matrix in multi-class case considering the class k ($0 \leq k \leq n$), where TN is the number of true negatives, TP is the number of true positives, and FN and FP are the number of false negatives and false positives respectively.

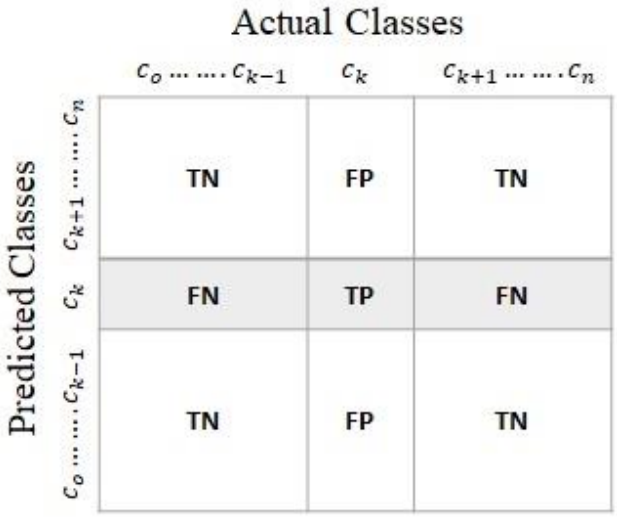


Figure 2. Confusion Matrix on multi-class case**. Adapted from:** (Flach, 2012).

Precision is the capability of the classifier to not classify a negative sample as positive.

$$Precision_k = \frac{TP_k}{TP_k + FN_k} \tag{3}$$

Recall is the ability of the classifier to classify properly all the positive samples.

$$Recall_k = \frac{TP_k}{TP_k + FN_k} \tag{4}$$

F-1 score is the harmonic mean of precision and recall and is more sensitive to data distribution than other metrics, this made it more suitable measure for classification problems on imbalanced data set.

$$F1_k = \frac{2 \times Precision_k \times Recall_k}{Precision_k + Recall_k} \tag{5}$$

Cohen's Kappa (Cohen, 1960) is a statistic that scores the level of agreement in a classification problem.

$$k = \frac{p_0 - p_e}{1 - p_e} \tag{6}$$

Where p_0 is the empirical proportional agreement between predicted values and actual values. This proportion can be calculated with the sum of the diagonal values of any confusion matrix divided by the sum of the rest of values of the confusion matrix. In addition, p_e is the probability that true values and false values agree randomly. This metric has been applied using the default options implemented in scikit learn library on python.

Receiver operating characteristic (ROC) (Fawcett, 2006) curve is a two-dimensional representation between true positive rate and false positive rate and may be used to evaluate a classifier performance, figure 3 represents an example of a ROC figure. To compare classifiers, it is possible to reduce ROC performance to a single value which can represent the performance obtained. AUC (Fawcett, 2006) score is the calculation of area under the ROC curve and its value is always between 0 and 1. However, a classifier should not have an AUC less than 0.5. AUC of a classifier represents the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance (Fawcett, 2006). Equation 7 and 8 clarified the concept of True Positive Rate and False Positive Rate represented on Figure 3.

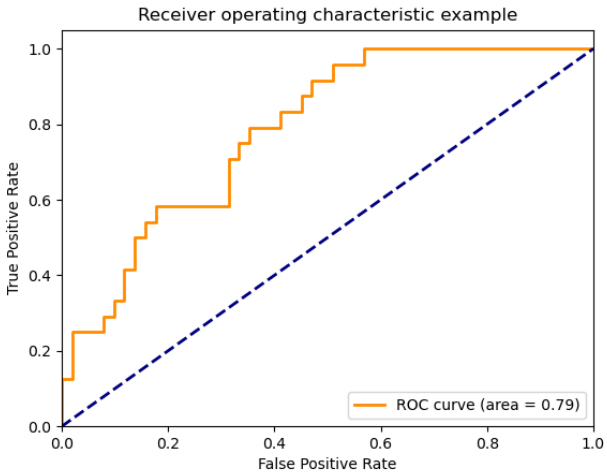


Figure 3. Example of ROC representation. Source: https://scikit-learn.org

$$True\ Negative\ Rate(TNR) = \frac{TN}{TN + FP} \quad (7)$$

$$True\ Positive\ Rate(TPR) = \frac{TP}{TP + FN} \quad (8)$$

The AUC score has been obtained by Scikit Learn library on Python, which is calculated through trapezoidal Rule Numerical Integration to find the area under ROC curve (Yeh, 2002). The area of trapezoid is:

$$(x_{i+1} - x_i) (y_i + y_{i+1}) / 2 \quad (9)$$

In the AUC case the formulation based in the last equation is:

$$(FPR_{i+1} - FPRi) (TPR_i + TPR_{i+1}) / 2 \quad (10)$$

Where FPR represents false positive rate calculated by:

$$FPR = 1 - TNR \quad (11)$$

Two strategies are currently supported to used AUC score in multi-class classification, one-vs-one algorithm calculated by using the average of the pairwise ROC AUC scores between each label, and the one-vs-rest algorithm which computes the average of ROC AUC scores for each label against the rest. In this paper, a one-vs-rest strategies have been applied. The rest of parameters have been maintenance in the default option using scikit learn on python.

4.4 Supervised Machine Learning.

The problem of ransomware detection using the data set selected have been conducted via 4 non-parametric models. CART, kNN, Random Forest and XGBoost have been the machine learning models selected based on previous related works studied in the same field.

4.4.1 K-Nearest Neighbors (*k*NN).

K-nearest Neighbour (*k*NN) (Wu et al., 2008) classification is an *instance-based learning* or *non-generalizing learning* method used to find a group of k objects during the learning from the training set which are closest to the test object and assign a label on the predominance class in this neighbourhood. A set of labels of objects, a set of stored records, a distance or similarity metric to calculate distance between object and the value of *k*, which represents the number of nearest neighbours, are the key elements of this approach. The *k*-nearest neighbours are identified, and the class label assigns of these neighbours are used to select the class label of the test object. The algorithm computes the distance or similarity between test objects and all the training objects to determine the nearest-neighbour list, them the test object is classified based on the majority class of its nearest neighbours:

$$\text{Majority Voting} = \text{argmax}_v \sum_{(x_i, y_i) \in D_z} I(v = y_i) \qquad (12)$$

Where *v* is a class label, y_i is the class label for the *i*th nearest neighbours, and *I* is a function that returns the value 1 if its argument is true and 0 in other case. *D* is the set of *k* training object and z is the test object.D_z is the nearest-neighbours list.

*K*NN is a lazy learning classifier (Flach, 2012) which requires computing the distance of the unlabelled object to all the objects in the labelled and can need a lot of computational time for large training sets but is possible to reach an efficient computational time using different techniques developed for that propose. KneighborsClassifier from scikit-learn has been applied which implements learning based on the number of neighbours within a floating-point value specified of each training point.

4.4.2 Decision Tree (CART).

The name of Decision Tree models is derived from a hierarchical model visually formed like a tree. This algorithm splits observations into multiple subsets, based on a decision node with a criteria determined from the features, and applying the most significant feature to perform the better split among the observations (Baesens, 2014). To perform the better split, the algorithm searches the most significant feature to be applied. The best split is choice computing an

impurity function or loss function based on three possible criteria, Misclassification, Entropy or Gini Index (Raileanu & Stoffel, 2004). Entropy shows the grade of homogeneity of a sample distribution, it means that in the scenario where classes are equally divided in two subsets, the entropy will be 1. When distribution becoming homogeneous, the entropy will be zero. G is the information gain and E the entropy in equation 13.

$$G(y, x) = E(y) - E(y, x) \tag{13}$$

Where E(y) represents the entropy in the class distribution before the split and E(y,x) the entropy after the subset has been split by the decision node. Target variable is *y* and *x* is the feature to be split on. Equation 14 lists the individual Entropy:

$$E = \sum_{I=1}^{C} -f_i log(f_i) \tag{14}$$

Where f_i is the frequency of class *i* at a node and *C* is the number of unique class in the sample. To estimate the information gain, the total Entropy for the split is calculated. Equation 15 shows Gini Index (Raileanu & Stoffel, 2004) function which is another way to compute the impurity function.

$$G = \sum_{I=1}^{C} f_i(1 - f_i) \tag{15}$$

Where G is the Gini Index and f_i is the frequency associated to class *i*, derived from the class distribution in the subset. The algorithm seeks for the feature split with the leads which perform the best split of classes by the lowest Gini Index. Decision tree algorithms have different versions such as ID3, C4.5, C.50 and CART. Classification and Regression Trees (CART) (Loh, 2008) have been applied using an optimized version available on scikit-learn library.

4.4.3 Random Forest.

Random Forest (Breiman, 2001) consists in a combination (ensemble) of individual decision trees training with a different random sample from data, generated by bootstrapping. Each tree is trained via different data, and observations are distributed by nodes which generate the tree

structure to reach a final node. The prediction of a new observation is obtained adding predictions from individual decision trees which are pooled to make the final prediction. Random Forest is referred as Ensemble technique because it uses a collection of results to achieve a final decision. Ensemble methods combinate multiple models to achieve better prediction than the individual original models. Two types of ensembles are the most popular: Bagging and Boosting. Random Forest constitutes a bagging technique (bootstrap aggregation) which aims to reduce the variance and improving the accuracy of predictions. Bagging on Random Forest can be explained by three steps. First, *B* training sets are generated via bootstrapping from original data set. Them, each *B* is trained by a tree without pruning. Finally, per each new observation, prediction of each *B* tree is obtained. The final prediction is the average of *B* prediction. Random Forest classifier has been tested via RandomForestClassifier from scikit learn library.

4.4.4 Gradient Boosting (XGBoost).

Gradient Boosting (Friedman, 2002) is a powerful algorithm in machine learning. It can be applied in regression and classification problems and is a generalization of AdaBoost algorithm. Gradient Boosting is an ensemble method which consists in learning models sequentially, each model fixes residual errors from previous models. First *weak learner* f_1 predicts output variable *y*, and residual error $y - f_1(x)$ is calculated. A new model f_2 aims at predicting residual error from f_1 , and consecutively, this process is repeated *M* times, and each new model minimizes residual error from the previous model. This process is sensitive to overfitting, *learning rate* (λ) parameter can be applied to solve this problem. However, more models are needed to constitute the ensemble, but better results can be reached. Next formulation shows the process explained.

$$f_1(x) \approx y \tag{16}$$

$$f_2(x) \approx y - \lambda f_1(x) \tag{17}$$

$$f_3(x) \approx y - \lambda f_1(x) - \lambda f_2(x) \tag{18}$$

$$y \approx \lambda f_1(x) + \lambda f_2(x) + \lambda f_3(x) + \cdots + \lambda f_m(x) \tag{19}$$

XGBoost (Chen, Tianqi & Guestrin, 2016) is an implementation of machine learning algorithms under the Gradient Boosting framework. XGBoost uses a parallel tree boosting that makes

possible solving many data science problems with less time and better accuracy, because of its scalability on all scenarios it runs more than ten times faster than other algorithms (Chen, Tianqi & Guestrin, 2016). In this paper XGBoost has been applied via XGBoost specific library (Brownlee, 2016) implemented on python based on scikit learn library.

4.4.5 Hyperparameters optimization.

The parameters of each estimator used during experimentation have been optimised by cross-validated grid-search over a specify parameter grid. Cross validation (Flach, 2012) is a resampling process to obtain different portions of data set to testing the model training on different iterations. In our case, 5 partitions have been used on optimization of parameters for each model. Grid Search (Liashchynskyi & Liashchynskyi, 2019) is a traditional method of hyperparameters searching over a given subset of the hyperparameters grid from training algorithms. Because of some values of machine learning algorithm parameter space can be real or unlimited values, it is necessary to specify a parameter grid to apply this method. In this paper grid search provides by GridSearchCV based on out-of-bag score from scikit-learn have been applied using only the training subset. This method generates candidates from a grid of parameter which its values of parameters have been specified with the param_grid parameter. Table 4 lists results obtained via GridSearchCV method. Parameters which are not list on table 4 have been used by default option available on scikit-learn and random_state parameter which control the randomize on learning have been selected as 123 in each model except Random Forest. Random Forest is a stochastic model and all results showed are the average between results obtained from 30 different seeds to counter random effects on results, appendice X shows each result achieved per each seed. XGBoost can behave as stochastic model also, however it acted as deterministic because of subsample parameter have been selected as 1, based on GridSearchCV results obtained. To experiment against imbalanced on class distribution, *class_weight* parameter as *balanced* has been applied on CART and Random Forest. No other techniques have been applied on kNN model, to leave one without any strategy applied to compare results of strategies adopted. In XGBoost model, max_delta_step parameter setup might help when classes are extremely imbalanced such as our problem, and no other techniques have been necessary to obtain good performance with this model. Equation 19 shows how works *class_weight* parameter (King & Zeng, 2001).

$$w_j = \frac{n}{kn_j} \tag{20}$$

Where w_j is the weight to class j, n is the number of observations, n_j is the number of observations in class j, and k is the total number of classes.

Table 4. The results of GridSearchCV.

Models	Parameters	Grid of values	Optimized value
***k*NN**	*n_neighbors*	*3, 9,15,1,25*	*15*
	weights	*uniform,distance*	*distance*
	algorithm	*auto,kd_tree*	*auto*
	leaf_size	*15,30,45*	*30*
	p	*1,2*	*2*
	n_jobs	*None, -1*	*-1*
CART	*criterion*	*gini,entropy*	*gini*
	splitter	*best,random*	*best*
	max_features	*auto, sqrt, log2*	*log2*
	max_depth	*1,5,10,50,100*	*50*
	max_leaf_nodes	*1,5,10,50,100*	*100*
	ccp_alpha	*0.1,0.001,0.00*	*0.00*
	n_jobs	*None, -1*	*-1*
	class_weight	*balanced*	*balanced*
Random Forest	*criterion*	*gini, entropy*	*gini*
	max_depth	*None,3,10,50*	*None*
	max_features	*auto, sqrt, log2*	*auto*
	n_estimator	*25,50, 150,300*	*150*
	n_jobs	*None, -1*	*-1*
	class_weight	*balanced*	*balanced*
XGBoost	*n_estimator*	*25,50,150,300*	*150*
	booster	*gbtree,gblinear,dart*	*gbtree*
	subsample	*0.5,0.8,1*	*1*
	colsample_bytree	*0.5,0.8,1*	*1*
	objective	*multi:softmax*	*multi:softmax*
	gamma	*0.5,1,2,5*	*1*
	n_jobs	*None, -1*	*-1*
	eta	*0.1,0.3,0.5,1*	*0.3*
	max_delta_step	*0,1*	*1*

5 Results

Table 5 lists the results of performance measures in the application of the experimentation described in three different scenarios, Base case, where feature selection has not been applied, eCFS which constitutes the extended correlation-based case and the case of Exhaustive search of CFS. Accuracy, AUC score, Cohen's Kappa have been collected to compare all results achieved, also, overall precision, overall Recall and overall F1-score have been listed for the same purpose. All models selected in each scenario presents similar behaviour in accuracy metric, standing up from 98%, and it is explained by high accuracy reached of majority class labelled as white. This situation means that accuracy could not be a good performance to compare the results, and it is better to focus on AUC score and Cohen´s Kappa which presents different results in each case.

Table 5. Test results obtained with selected models.

ML method	Feature set	Accuracy	AUC	Cohen's Kappa	Precision	Recall	F1-score	Elapsed time
kNN	Base	0.9872	0.7686	0.3251	0.6162	0.4317	0.4811	00:15.32
	eCFS	0.9876	0.7729	0.3844	0.6300	0.4833	0.5261	00:13.33
	XCFS	0.9880	0.7841	0.4548	0.6447	0.5608	0.5836	00:07.95
CART	Base	0.9835	0.7173	0.3766	0.5490	0.5405	0.5445	00:08.68
	eCFS	0.9827	0.7157	0.3626	0.5354	0.5381	0.5367	00:07.53
	XCFS	0.9808	0.7822	0.4096	0.5425	0.6482	0.5864	00:06.26
Random Forest*	Base	0.9887	0.9493	0.4145	0.7399	0.4937	0.5650	01:17.20
	eCFS	0.9885	0.9430	0.4358	0.7128	0.5194	0.5816	01:08.41
	XCFS	0.9844	0.8868	0.4513	0.5912	0.6366	0.6106	01:01.75
XGBoost	Base	0.9901	0.9874	0.4877	0.8391	0.5525	0.6064	08:28.37
	eCFS	0.9900	0.9868	0.4847	0.8337	0.5510	0.6033	08:21.88
	XCFS	0.9897	0.9840	0.4613	0.8282	0.5348	0.5839	07:46.20

Note: * Random Forest results listed are the average results from results of 30 different seed. Desviation has not been included because of the minimal variability among seeds.

CART and kNN have been achieved equivalent level of AUC. However, feature selection has been increased the AUC score in higher level on CART model, especially in the XCFS scenario, where an exhaustive search has been applied. Random Forest and XGBoost have been presented better results than the previous models but feature selection scenarios have not reached better results comparing with base case on these models. Anyway, XGBoost which is the method that have been achieved the highest result, does not present a big variability between the three scenarios of experimentation applied. This situation is different for Random Forest, which presents a lower level on AUC measure in XCFS feature selection case.

XGboost has the same behaviour in the case of Kappa performance, being again the model that achieves the best results, it does not present high variability of results between the three scenarios, although XCFS is the one that achieves the worst results. The two scenarios of feature selection seem to perform better in CART and kNN techniques, where XCFS improves the result obtained, bringing results closer to Random Forest and XGBoost metrics achieved, standing up from 0.40 in both cases. Same conduct can be found on Random Forest model, where 0.45 have been reached in XCFS scenario.

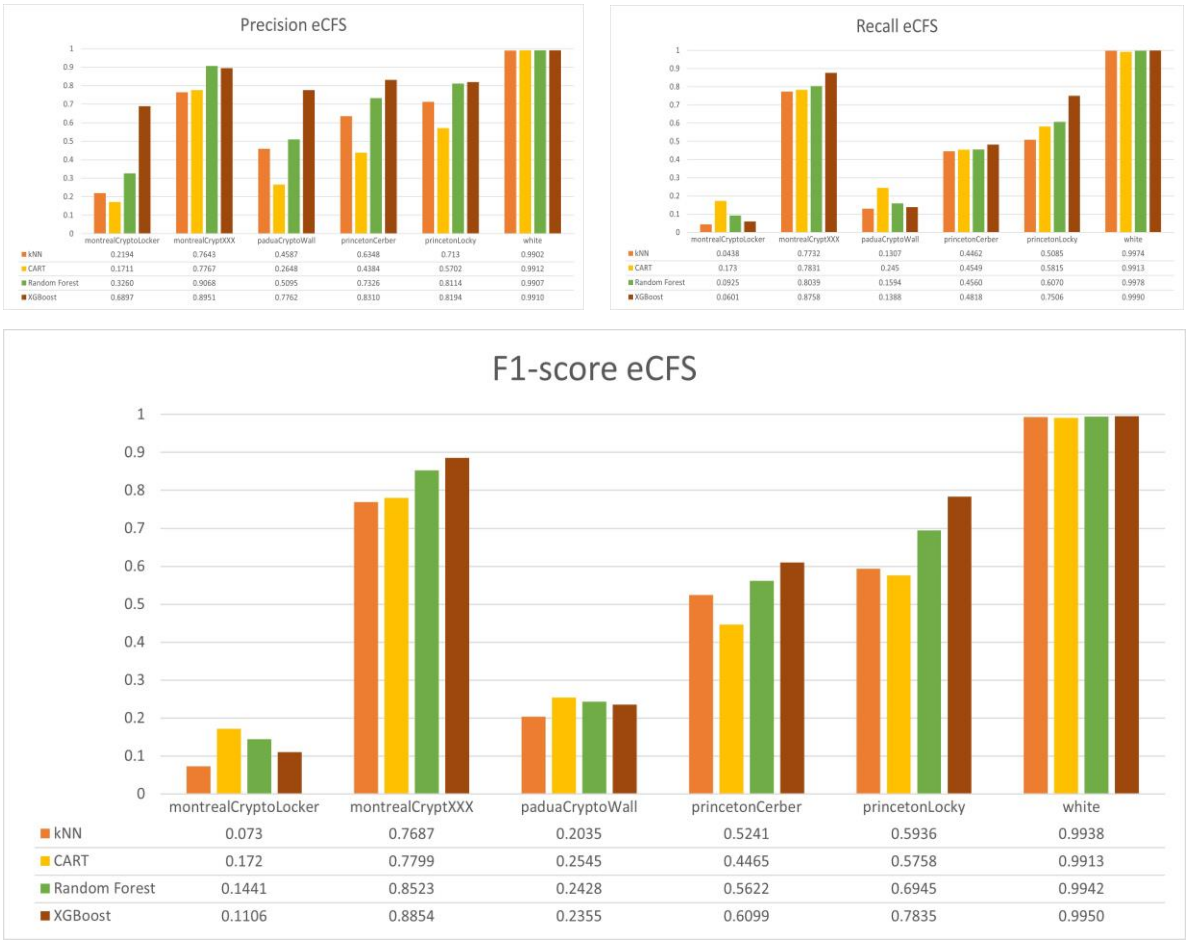
Focusing on precision, recall and F1-score overall metrics, every model in each scenarios seems to perform same pattern on these metrics, a good level on precision, and lower level on recall a therefore on F1-score. CART has been the model with lower performance, follow by kNN. XGBoost has been again the best method and reached the highest perform, but Random Forest achieves the best results on F1-score and recall in XCFS scenario.

Figure 4. Precision, Recall and F1-score on base set.

Note: * Random Forest results listed are the average results from results of 30 different seed. Desviation has not been included because of the minimal variability among seeds.

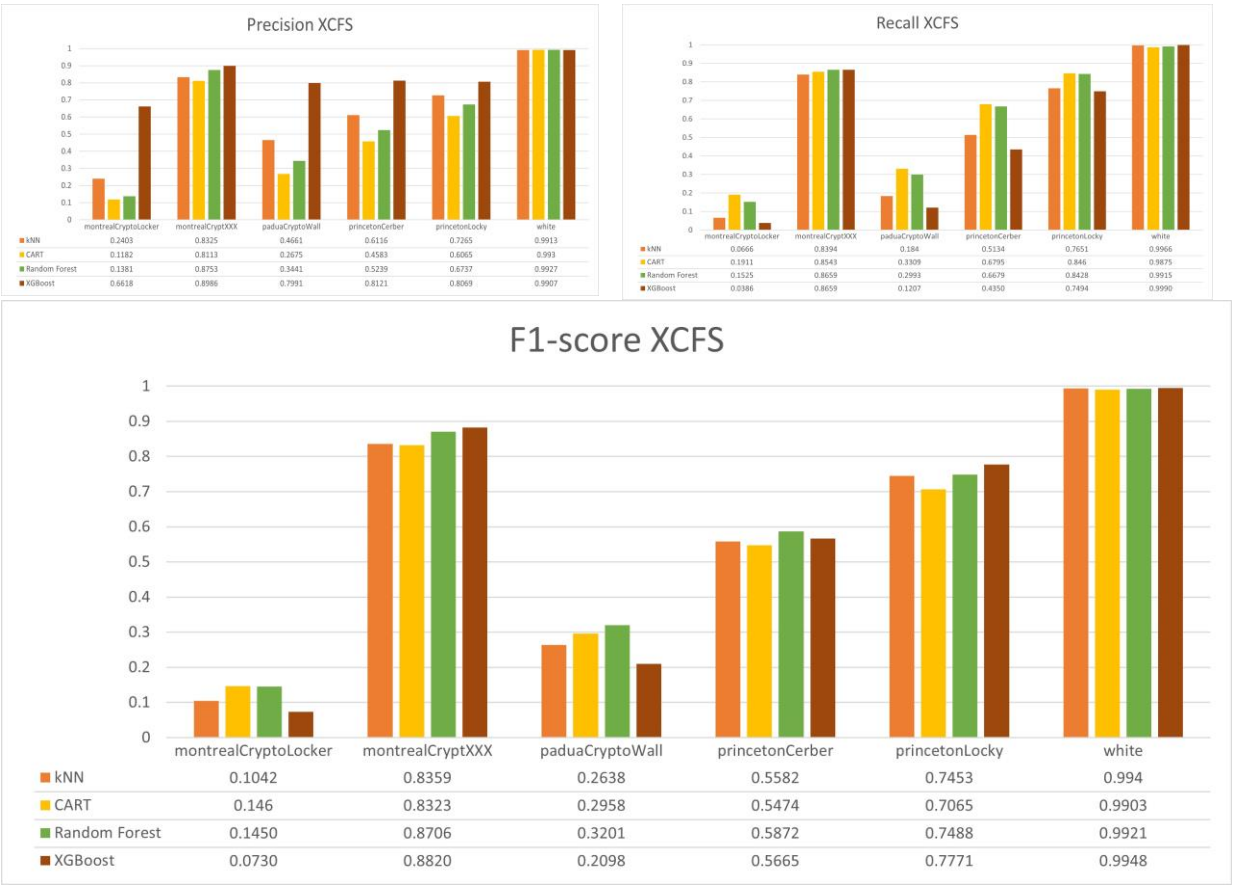
Figures 4, 5, and 6 includes results reached about precision, recall and F1-score of each ransomware label in the three scenarios analysed. As these figures show, the pattern observed in the performance of overall metrics is due to the variability between the false positives and false negatives belonging to the different classes of ransomare analysed, achieving worse results in the cases of *montrealCryptoLocker* and *paduaCryptoWall*, and reaching the best results of precision, recall and f1-score for the *montrealCryptXXX* class, and less in *Princeton* Family. Figure 4 shows the results from the base case, and it is easy to see how XGBoost results obtained the best results of Precision in all classes. However, CART achieved better performance with *montrealCryptoLocker* and *paduaCryptoWall* of Recall metric and consequently on F1-score. Random Forest model also obtained better Recall and F1-score with these classes but in smaller measure than CART.

Figure 5. Precision, Recall and F1-score on eCFS feature subset.



Note: * Random Forest results listed are the average results from results of 30 different seed. Desviation has not been included because of the minimal variability among seeds.

Figure 6. Precision, Recall and F1-score on XCFS feature subset.



Note: * Random Forest results listed are the average results from results of 30 different seed. Desviation has not been included because of the minimal variability among seeds.

Figure 5 represents the results obtained in eCFS feature subset, where the same different in models metrics can be found with less variability between them. Also, Random Forest reached better Recall and F1-score with *montrealCryptoLocker* and *paduaCryptoWall* classes than CART, and the feature selection based on extended correlation improved the learning of Random Forest technique in higher way than the rest of models. The variability of results has been further reduced with the XCFS feature selection application as figure 6 shows. In all scenarios, montrealCryptXXX is the easiest ransomware label to be detected excluding white label follow by both labels from *princeton* family. To extend this analysis, appendix A collects normalized confusion matrices of each scenario studied, where the explanation of the differences obtained in the classification of labels can be found. As confusion matrices shows, the false negatives of ransomware labels have been concentrated as white. However, no false negatives have been founded between ransomware labels, and white label has not been classified as ransomware label

in any case. The recall metrics reached are explained by this fact because of the high true negative rate obtained. Also, both feature selection methods applied have been improved the false negative and true positive rate in each case.

Feature selection provides a faster and more optimal process in a huge data set such the used on this work, using less information represented by less features selected did not generate a remarkable variability on results obtained, but needs less time to reach equal results. Elapsed time lists on table 5 consists in the duration of training phase and is measure in minutes. XCFS improves a lot the results achieved, and only in the XGBoost method, the results are lower than base scenario, although with a minimal variability. This situation allows to affirm that in our case, XCFS is a valid strategy which enables to achieve similar or better results with less features compared to the baseline data set.

6 Conclusions

Bitcoin Network constitutes a public ledger without the presence of a central authority and involves a pseudo-anonymous around the blockchain transactions which allows to criminals extorting money from their victims. Ransomware malware could be one of the most dangerous cybercriminal activities on this scenario.

The technology of blockchain used in bitcoin network makes possible to organise massive data such the collected on the data set used in our analysis, that makes possible the implementation of KDD methodology to extract information from transactions of cybercrime and illegal activities such as ransomware payments. However, implementing machine learning techniques with the dimension of the studied data set can lead to high computation times and make difficult the knowledge extraction from data. In this context, feature selection enables to simplify classification process using less information represented by less features. Also, the recorder of information described may constitutes an imbalanced of classes in data set such as used on this work, and only few instances have been recorded of a lot of labels included on our data set, implying the necessity to focus on most common ransomware labels added, analysing only the 6 top labels including the free ransomware class categorized as *white* in a subsampling of the total of data. A few methods to solve this problem have been described on this paper such as cost sensitive learning and SMOTE. Anyway, we focused on the use of machine learning techniques applied without any imbalanced strategy such as kNN and XGBoost, and the use of weights per each class in CART and Random Forest. The results confirms that ensemble methods are much better to classify the instances testing, and XGBoost is the one which has been achieved the best results in all performances, by fixing the parameter *max_delta_step*.

The methodology proposed on this paper aims to detect ransomware from three different families, *Monreal, Padua and Princeton*, and we discovered that the performance achieved of each family have been different in the transaction's classification from ransomware payments. *MonrealCryptoLocker* and *paduaCryptoWall* have been the most difficult to detect in our case, and *montrealCryptXXX* the easiest one. This situation implies the necessity of more information and data from the rest of labelled categorized and ransomware developers are not blind, as new techniques and methods come out to detect ransomware cybercriminal activities, they improve

their development, creating a tug of war that makes the balance fall on their side. It is very important to continue making progress in stopping ransomware on cryptocurrency networks to achieve that blockchain transactions are not the epicenter of current cybercrime, due to the great advantages that offer to development of these activities.

References

Akcora, C. G., Li, Y., Gel, Y. R., & Kantarcioglu, M. (2020). Bitcoinheist: Topological data analysis for ransomware prediction on the bitcoin blockchain. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence,*

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.

Baesens, B. (2014). *Analytics in a big data world: The essential guide to data science and its applications* John Wiley & Sons.

Bartoletti, M., Pes, B., & Serusi, S. (2018). Data mining for detecting bitcoin ponzi schemes. Paper presented at the *2018 Crypto Valley Conference on Blockchain Technology (CVCBT),* 75-84.

Blanco, J. A., & Tallón-Ballesteros, A. J. (2021). Supervised machine learning techniques in the bitcoin transactions. A case of ransomware classification. Paper presented at the *International Workshop on Soft Computing Models in Industrial and Environmental Applications,* 803-810.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5-32.

Brewer, R. (2016). Ransomware attacks: Detection, prevention and cure. *Network Security, 2016*(9), 5-9.

Brownlee, J. (2016). *XGBoost with python: Gradient boosted trees with XGBoost and scikit-learn* Machine Learning Mastery.

Cabaj, K., Gregorczyk, M., & Mazurczyk, W. (2018). Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics. *Computers & Electrical Engineering, 66*, 353-368.

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing, 300*, 70-79.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321-357.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. Paper presented at the *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 785-794.

Chen, T., Li, Z., Zhu, Y., Chen, J., Luo, X., Lui, J. C., . . . Zhang, X. (2020). Understanding ethereum via graph analysis. *ACM Transactions on Internet Technology (TOIT), 20*(2), 1-32.

Chen, W., Zheng, Z., Ngai, E. C., Zheng, P., & Zhou, Y. (2019). Exploiting blockchain data to detect smart ponzi schemes on ethereum. *IEEE Access, 7*, 37575-37586.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20*(1), 37-46.

Dalal, S., Wang, Z., & Sabharwal, S. (2021). Identifying ransomware actors in the bitcoin network. *arXiv Preprint arXiv:2108.13807,*

Egunjobi, S., Parkinson, S., & Crampton, A. (2019). Classifying ransomware using machine learning algorithms. Paper presented at the *International Conference on Intelligent Data Engineering and Automated Learning,* 45-52.

Elkan, C. (2001). The foundations of cost-sensitive learning. Paper presented at the *International Joint Conference on Artificial Intelligence, , 17*(1) 973-978.

F. Reid, & M. Harrigan. (2011). An analysis of anonymity in the bitcoin system. Paper presented at the - *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing,* 1318-1326. doi:10.1109/PASSAT/SocialCom.2011.79

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27*(8), 861-874.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine, 17*(3), 37.

Flach, P. (2012). *Machine learning: The art and science of algorithms that make sense of data* Cambridge University Press.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis, 38*(4), 367-378.

Goldsmith, D., Grauer, K., & Shmalo, Y. (2020). Analyzing hack subnetworks in the bitcoin transaction graph. *Applied Network Science, 5*(1), 1-20.

Hall, M. A. (1999). Correlation-based feature selection for machine learning.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter, 11*(1), 10-18.

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. Paper presented at the *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 1322-1328.

Jang, H., & Lee, J. (2017). An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information. *Ieee Access, 6*, 5427-5437.

Jay, P., Kalariya, V., Parmar, P., Tanwar, S., Kumar, N., & Alazab, M. (2020). Stochastic neural networks for cryptocurrency price prediction. *IEEE Access, 8*, 82804-82818.

Jourdan, M., Blandin, S., Wynter, L., & Deshpande, P. (2018). Characterizing entities in the bitcoin blockchain. Paper presented at the *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, 55-62.

Kavipriya, P., & Karthikeyan, K. (2017). A comparative study of feature selection algorithms in data mining. *Int.J.Adv.Res.Comput.Commun.Eng, 6*(11)

King, G., & Zeng, L. (2001). Logistic regression in rare events data. *Political Analysis, 9*(2), 137-163.

Koops, B. (2010). The internet and its opportunities for cybercrime. *Transnational Criminology Manual, M.Herzog-Evans, Ed, 1*, 735-754.

Liashchynskyi, P., & Liashchynskyi, P. (2019). Grid search, random search, genetic algorithm: A big comparison for NAS. *arXiv Preprint arXiv:1912.06059,*

Ling, C. X., & Sheng, V. S. (2008). Cost-sensitive learning and the class imbalance problem. *Encyclopedia of Machine Learning, 2011*, 231-235.

Linoy, S., Stakhanova, N., & Ray, S. (2021). De-anonymizing ethereum blockchain smart contracts through code attribution. *International Journal of Network Management, 31*(1), e2130.

Liu, X. F., Jiang, X., Liu, S., & Tse, C. K. (2021). Knowledge discovery in cryptocurrency transactions: A survey. *IEEE Access, 9*, 37229-37254.

Loh, W. (2008). Classification and regression tree methods. *Encyclopedia of Statistics in Quality and Reliability, 1*, 315-323.

McKinney, W. (2011). Pandas: A foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing, 14*(9), 1-9.

Mnich, K., & Rudnicki, W. R. (2020). All-relevant feature selection using multidimensional filters with exhaustive search. *Information Sciences, 524*, 277-297.

Montalvo-Garcia, J., Quintero, J. B., & Manrique-Losada, B. (2020). Crisp-dm/smes: A data analytics methodology for non-profit smes. Paper presented at the *Fourth International Congress on Information and Communication Technology*, 449-457.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review, ,* 21260.

Ojala, M., & Garriga, G. C. (2010). Permutation tests for studying classifier performance. *Journal of Machine Learning Research, 11*(6)

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

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

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Dubourg, V. (2011). Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research, 12*, 2825-2830.

Perkel, J. M. (2018). Why jupyter is data scientists' computational notebook of choice. *Nature, 563*(7732), 145-147.

Pham, T., & Lee, S. (2016). Anomaly detection in bitcoin network using unsupervised learning methods. *arXiv Preprint arXiv:1611.03941,*

Provost, F., & Kohavi, R. (1998). Glossary of terms. *Journal of Machine Learning, 30*(2-3), 271-274.

Raileanu, L. E., & Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence, 41*(1), 77-93.

Ranshous, S., Joslyn, C. A., Kreyling, S., Nowak, K., Samatova, N. F., West, C. L., & Winters, S. (2017). Exchange pattern mining in the bitcoin transaction directed hypergraph. Paper

presented at the *International Conference on Financial Cryptography and Data Security*, 248-263.

Saad, M., Choi, J., Nyang, D., Kim, J., & Mohaisen, A. (2019). Toward characterizing blockchain-based cryptocurrencies for highly accurate predictions. *IEEE Systems Journal, 14*(1), 321-332.

Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science, 2*(3), 1-21.

Sathya, R., & Abraham, A. (2013). Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence, 2*(2), 34-38.

Schwab, K. (2017). *The fourth industrial revolution* Currency.

Sun Yin, H. H., Langenheldt, K., Harlev, M., Mukkamala, R. R., & Vatrapu, R. (2019). Regulating cryptocurrencies: A supervised machine learning approach to de-anonymizing the bitcoin blockchain. *Journal of Management Information Systems, 36*(1), 37-73.

Tallón-Ballesteros, A. J., Cavique, L., & Fong, S. (2019). Addressing low dimensionality feature subset selection: ReliefF (-k) or extended correlation-based feature selection (eCFS)? Paper presented at the *International Workshop on Soft Computing Models in Industrial and Environmental Applications*, 251-260.

Toyoda, K., Ohtsuki, T., & Mathiopoulos, P. T. (2018). Multi-class bitcoin-enabled service identification based on transaction history summarization. Paper presented at the *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and*

Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 1153-1160.

VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data* " O'Reilly Media, Inc.".

Wang, J. (2013). Pearson correlation coefficient. *Encyclopedia of Systems Biology, 1671*

Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., & Leiserson, C. E. (2019). Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv Preprint arXiv:1908.02591*,

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., . . . Philip, S. Y. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems, 14*(1), 1-37.

Yeh, S. (2002). Using trapezoidal rule for the area under a curve calculation. *Proceedings of the 27th Annual SAS® User Group International (SUGI'02),*

Yen, S., & Lee, Y. (2006). Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. *Intelligent control and automation* (pp. 731-740) Springer.

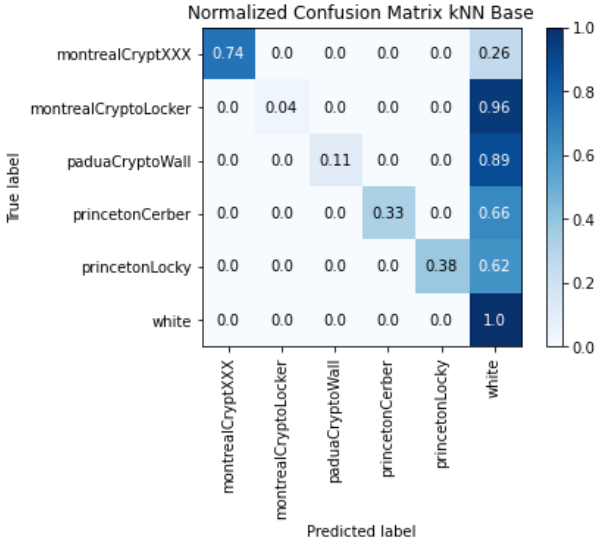
Zimba, A., Simukonda, L., & Chishimba, M. (2017). Demystifying ransomware attacks: Reverse engineering and dynamic malware analysis of wannacry for network and information security. *Zambia ICT Journal, 1*(1), 35-40.

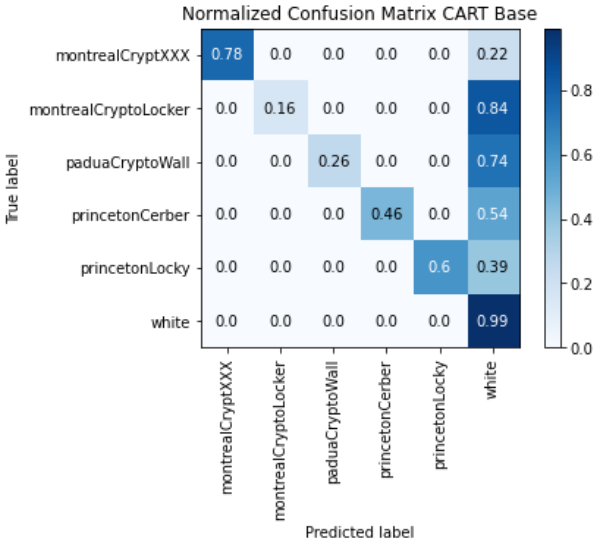
Zola, F., Eguimendia, M., Bruse, J. L., & Urrutia, R. O. (2019). Cascading machine learning to attack bitcoin anonymity. Paper presented at the *2019 IEEE International Conference on Blockchain (Blockchain),* 10-17.

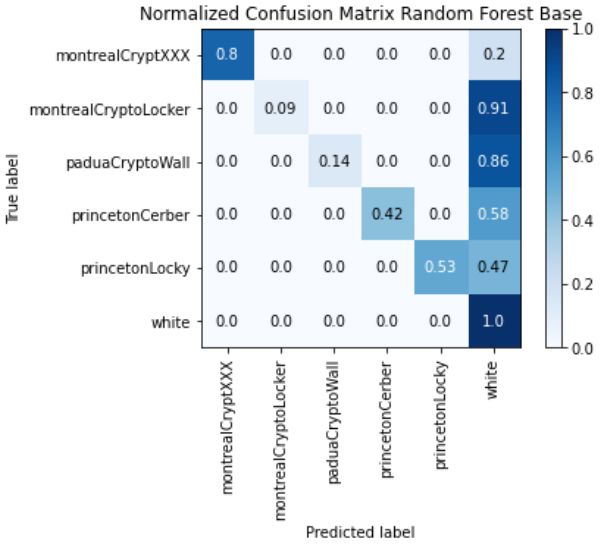
Appendices

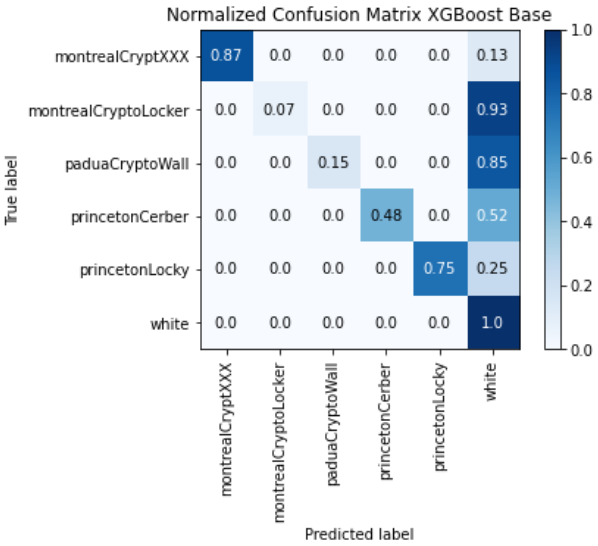
Appendix A. Normalized Confusion Matrices of each scenario studied.

Appendix A.1. Normalized Confusion Matrix from Base feature set.

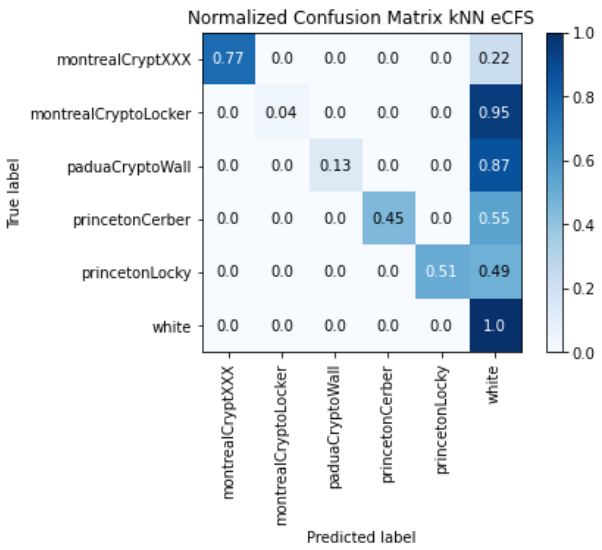


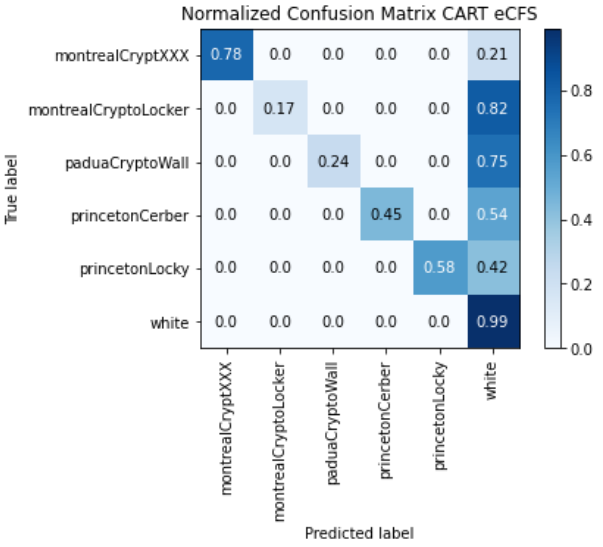


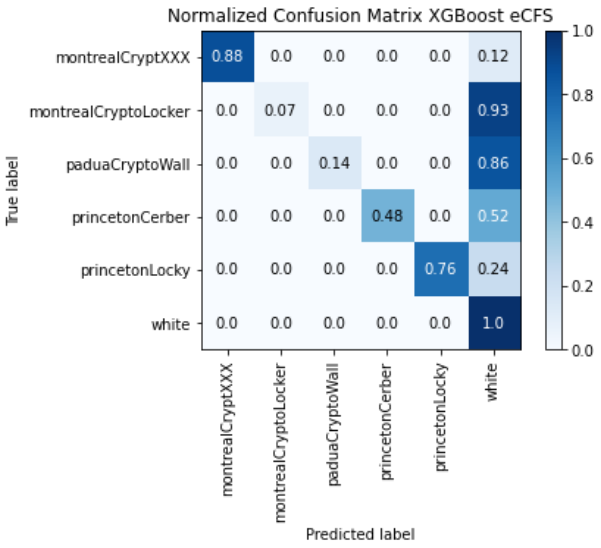


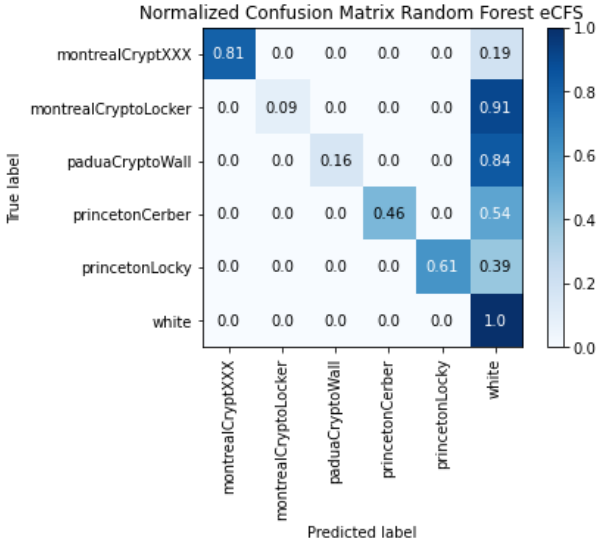


Appendix A.2. Normalized Confusion Matrix from eCFS feature subset.

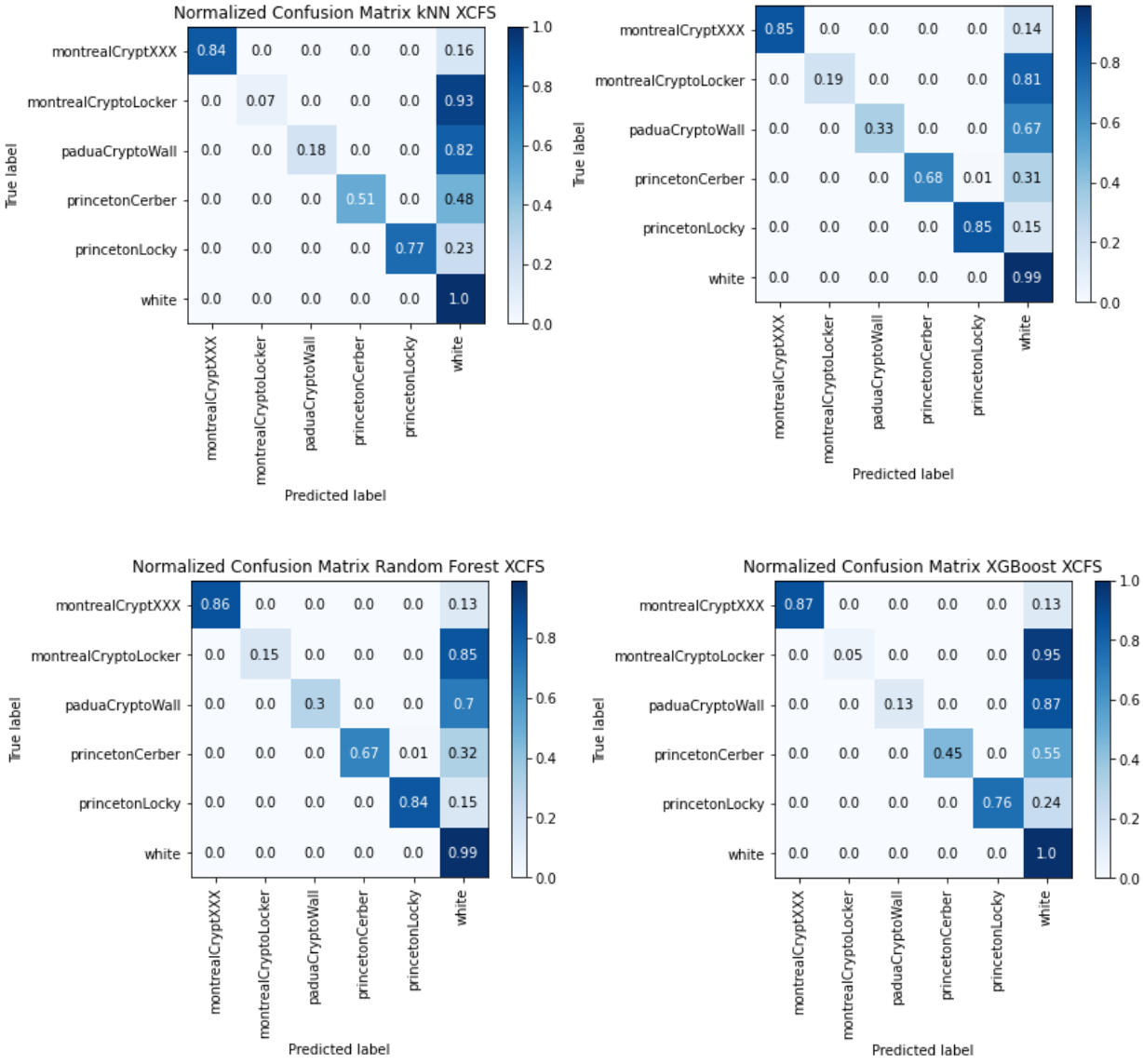








Appendix A.3. Normalized Confusion Matrix from XCFS feature subset.



Appendix B. Random Forest results from 30 different seeds.

Appendix B.1. Results of seed from Random Forest on Base feature subset.

```
SEED: {'random_state ': 111}
elapsed time: 0:01:18.242116 minutes
                     precision    recall  f1-score   support

    montrealCryptXXX     0.9151    0.8030    0.8554       604
montrealCryptoLocker     0.4070    0.0846    0.1401      2329
     paduaCryptoWall     0.5444    0.1385    0.2208      3098
     princetonCerber     0.7836    0.4271    0.5529      2306
      princetonLocky     0.8187    0.5290    0.6427      1656
               white     0.9903    0.9984    0.9943    718821

            accuracy                         0.9888    728814
           macro avg     0.7432    0.4968    0.5677    728814
        weighted avg     0.9855    0.9888    0.9860    728814

Cohen's Kappa: 0.4184425661811627
roc auc score: 0.9497114355128738
--------------------------------------------------------------------------------------------
SEED: {'random_state ': 123}
                     precision    recall  f1-score   support

    montrealCryptXXX     0.9141    0.7930    0.8493       604
montrealCryptoLocker     0.3959    0.0833    0.1376      2329
     paduaCryptoWall     0.5357    0.1356    0.2164      3098
     princetonCerber     0.7772    0.4206    0.5459      2306
      princetonLocky     0.8111    0.5314    0.6421      1656
               white     0.9903    0.9984    0.9943    718821

            accuracy                         0.9887    728814
           macro avg     0.7374    0.4937    0.5643    728814
        weighted avg     0.9853    0.9887    0.9859    728814

Cohen's Kappa: 0.41425530882317085
roc auc score: 0.9492726850586313
--------------------------------------------------------------------------------------------
SEED: {'random_state ': 222}
elapsed time: 0:01:16.715073 minutes
                     precision    recall  f1-score   support

    montrealCryptXXX     0.9140    0.7914    0.8483       604
montrealCryptoLocker     0.3876    0.0807    0.1336      2329
     paduaCryptoWall     0.5405    0.1378    0.2197      3098
     princetonCerber     0.7797    0.4237    0.5490      2306
      princetonLocky     0.8158    0.5242    0.6382      1656
               white     0.9903    0.9984    0.9943    718821

            accuracy                         0.9887    728814
           macro avg     0.7380    0.4927    0.5639    728814
        weighted avg     0.9853    0.9887    0.9859    728814

Cohen's Kappa: 0.41409648768355967
roc auc score: 0.9491407194416545
--------------------------------------------------------------------------------------------
SEED: {'random_state ': 234}
elapsed time: 0:01:21.401691 minutes
                     precision    recall  f1-score   support

    montrealCryptXXX     0.9157    0.7914    0.8490       604
```

```
montrealCryptoLocker     0.4008    0.0842    0.1391      2329
     paduaCryptoWall     0.5409    0.1388    0.2209      3098
     princetonCerber     0.7835    0.4206    0.5474      2306
      princetonLocky     0.8196    0.5296    0.6434      1656
               white     0.9903    0.9984    0.9943    718821

            accuracy                         0.9887    728814
           macro avg     0.7418    0.4938    0.5657    728814
        weighted avg     0.9854    0.9887    0.9860    728814

Cohen's Kappa: 0.41590735249527777
roc auc score: 0.9488817938194224
-----------------------------------------------------------------------------------------
SEED: {'random_state ': 456}
elapsed time: 0:01:19.922763 minutes
                      precision    recall  f1-score   support

    montrealCryptXXX     0.9143    0.7947    0.8503       604
montrealCryptoLocker     0.4020    0.0854    0.1409      2329
     paduaCryptoWall     0.5399    0.1375    0.2192      3098
     princetonCerber     0.7756    0.4211    0.5458      2306
      princetonLocky     0.8148    0.5127    0.6294      1656
               white     0.9903    0.9984    0.9943    718821

            accuracy                         0.9887    728814
           macro avg     0.7395    0.4916    0.5633    728814
        weighted avg     0.9853    0.9887    0.9859    728814

Cohen's Kappa: 0.4125338675640249
roc auc score: 0.9495523162753577
-----------------------------------------------------------------------------------------
SEED: {'random_state ': 124}
elapsed time: 0:01:25.055930 minutes
                      precision    recall  f1-score   support

    montrealCryptXXX     0.9235    0.7997    0.8571       604
montrealCryptoLocker     0.3947    0.0837    0.1382      2329
     paduaCryptoWall     0.5339    0.1346    0.2150      3098
     princetonCerber     0.7841    0.4267    0.5527      2306
      princetonLocky     0.8168    0.5332    0.6452      1656
               white     0.9903    0.9984    0.9943    718821

            accuracy                         0.9888    728814
           macro avg     0.7406    0.4961    0.5671    728814
        weighted avg     0.9854    0.9888    0.9860    728814

Cohen's Kappa: 0.4168676647394576
roc auc score: 0.9499861806298356
-----------------------------------------------------------------------------------------
SEED: {'random_state ': 221}
elapsed time: 0:01:19.527963 minutes
                      precision    recall  f1-score   support

    montrealCryptXXX     0.9195    0.7947    0.8526       604
montrealCryptoLocker     0.3900    0.0837    0.1379      2329
     paduaCryptoWall     0.5317    0.1352    0.2156      3098
     princetonCerber     0.7822    0.4159    0.5430      2306
      princetonLocky     0.8204    0.5296    0.6437      1656
               white     0.9903    0.9984    0.9943    718821

            accuracy                         0.9887    728814
           macro avg     0.7390    0.4929    0.5645    728814
        weighted avg     0.9853    0.9887    0.9859    728814

Cohen's Kappa: 0.4131871579077505
roc auc score: 0.9476592440389755
```

Appendix B.2. Results of seed from Random Forest on eCFS feature subset.

```
SEED: {'random_state ': 111}
elapsed time: 0:01:09.171795 minutes
                      precision    recall  f1-score   support

    montrealCryptXXX     0.9026    0.7980    0.8471       604
montrealCryptoLocker     0.3267    0.0919    0.1434      2329
     paduaCryptoWall     0.5056    0.1598    0.2428      3098
     princetonCerber     0.7330    0.4536    0.5604      2306
      princetonLocky     0.8131    0.6069    0.6950      1656
               white     0.9907    0.9978    0.9942    718821

            accuracy                         0.9886    728814
           macro avg     0.7120    0.5180    0.5805    728814
        weighted avg     0.9852    0.9886    0.9861    728814

Cohen's Kappa: 0.43494662702029285
roc auc score: 0.9434081834391274
----------------------------------------------------------------------------------------------
SEED: {'random_state ': 123}
elapsed time: 0:01:09.765087 minutes
                      precision    recall  f1-score   support

    montrealCryptXXX     0.9072    0.7930    0.8463       604
montrealCryptoLocker     0.3288    0.0927    0.1447      2329
     paduaCryptoWall     0.5120    0.1578    0.2413      3098
     princetonCerber     0.7325    0.4571    0.5629      2306
      princetonLocky     0.8099    0.6123    0.6974      1656
               white     0.9907    0.9978    0.9942    718821

            accuracy                         0.9886    728814
           macro avg     0.7135    0.5185    0.5811    728814
        weighted avg     0.9853    0.9886    0.9862    728814

Cohen's Kappa: 0.43637196480358875
roc auc score: 0.9430684424930051
----------------------------------------------------------------------------------------------
SEED: {'random_state ': 222}
elapsed time: 0:01:08.184019 minutes
                      precision    recall  f1-score   support

    montrealCryptXXX     0.9118    0.8046    0.8549       604
montrealCryptoLocker     0.3278    0.0940    0.1461      2329
     paduaCryptoWall     0.5025    0.1595    0.2421      3098
     princetonCerber     0.7307    0.4566    0.5620      2306
      princetonLocky     0.8089    0.6135    0.6978      1656
               white     0.9907    0.9978    0.9942    718821

            accuracy                         0.9886    728814
           macro avg     0.7121    0.5210    0.5829    728814
        weighted avg     0.9852    0.9886    0.9862    728814

Cohen's Kappa: 0.4369598868998912
roc auc score: 0.9433822407037198
----------------------------------------------------------------------------------------------
SEED: {'random_state ': 234}
elapsed time: 0:01:07.748275 minutes
                      precision    recall  f1-score   support

    montrealCryptXXX     0.8987    0.8079    0.8509       604
montrealCryptoLocker     0.3293    0.0936    0.1458      2329
     paduaCryptoWall     0.5099    0.1582    0.2414      3098
     princetonCerber     0.7390    0.4592    0.5665      2306
      princetonLocky     0.8159    0.6129    0.7000      1656
```

```
               white     0.9907    0.9978    0.9943    718821

            accuracy                         0.9886    728814
           macro avg     0.7139    0.5216    0.5831    728814
        weighted avg     0.9853    0.9886    0.9862    728814

Cohen's Kappa: 0.4381788303179135
roc auc score: 0.9442334432137223
-----------------------------------------------------------------------------------------------
SEED: {'random_state ': 456}
elapsed time: 0:01:10.085568 minutes
                      precision    recall  f1-score   support

    montrealCryptXXX     0.9096    0.7997    0.8511       604
montrealCryptoLocker     0.3236    0.0906    0.1416      2329
     paduaCryptoWall     0.5166    0.1611    0.2456      3098
     princetonCerber     0.7313    0.4545    0.5606      2306
      princetonLocky     0.8106    0.6075    0.6945      1656
               white     0.9907    0.9978    0.9942    718821

            accuracy                         0.9886    728814
           macro avg     0.7137    0.5185    0.5813    728814
        weighted avg     0.9852    0.9886    0.9862    728814

Cohen's Kappa: 0.43573127298066685
roc auc score: 0.9427998090759847
-----------------------------------------------------------------------------------------------
SEED: {'random_state ': 124}
elapsed time: 0:01:08.765559 minutes
                      precision    recall  f1-score   support

    montrealCryptXXX     0.9103    0.8063    0.8551       604
montrealCryptoLocker     0.3226    0.0902    0.1409      2329
     paduaCryptoWall     0.5046    0.1582    0.2408      3098
     princetonCerber     0.7304    0.4558    0.5613      2306
      princetonLocky     0.8129    0.6111    0.6977      1656
               white     0.9907    0.9978    0.9942    718821

            accuracy                         0.9886    728814
           macro avg     0.7119    0.5199    0.5817    728814
        weighted avg     0.9852    0.9886    0.9861    728814

Cohen's Kappa: 0.43559501276497437
roc auc score: 0.9432287075724529
-----------------------------------------------------------------------------------------------
SEED: {'random_state ': 221}
elapsed time: 0:01:08.303372 minutes
                      precision    recall  f1-score   support

    montrealCryptXXX     0.9041    0.7964    0.8468       604
montrealCryptoLocker     0.3263    0.0927    0.1444      2329
     paduaCryptoWall     0.5133    0.1614    0.2456      3098
     princetonCerber     0.7332    0.4540    0.5608      2306
      princetonLocky     0.8069    0.6105    0.6951      1656
               white     0.9907    0.9978    0.9942    718821

            accuracy                         0.9886    728814
           macro avg     0.7124    0.5188    0.5812    728814
        weighted avg     0.9852    0.9886    0.9862    728814

Cohen's Kappa: 0.43603108643058
roc auc score: 0.9427206803394461
```

Appendix B.3. Results of seed from Random Forest on XCFS feature subset.

```
SEED: {'random_state ': 111}
elapsed time: 0:01:05.163375 minutes
```

```
                     precision    recall  f1-score   support

    montrealCryptXXX     0.8763    0.8675    0.8719       604
montrealCryptoLocker     0.1396    0.1537    0.1463      2329
     paduaCryptoWall     0.3451    0.2999    0.3209      3098
     princetonCerber     0.5241    0.6696    0.5880      2306
      princetonLocky     0.6710    0.8436    0.7475      1656
               white     0.9928    0.9915    0.9921    718821

            accuracy                         0.9844    728814
           macro avg     0.5915    0.6376    0.6111    728814
        weighted avg     0.9850    0.9844    0.9846    728814

Cohen's Kappa: 0.45219340860099655
roc auc score: 0.9008175256103877
----------------------------------------------------------------------------------------------
SEED: {'random_state ': 123}
elapsed time: 0:01:05.044806 minutes
                     precision    recall  f1-score   support

    montrealCryptXXX     0.8760    0.8659    0.8709       604
montrealCryptoLocker     0.1400    0.1546    0.1469      2329
     paduaCryptoWall     0.3452    0.3002    0.3211      3098
     princetonCerber     0.5233    0.6678    0.5868      2306
      princetonLocky     0.6746    0.8424    0.7492      1656
               white     0.9927    0.9915    0.9921    718821

            accuracy                         0.9844    728814
           macro avg     0.5920    0.6371    0.6112    728814
        weighted avg     0.9850    0.9844    0.9846    728814

Cohen's Kappa: 0.45193630090700687
roc auc score: 0.9015352739098553
----------------------------------------------------------------------------------------------
SEED: {'random_state ': 222}
elapsed time: 0:01:04.153473 minutes
                     precision    recall  f1-score   support

    montrealCryptXXX     0.8746    0.8659    0.8702       604
montrealCryptoLocker     0.1372    0.1516    0.1441      2329
     paduaCryptoWall     0.3386    0.2950    0.3153      3098
     princetonCerber     0.5229    0.6678    0.5866      2306
      princetonLocky     0.6726    0.8412    0.7475      1656
               white     0.9927    0.9915    0.9921    718821

            accuracy                         0.9844    728814
           macro avg     0.5898    0.6355    0.6093    728814
        weighted avg     0.9849    0.9844    0.9846    728814

Cohen's Kappa: 0.4494683424611059
roc auc score: 0.9012223682169104
----------------------------------------------------------------------------------------------
SEED: {'random_state ': 234}
elapsed time: 0:01:03.041988 minutes
                     precision    recall  f1-score   support

    montrealCryptXXX     0.8729    0.8642    0.8686       604
montrealCryptoLocker     0.1366    0.1516    0.1437      2329
     paduaCryptoWall     0.3416    0.2966    0.3176      3098
     princetonCerber     0.5245    0.6683    0.5877      2306
      princetonLocky     0.6721    0.8406    0.7470      1656
               white     0.9927    0.9915    0.9921    718821

            accuracy                         0.9844    728814
           macro avg     0.5901    0.6355    0.6094    728814
        weighted avg     0.9849    0.9844    0.9846    728814

Cohen's Kappa: 0.4498472683552991
roc auc score: 0.9013118810516
----------------------------------------------------------------------------------------------
SEED: {'random_state ': 456}
elapsed time: 0:01:04.510770 minutes
```

```
                     precision    recall  f1-score   support

    montrealCryptXXX     0.8773    0.8642    0.8707       604
montrealCryptoLocker     0.1386    0.1520    0.1450      2329
     paduaCryptoWall     0.3461    0.3012    0.3221      3098
     princetonCerber     0.5230    0.6657    0.5858      2306
      princetonLocky     0.6726    0.8424    0.7480      1656
               white     0.9927    0.9915    0.9921    718821

            accuracy                         0.9844    728814
           macro avg     0.5917    0.6362    0.6106    728814
        weighted avg     0.9849    0.9844    0.9846    728814

Cohen's Kappa: 0.45153765638374554
roc auc score: 0.9023455287070979
----------------------------------------------------------------------------------------------------
SEED: {'random_state ': 124}
elapsed time: 0:01:06.649751 minutes
                     precision    recall  f1-score   support

    montrealCryptXXX     0.8792    0.8675    0.8733       604
montrealCryptoLocker     0.1381    0.1516    0.1445      2329
     paduaCryptoWall     0.3473    0.2992    0.3215      3098
     princetonCerber     0.5266    0.6700    0.5897      2306
      princetonLocky     0.6724    0.8430    0.7481      1656
               white     0.9927    0.9916    0.9922    718821

            accuracy                         0.9845    728814
           macro avg     0.5927    0.6372    0.6116    728814
        weighted avg     0.9850    0.9845    0.9847    728814

Cohen's Kappa: 0.45270411168679625
roc auc score: 0.9011935345065591
----------------------------------------------------------------------------------------------------
SEED: {'random_state ': 221}
elapsed time: 0:01:03.709106 minutes
                     precision    recall  f1-score   support

    montrealCryptXXX     0.8790    0.8659    0.8724       604
montrealCryptoLocker     0.1386    0.1529    0.1454      2329
     paduaCryptoWall     0.3450    0.2989    0.3203      3098
     princetonCerber     0.5231    0.6674    0.5865      2306
      princetonLocky     0.6720    0.8412    0.7471      1656
               white     0.9927    0.9915    0.9921    718821

            accuracy                         0.9844    728814
           macro avg     0.5917    0.6363    0.6106    728814
        weighted avg     0.9849    0.9844    0.9846    728814

Cohen's Kappa: 0.4512145320528079
roc auc score: 0.9005525358577926
```