

## Article

# On Detecting Cryptojacking on Websites: Revisiting the Use of Classifiers

Fredy Andrés Aponte-Novoa <sup>1,2,\*</sup> , Daniel Povedano Álvarez <sup>3</sup> , Ricardo Villanueva-Polanco <sup>1</sup> ,  
Ana Lucila Sandoval Orozco <sup>3</sup>  and Luis Javier García Villalba <sup>3</sup> 

<sup>1</sup> Department of Computer Science and Engineering, Universidad del Norte, Barranquilla 081007, Colombia  
<sup>2</sup> Department of Systems Engineering, Universidad Santo Tomás, Tunja 150003, Colombia  
<sup>3</sup> Group of Analysis, Security and Systems (GASS), Department of Software Engineering and Artificial Intelligence (DISIA), Faculty of Computer Science and Engineering, Office 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases 9, Ciudad Universitaria, 28040 Madrid, Spain  
\* Correspondence: faponte@uninorte.edu.co

**Abstract:** Cryptojacking or illegal mining is a form of malware that hides in the victim's computer and takes the computational resources to extract cryptocurrencies in favor of the attacker. It generates significant computational consumption, reducing the computational efficiency of the victim's computer. This attack has increased due to the rise of cryptocurrencies and their profitability and its difficult detection by the user. The identification and blocking of this type of malware have become an aspect of research related to cryptocurrencies and blockchain technology; in the literature, some machine learning and deep learning techniques are presented, but they are still susceptible to improvement. In this work, we explore multiple Machine Learning classification models for detecting cryptojacking on websites, such as Logistic Regression, Decision Tree, Random Forest, Gradient Boosting Classifier, *k*-Nearest Neighbor, and XGBoost. To this end, we make use of a dataset, composed of network and host features' samples, to which we apply various feature selection methods such as those based on statistical methods, e.g., Test Anova, and other methods as Wrappers, not only to reduce the complexity of the built models but also to discover the features with the greatest predictive power. Our results suggest that simple models such as Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, and *k*-Nearest Neighbor models, can achieve success rate similar to or greater than that of advanced algorithms such as XGBoost and even those of other works based on Deep Learning.

**Keywords:** blockchain; cryptojacking; illegal mining; malware; machine learning



**Citation:** Aponte-Novoa, F.A.; Povedano Álvarez, D.; Villanueva-Polanco, R.; Sandoval Orozco, A.L.; García Villalba, L.J. On Detecting Cryptojacking on Websites: Revisiting the Use of Classifiers. *Sensors* **2022**, *22*, 9219. <https://doi.org/10.3390/s22239219>

Academic Editors: Nima Karimian and Hossein Sayadi

Received: 8 November 2022  
Accepted: 24 November 2022  
Published: 27 November 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Cryptojacking is an illegal and unauthorized mining activity on the victim's computer, using the computational power of the victim's computer to extract cryptocurrencies, which generates large computational consumption, reducing the computational efficiency of the victim's computer. Moreover, this attack may be used by a powerful attacker to increment their computationally power, posing a risk to any blockchain based on mining [1–7].

Cryptojacking detection techniques such as browser extensions and antiviruses provide a partial solution to the cryptojacking problem since attackers can avoid them by employing obfuscation techniques or renewing domains or malicious scripts relatively frequently [8]. Cryptojacking boomed with the birth of service providers that offer ready-to-use implementations of mining scripts in web browsers. Therefore, attackers can reach many more victims via websites. These service providers are coinhive [9] and cryptoloot [10].

Cryptojacking on websites uses JavaScript code to mine cryptocurrencies. This technique does not require installing javascript code to perform the mining process. All it takes is for the user to load the infected website in their browser for the illegal mining code to execute in the browser of the victim's computer [11]. According to [12], many websites

have been infected by cryptojacking, such as personal blogs up to Alexa-ranking websites. Moreover, it noted that as of January 2022, there were around 3000 websites that offered online cryptojacking scripts.

As evidenced in [13–19], there is a recent trend of applying specialized deep learning models to detect cryptojacking in websites. However, specialized deep learning models might pose challenges with reference to their deployment and performance. Thus, this paper seeks to explore machine learning models (comparatively simpler than deep learning models) for cryptojacking classification to identify which of these machine learning models may render desirable results. Hence, this work takes previous works as a reference, particularly the recent specialized deep learning model and the dataset presented and collected in [14]. We use this dataset to train and validate multiple machine learning classification algorithms to detect cryptojacking on websites and make a comparison between the machine learning models and the specialized deep learning model.

The rest of this paper is structured as follows: In Section 2, we present three cases of cryptojacking as a motivation for addressing cryptojacking detection. Section 3 presents related works on cryptojacking detection techniques. Section 4 describes the methodology we follow for our study. Also, we describe the dataset's selection and content and present an exploratory data analysis. This section also shows data correlation and clustering, as well as the feature selection, split, and normalization process. At the end of this section, we present some machine-learning models for detecting cryptojacking on websites. Section 5 analyses the results we obtained after going through our methodology. Section 6 narrates a possible integration scenario of our techniques with other known approaches for detecting malware and filtering/blocking websites. Finally, we give some conclusions in Section 7.

## 2. Some Cases of Cryptojacking

### 2.1. Cryptojacking on Websites

Websites of different kinds have been victims of strong cryptojacking-type malicious software actions; some famous cases are: In early 2018, the online video-sharing platform YouTube was illegally compromised where the CoinHive miner ran on its ads [20,21], and the Russian Nuclear Weapon Research Center [22]

### 2.2. Cryptojacking with Advanced Techniques

Other attacks have used advanced techniques in the spread of cryptojacking; an example of this was when the botnet called vollgar attacked all Microsoft SQL (MSSQL) databases servers to take control of administrative accounts and inject malicious miners into said servers [23]. Another example of the use of these advanced techniques was presented with the Zoom video conferencing software. In this case, the attackers merged cryptojacking malware with the main zoom application and published it on different file-sharing platforms [24]. Similarly, this technique was used in the Nintendo Switch consoles [25]. Another use case of these advanced techniques was presented in MikroTik routers between July and August of the year 2018, where a cryptojacking campaign managed to compromise more than 200,000 MikroTik routers; these routers were primarily located in Brazil; in the same way, researchers observed that routers that were not MikroTik were also compromised [26]. Moreover, a recent study explores cryptojacking and its impact on Electric Vehicles [27].

In 2019, eight applications were detected and removed from the Microsoft Store in Windows 10. When the user installed and opened these applications, they secretly downloaded cryptojacking's JavaScript code, which carried out mining tasks for the Monero cryptocurrency, notably affecting the user device performance [28,29].

In 2018, researchers from RedLock dedicated to computer security discovered that at least one unknown computer criminal broke into an Amazon cloud account associated with Tesla and used it to mine cryptocurrencies. In this attack, the Stratum mining protocol was used, and the true IP address of the mining pool was hidden and kept the CPU consumption low [30].

### 2.3. Cryptojacking in Industrial Control Systems or Critical Servers

The impact of cryptojacking has surpassed the borders of traditional websites, affecting industrial control systems and Critical Servers; some of those cases are: In January 2020, following a report on the bug bounty website [www.hackerone.com](http://www.hackerone.com) (accessed on 1 November 2022), the US Department of Defense discovered that its government and military servers were affected by cryptojacking to mine the currency Monero illegally [31,32]. In 2019, a Russian nuclear warhead facility employee was fined around \$7000 for illegally mining bitcoin using the facility's servers [33]

### 3. Related Works

The literature presents different techniques for the detection of cryptojacking on websites; however, these solutions present some limitations related to performance, transparency and effectiveness.

A hardware-based approach to cryptojacking detection is presented in [13]; this method takes advantage of the Intel Processor Trace mechanism to collect control flow information at runtime from the web browser. This technique uses two optimization approaches based on library functionality and information gain to preprocess control flow information. It also takes advantage of a Recurrent Neural Network (RNN) for cryptojacking detection.

A method that performs a fingerprinting technique to detect possible malicious sites is presented in [14], which is then characterized with an autoencoding algorithm that keeps the best information of the infection vestiges to maximize the classification power by means of a deep dense neural network.

A lightweight cryptojacking detection system that uses deep learning techniques to accurately detect the presence of unwarranted mining activity based on emerging WebAssembly (Wasm)-based cryptojacking malware in real-time is introduced at [15]. This system employs an image-based classification technique to distinguish between benign web pages and those using Wasm. Specifically, the classifier implements a convolutional neural network (CNN) model.

A detection and control method for IoT botnets is presented in [16]; it uses a deep learning model, and cryptojacking activities carried out by the bot. This method performs malicious attack detection by implementing a sparse autoencoder composed of an input layer, a hidden layer, and an output layer.

A method for detecting silent browser mining behavior is presented in [17]; this method drives known malicious mining samples, extracts heap snapshots and stack code functions of a dynamically running browser, and performs automatic detection based on a recurrent neural network (RNN).

A method called CapJack to identify illicit bitcoin mining activity in a web browser using cutting-edge CapsNet technology is presented in [18]. Deep learning framework CapsNet employs heuristics based on system behavior to detect malware effectively.

A detection method called CoinPolice is presented in [19]; that method flips throttling against cryptojackers, artificially varying the browser's CPU power to observe the presence of throttling. Based on a deep neural network classifier, coinPolice can detect hidden miners.

### 4. Methodology

In this work, we follow a methodology, whose phases are shown in Figure 1 and described next.

1. Selection of the dataset.
2. Exploratory data analysis. In this phase, we calculate a correlation matrix and perform clustering of the dataset. Additionally, we perform feature selection process, dataset normalization and dataset splitting.

3. Exploration of Classification Models. In this phase, we reproduce the Sparse Autoencoder + Deep dense Neural Network model for Cryptojacking detection [14]. Additionally, we train and test different classifiers for Cryptojacking detection.
4. Results' Presentation. We present the results of applying *K*-means clustering to the dataset, as well as the results of cross-validation and evaluation of the different classification models.

The development of these phases was carried out in a notebook in the Google Colaboratory (Google Colab) environment with the python programming language and libraries such as numpy, pandas, seaborn, matplotlib, tensorflow, keras and sklearn. The interested reader can see the source code here [34]. The different classification models were executed in the virtual machine offered by Google Colab, which has the following characteristics: 2.20GHz Intel Xeon processor, 12G of RAM, and Ubuntu 18.04.6 LTS Operating system.



**Figure 1.** Methodology .

#### 4.1. Selection of Dataset

We use the dataset that was collected in [14]. This dataset contains records of host-based and network-based features associated with websites. Moreover, each record is labeled whether a website is infected or not by cryptojacking. According to [14], the procedure to capture these samples includes three layers: the first of these is called fingerprinting, in which the sites that may contain signs of cryptomining are captured; after this, data based on the network is captured, which includes the information flows of the traffic that transits through the HTTP, HTTPS, and TCP protocols, and finally, the host-based data is captured, which consists of the tracking of the website once that interacts with the browser. The dataset used in this paper comprises 9292 benign sites and 3434 sites labeled and validated as cryptojacking infected. The host-based and network-based features that compose the dataset are indicated in Table 1 [14]. In addition to these features, the dataset presents an attribute called label, representing the class or classification of the sample, where 1 indicates a sample of a site labeled as cryptojacking infected and 0 as a benign site.

**Table 1.** Host-based and Network-based Features.

Host-Based	
Feature	Description
C1	Time in C1 is a subset of the total processor idle time
C2	Time at C2 is a state of lower energy and higher output latency than Time at C1
C3	Time at C3 is a lower energy state and higher output latency than Time at C2
I/O Data Operations	Speed at which the process is issuing read and write I/O operations
I/O Data Bytes	Speed at which the process is reading and writing bytes in I/O operations
Number of subprocesses	Number of sub-processes that are currently active in a parent process
Time on processor	The total time, in seconds, that a process has been running
Disk Reading/sec	Speed of disk reading operations
Disc Writing/sec	Speed of writing operations to disk
Confirmed byte radius	The ratio of Memory/Bytes committed and Memory/Confirmation limit
Percentage of processor usage	Elapsed time on the processor when an active thread is running
Pages Read/sec	Speed rate at which the disk was read in order to resolve hard page errors
Pages Input/sec	Speed at which pages are written to disk to free up space in physical memory
Page Errors/sec	This is the average number of pages with faults per second

Table 1. Cont.

Network-Based	
Feature	Description
Bytes Sent	The rate at which bytes leave the browser’s HTTP requests
Received Bytes (HTTP)	Speed of bytes arriving to the browser’s HTTP responses
Network packets sent	Speed of sending packets in the TCP protocol
Network packets received	Packet reception speed over the TCP protocol

### 4.2. Exploratory Data Analysis

This section describes the data correlation and clustering, as well as the feature selection, splitting, and normalization process on the dataset.

#### 4.2.1. Correlation

We verify that the dataset does not present missing cells and duplicate rows. As expected, the dataset does not present these cases thanks to the fact that after its collection, its collectors went through a pre-processing process. To identify the existing correlation between the different features that make up the dataset and the classification class, we built a correlation matrix by employing the method `corr` of `pandas.DataFrame` with the default parameters (`method = 'pearson'`, `min_periods = 1`, `numeric_only = _NoDefault.no_default`) (see Figure 2).

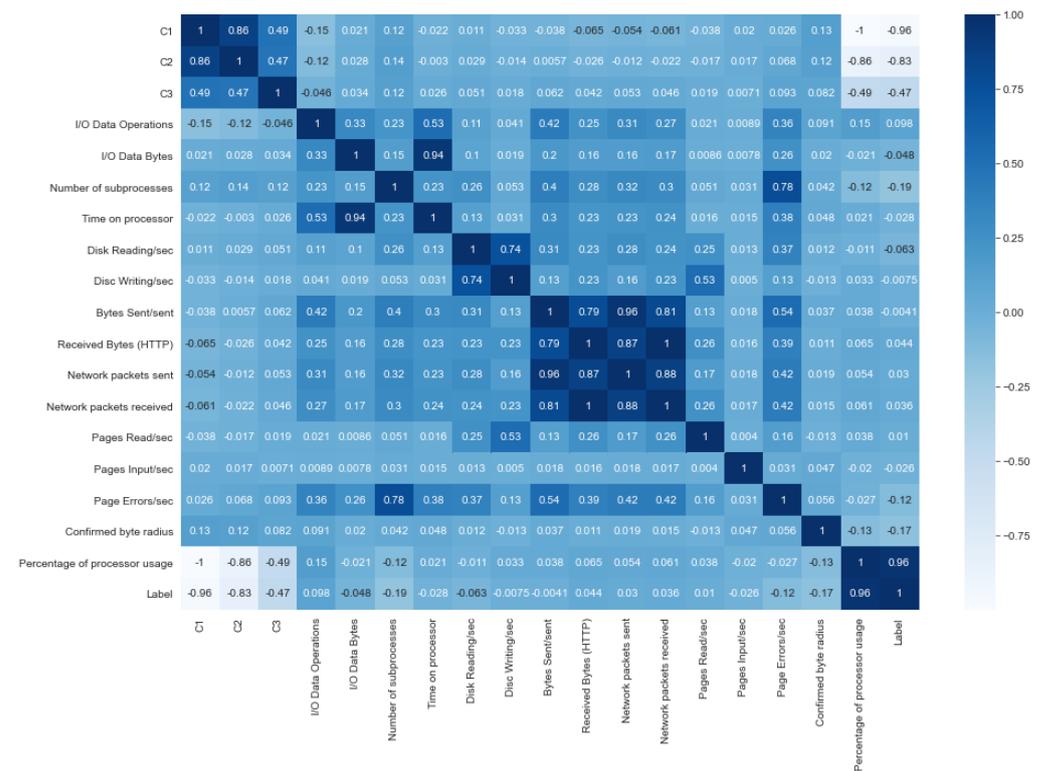


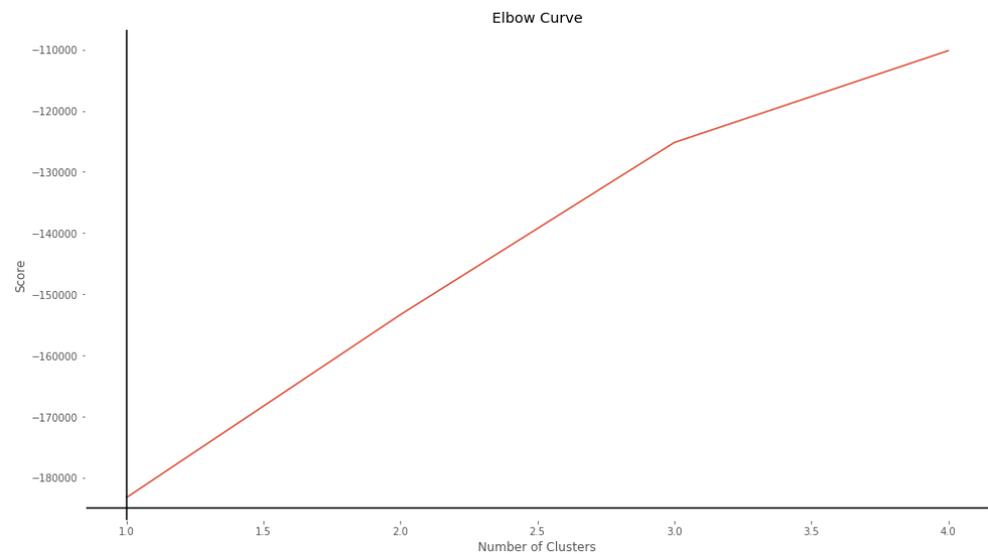
Figure 2. Feature correlation matrix.

This matrix tells us that the feature “utility of the processor” is the one that presents the highest positive correlation with the classification of the website. On the other hand, the features C1, C2, C3, “Number of subprocesses”, “confirmed byte radius” and “Page Errors/sec” have, in their order, the highest negative correlation with website ranking.

#### 4.2.2. Clustering

Due to the essence of the dataset, we know that its records are divided into two categories; the first is the records of sites infected with Cryptojacking (labeled with 1) and

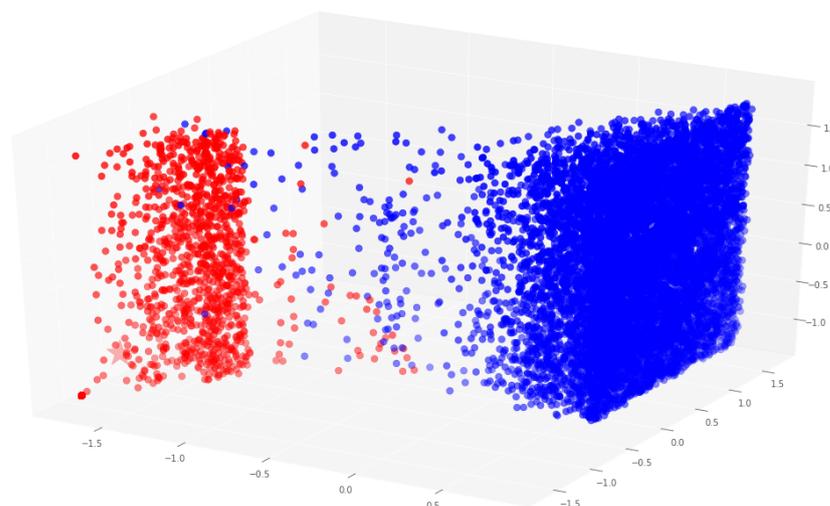
the records of benign sites (labeled with 0). To verify and analyze how well these records are grouped according to their features, we use the Unsupervised Learning algorithm  $k$ -Means Clustering, which groups the unlabeled dataset into different clusters. First, we used the elbow method to determine the number of clusters in the dataset. Figure 3 shows the elbow of the curve at three, but for the reason that we explained previously, we give two as the number of clusters parameter to the  $k$ -means algorithm.



**Figure 3.** Elbow curve.

Next, we cluster the dataset using the  $k$ -means algorithm with the parameters  $n\_clusters = 2$ ,  $random\_state = 0$  and  $init = "k-means++"$ . " $k$ -means++" is a method that chooses the first centroid to the location of a randomly selected data point and then chooses subsequent centroids of the remaining data points based on a probability proportional to the square of the distance from the nearest existing centroid of a given point. It helps in choosing the centroids to be as far away as possible, trying to cover the data space occupied since the initialization [35].

We apply the  $k$ -means algorithm to all the dataset entries and generate a figure to visualize the group assignment for each sample of the dataset group, only taking C1, C2, and C3 features. Figure 4 shows that the application of clustering to the dataset results in two well-defined groups corresponding to the samples of benign sites and infected websites.



**Figure 4.** Clustering of dataset entries.

#### 4.2.3. Feature Selection

Extracting relevant features from the raw data is paramount for Intrusion Detection System (IDS) classification [36]. For feature selection, first, we select the features with a  $p$ -value less than or equal to 0.05 and later apply statistical methods with the `f_classif` function and the univariate filtering function “Anova Scores” for 10 and 4 features. “Anova Score” fits a simple linear model between a single feature and the outcome, then the  $p$ -value for the whole model `Ftest` is returned [37]. In addition, an RFECV wrapper method was applied, which is a Feature ranking with recursive feature elimination and cross-validated selection of the best number of features [38]. The parameters used in the feature selection are in Table 2. For each of these filters, we extracted a subset of data that we used in the cross-validation.

**Table 2.** Parameters used in the feature selection methods.

Method	Parameters
$p$ -value	$p$ -value $\leq$ 0.05
Anova scores	<code>n_splits=5, n_repeats=5, random_state=12345, score_func = f_classif, k= 5/10</code>
RFECV wrapper	<code>estimator = LogisticRegression, step = 1, cv = StratifiedKFold(n_splits = 5, random_state = 12345, shuffle = True), min_features_to_select = 1, scoring = 'f1', verbose = 2, n_jobs = -1</code>

#### 4.2.4. Split and Normalization of the Dataset

After exploring the dataset, we separate the features in a  $X$  set and the label category in a  $Y$  set; we then split the records of the dataset for training (80% of them) and testing (20% of them). For this partition, we use the parameters `random_state = 42` and `stratify = y`. The `stratify` parameter makes a split so that the proportion of values in the sample produced will be the same as the proportion of values provided to the parameter, `stratify`. Because the ranges for the different data features are very different, we apply the Standard Scaler to the training and testing datasets for the original dataset and the other datasets created as explained in Section 4.2.3.

#### 4.3. Exploration of Classification Models

Having the dataset, we prepare a subset of training data and a subset of test data. Moreover, we applied normalization to the data. With the normalized data, we replicate the sparse autoencoder (SAE) + deep, dense neural network (DDNN) model, proposed in [14], to have a point of comparison for the proposed models. Later, some techniques are applied for the selection of features; after this, six reference models are defined for classification such as Logistic regression, Decision Tree, Random Forest, Gradient Boosting,  $k$ -Nearest Neighbor, and XGBoost. With these reference models, the different datasets obtained in the selection of features are tested, performing Cross-Validation for each of these sets. Finally, the performance of the classification models is measured using precision, recall, and F1-score.

##### 4.3.1. Sparse Autoencoder + Deep dense Neural Network

According to the results shown in [14], the Sparse Autoencoder + Deep dense Neural Network model is an state-of-the-art model for cryptojacking detection. We refer the reader to [14] to see a deep comparison between this model and other proposed models. In this paper, we reproduce this model to validate the results obtained with the corresponding dataset and thus have a point of comparison with the other models explored in this work.

##### 4.3.2. Classification Models

In this work, we consider the following classification models.

- Logistic regression or LR is a standard probabilistic statistical classification model widely used in different disciplines, such as computer vision, and marketing, among others [39].
- A Decision Tree is a hierarchical structure built using a data set's features (independent variables). In a decision tree, each node is partitioned according to a measure associated with a subset of features [40]. This algorithm repeatedly divides the data set according to a criterion that seeks to maximize the separation of the data, resulting in a tree-like structure [41].
- A random forest is a collection of decision trees associated with a set of bootstrap samples generated from the original data set. The nodes are partitioned based on the entropy or Gini index of a selected subset of the features [40].
- Gradient Boosting is a widely used machine learning algorithm due to its efficiency, accuracy, and interpretability [42]. This algorithm achieves state-of-the-art performance in many machine learning tasks, such as multi-class classification [43], click prediction [44] and learning to rank [45].
- *k*-Nearest Neighbor classifier, unlike other methods, uses the data directly for classification, without first building a model [46,47]. One of the advantages of the *k*-nearest neighbors algorithm over other algorithms is the fact that the neighbors can provide an explanation for the classification result [48].
- XGBoost is a scalable ensemble machine learning and gradient boosting technique focusing on performance and speed. This technique allows for solving problems of ranking, classification, and regression [49].

We instantiate and train the classification models: Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, *k*-Nearest Neighbor, and XGBoost. Table 3 shows the parameters used in these algorithms. With the different models, cross-validation is performed with the different datasets, and the mean and standard deviation of the accuracy are calculated.

**Table 3.** Models' parameters.

Model	Parameters
Logistic regression	default
Decision Tree	criterion = 'gini', max_depth = 5, min_samples_leaf = 20, random_state = 12345
Random Forest	n_estimators = 25
Gradient Boosting	n_estimators = 20, learning_rate = 0.5, max_features = 2, max_depth = 2, random_state = 0
<i>k</i> -Nearest Neighbor	n_neighbors = 36, p = 1 (manhattan_distance)
XGBoost	max_bin = 255

## 5. Results

### 5.1. K-Means Clustering

Once the clustering model has been trained, we take the dataset and look for which element is the closest to the centroid of each of the two groups, these elements will be taken as the representatives of each of the groups. For each of these representatives, we verify the values of their features and category (Table 4). We identify that the category for each representative is different (benign/infected), and the values of some features are distant from each other.

With the clustering model created and trained, we apply this clustering model to the test dataset, obtaining the two clusters; with the help of the representative elements of each cluster, we identify which of the clusters corresponds to the benign sites and which to the infected ones. With this information, the cluster assigned to each element of the test dataset and the set of labels (*y*) corresponding to the test dataset, we calculate a classification report where we assume that the cluster assignment to each sample of the dataset corresponds to classification between the benign site and infected site. In this report, we obtain an accuracy = 0.9902 and precision = 0.988201.

**Table 4.** Values of the features of representatives of clusters.

Feature	Cluster 1 (y = 0)	Cluster 2 (y = 1)
C1	82.2145	8.2148
C2	73.1083	25.4516
C3	3.9767	6.4803
I/O Data Operations	32.7094	18.8409
I/O Data Bytes	121124.1414	39403.4743
Number of subprocesses	30	31
Time on processor	0.4967	0.1733
Disk Reading/sec	16.6267	4.9044
Disc Writing/sec	1.3781	0
Bytes Sent/sent	915.6518	210.0906
Received Bytes (HTTP)	12258.8161	3844.3897
Network packets sent	6.1794	1.6865
Network packets received	10.1138	3.6172
Pages Read/sec	0.6446	0.0665
Pages Input/sec	0	0
Page Errors/sec	13351.0499	1633.1676
Confirmed byte radius	28.9634	27.1788
Percentage of processor usage	17.7854	91.7852

### 5.2. Feature Selection

Table 5 presents the results of applying the methods for selecting features. It shows the method used and its selected features. We obtain 15 and 12 features with  $p$ -value and “RFECV wrapper” methods, respectively.

**Table 5.** Selected features.

Method	Selected Features
$p$ -value	C1, C2, C3, I/O Data Operations, I/O Data Bytes, Number of subprocesses, Time on processor, Disk Reading/sec, Received Bytes (HTTP), Network packets sent, Network packets received, Pages Input/sec, Page Errors/sec, Confirmed byte radius, Percentage of processor usage
anova scores 4 features	Percentage of processor usage, C1, C2, C3
anova scores 10 features	Percentage of processor usage, C1, C2, C3, Number of subprocesses, Confirmed byte radius, Page Errors/sec, I/O Data Operations, Disk Reading/sec, I/O Data Bytes
RFECV wrapper	Percentage of processor usage, Network packets sent, I/O Data Bytes, I/O Data Operations, Received Bytes (HTTP), Disc Writing/sec, Network packets received, C2, Bytes Sent/sent, Confirmed byte radius, Time on processor, C1

### 5.3. Cross-Validation

For the cross-validation and evaluation of the models, we employed some datasets, which are described in Table 6 and used in Tables 7 and 8.

**Table 6.** Datasets’ description.

Dataset Name	Description
All features dataset	complete dataset
15 features dataset	dataset of 15 features selected with $p$ -value
4 features dataset	dataset of 4 features selected with anova Scores method
10 features dataset	dataset of 10 features selected with anova Scores method
12 features dataset	dataset of 12 features selected with RFECV wrapper method

The results of the Cross-Validation of the different classification models with each of the datasets are presented in Table 7.

**Table 7.** Cross-validation models

<b>Logistic Regression</b>		
<b>Dataset</b>	<b>Mean Accuracy</b>	<b>Standard Deviation Accuracy</b>
All features dataset	0.9917	0.0020
15 features dataset	0.9911	0.0020
4 features dataset	0.9911	0.0023
10 features dataset	0.9912	0.0021
12 features dataset	0.9919	0.0018
<b>Decision Tree</b>		
All features dataset	0.9914	0.0018
15 features dataset	0.9921	0.0022
4 features dataset	0.991	0.0018
10 features dataset	0.991	0.0018
12 features dataset	0.9915	0.0020
<b>Random Forest</b>		
All features dataset	0.9944	0.0015
15 features dataset	0.9944	0.0014
4 features dataset	0.9906	0.0017
10 features dataset	0.9943	0.0015
12 features dataset	0.9947	0.0013
<b>Gradient Boosting</b>		
All features dataset	0.9926	0.0018
15 features dataset	0.9919	0.0019
4 features dataset	0.9905	0.0022
10 features dataset	0.9919	0.0020
12 features dataset	0.9932	0.0016
<b>k-Nearest Neighbor</b>		
All features dataset	0.9878	0.0018
15 features dataset	0.9878	0.0020
4 features dataset	0.9913	0.0021
10 features dataset	0.9891	0.0019
12 features dataset	0.9889	0.0019
<b>XGBoost</b>		
All features dataset	0.9947	0.0016
15 features dataset	0.9946	0.0016
4 features dataset	0.9908	0.0018
10 features dataset	0.9942	0.0017
12 features dataset	0.9950	0.0017

#### 5.4. Model Selection and Evaluation

Table 8 shows the main metrics obtained with the classification models.

**Table 8.** Results of evaluating the models.

<b>SAE + DDNN [14]</b>							
benign				malign			
Dataset	Accuracy	Precision	Recall	F1-score	Precision	Recall	F1-score
All features dataset	0.9893	0.9941	0.9911	0.9926	0.9823	0.9712	0.9891
<b>Logistic regression</b>							
benign				malign			
Dataset	Accuracy	Precision	Recall	F1-score	Precision	Recall	F1-score
All features dataset	0.9937	0.9957	0.9957	0.9957	0.9884	0.9884	0.9884
15 features dataset	0.9937	0.9957	0.9957	0.9957	0.9884	0.9884	0.9884
4 features dataset	0.9918	0.9946	0.9941	0.9944	0.984	0.9854	0.9847
10 features dataset	0.9925	0.9957	0.9941	0.9949	0.9841	0.9884	0.9862
12 features dataset	0.9941	0.9957	0.9962	0.996	0.9898	0.884	0.9891

Table 8. Cont.

SAE + DDNN [14]							
Decision Tree							
benign				malign			
Dataset	Accuracy	Precision	Recall	F1-score	Precision	Recall	F1-score
All features dataset	0.9902	0.9904	0.9962	0.9933	0.9896	0.9738	0.9817
15 features dataset	0.9918	0.993	0.9957	0.9944	0.9883	0.9811	0.9847
4 features dataset	0.9906	0.9925	0.9946	0.9936	0.9854	0.9796	0.9825
10 features dataset	0.9906	0.9946	0.9925	0.9935	0.9797	0.9854	0.9826
12 features dataset	0.9918	0.993	0.9957	0.9944	0.9883	0.9811	0.9847
Random Forest							
benign				malign			
Dataset	Accuracy	Precision	Recall	F1-score	Precision	Recall	F1-score
All features dataset	0.9957	0.9957	0.9984	0.997	0.9956	0.9884	0.992
15 features dataset	0.9953	0.9962	0.9973	0.9968	0.9927	0.9898	0.9913
4 features dataset	0.9906	0.9925	0.9946	0.9936	0.9854	0.9796	0.9825
10 features dataset	0.9949	0.9952	0.9978	0.9965	0.9941	0.9869	0.9905
12 features dataset	0.9949	0.9952	0.9978	0.9965	0.9941	0.9869	0.9905
Gradient Boosting							
benign				malign			
Dataset	Accuracy	Precision	Recall	F1-score	Precision	Recall	F1-score
All features dataset	0.9937	0.9957	0.9957	0.9957	0.9884	0.9884	0.9884
15 features dataset	0.9929	0.9952	0.9952	0.9952	0.9869	0.9869	0.9869
4 features dataset	0.991	0.9935	0.9941	0.9938	0.984	0.9825	0.9832
10 features dataset	0.9925	0.9946	0.9952	0.9949	0.9869	0.9854	0.9862
12 features dataset	0.9929	0.9946	0.9957	0.9952	0.9883	0.9854	0.9869
<i>k</i> -Nearest Neighbor							
benign				malign			
Dataset	Accuracy	Precision	Recall	F1-score	Precision	Recall	F1-score
All features dataset	0.9906	0.9904	0.9968	0.9936	0.9911	0.9738	0.9824
15 features dataset	0.9906	0.9909	0.9962	0.9936	0.9897	0.9753	0.9824
4 features dataset	0.9921	0.9946	0.9946	0.9946	0.9854	0.9854	0.9854
10 features dataset	0.9894	0.9909	0.9946	0.9928	0.9853	0.9753	0.9802
12 features dataset	0.9902	0.992	0.9946	0.9933	0.9853	0.9782	0.9817
XGBoost							
benign				malign			
Dataset	Accuracy	Precision	Recall	F1-score	Precision	Recall	F1-score
All features dataset	0.9965	0.9973	0.9978	0.9976	0.9942	0.9927	0.9934
15 features dataset	0.9965	0.9973	0.9978	0.9976	0.9942	0.9927	0.9934
4 features dataset	0.9910	0.9925	0.9952	0.9938	0.9868	0.9796	0.9832
10 features dataset	0.9965	0.9973	0.9978	0.9976	0.9942	0.9927	0.9934
12 features dataset	0.9965	0.9973	0.9978	0.9976	0.9942	0.9927	0.9934

We take the simple model Logistic Regression and the advanced model XGBoost as a reference and search for the features with the most importance in the classification process. As observed in Figure 5, positive coefficients indicate that the event (malign) is more likely

at that level of the predictor than at the reference level. Negative coefficients indicate that the event (malign) is less likely at that level of the predictor than at the reference level. It can be observed that the features' Percentage of processor usage or Network packets sent indicate that the event is more likely, while the variables C1 or Confirmed byte radius suggest that the event is much less likely. As observed in Figure 6, we can notice that the variables C1, C2, Percentage of processor usage, and Disc Writing/sec are the most important in the classification process of XGBoost model.

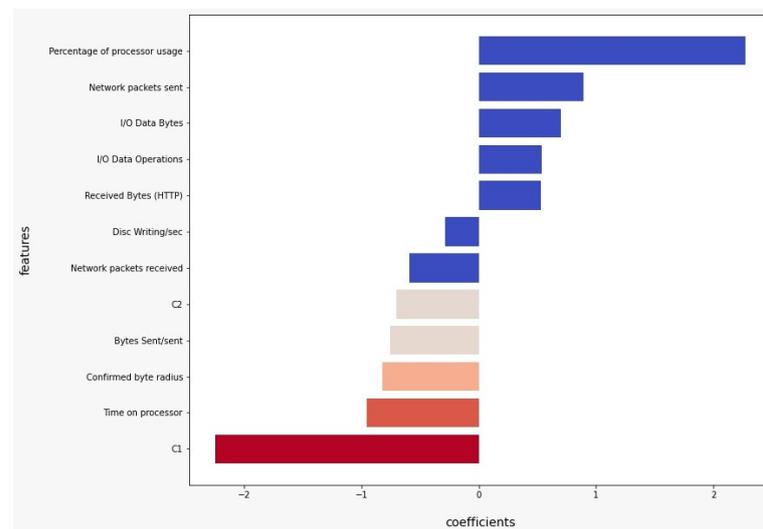


Figure 5. Feature importance logistic regression model.

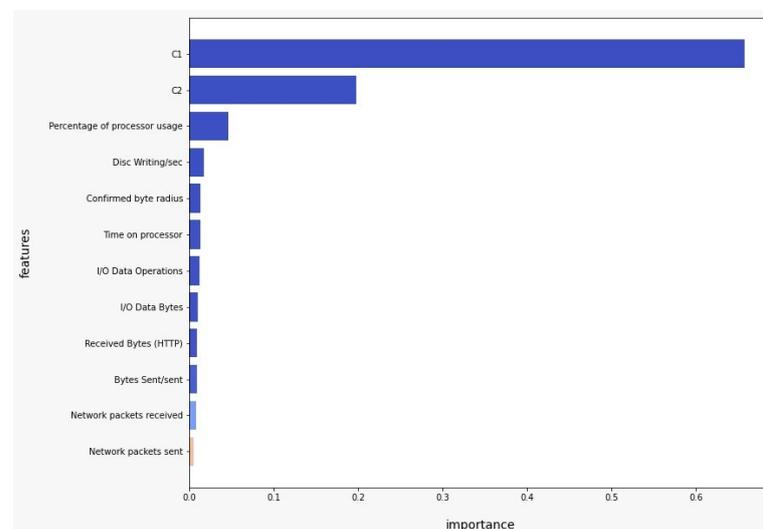


Figure 6. Feature importance XGBoost Model.

## 6. Integration Scenario

Taking the ideas described in [50–52], we can think of implementing a hybrid, lightweight, usable, privacy-preserving mechanism added to a web browser for blocking websites that potentially may be infected by Cryptojacking. The envisioned approach exploits the blacklisting technique (widely used in this field) and a machine learning classifier to classify websites as benign or malign. The output from the classifier allows for updating the blacklists used to filter/block blacklisted websites. Additionally, this approach can be enhanced by introducing an ML-Based model to detect JavaScript malicious code inserted in websites or content shared with the user. To improve the usability of the mechanism, live alerts may be generated for the users for providing them with a comprehensive awareness and full control of potential cryptojacking threats.

## 7. Conclusions

In this work, we explored six Machine Learning models for Detecting cryptojacking on Websites. Our exploration started with a simple model as Logistic Regression, and then moved to more advanced algorithms in terms of tabular data classification, such as XGBoost, Decision Trees, Random Forest, Gradient Boosting, and  $k$ -Nearest Neighbor models. Furthermore, various feature selection methods were used, such as those based on statistical methods, e.g., Test Anova, and other methods called Wrappers, in order not only to reduce the complexity of the built models but also to know the features with greater predictive power.

From our results, we observed the following.

1. With 12 of the 18 features obtained with the RFECV method, an accuracy similar to that of other works [14–17,53] based on Deep Learning techniques was reached. Even, as observed in Table 7, with a dataset of only 4 features, an accuracy of 99.11% was obtained using Logistic Regression, and an accuracy of 99.13 was obtained with  $k$ -Nearest Neighbor.
2. The most relevant features in the case of Logistic Regression were C1, Percentage of processor usage, I/O Data Bytes and Network packets sent, while the most important features in the case of XGBoost were Percentage of processor usage, Network packets sent, Time on processor and C1.

We conclude that by using simple models such as Logistic Regression, Decision Tree, Random Forest, Gradient Boosting and  $k$ -Nearest Neighbor models, we can build ML-based classification components with a success rate similar to or greater than that of advanced algorithms such as XGBoost and even those of other works based on Deep Learning. Additionally, the simplicity of these models help the evaluator interpret the results and know the inner-working of these models in comparison with other advanced models based on Deep Learning (which are regarded as black boxes).

**Author Contributions:** Conceptualization, F.A.A.-N., D.P.Á., R.V.-P., A.L.S.O. and L.J.G.V.; methodology, F.A.A.-N., D.P.Á., R.V.-P., A.L.S.O. and L.J.G.V.; software, F.A.A.-N. and D.P.Á.; validation, F.A.A.-N., D.P.Á. and R.V.-P.; formal analysis, F.A.A.-N., D.P.Á. and R.V.-P.; investigation, F.A.A.-N., D.P.Á., R.V.-P., A.L.S.O. and L.J.G.V.; resources, F.A.A.-N., R.V.-P., A.L.S.O. and L.J.G.V.; data curation, F.A.A.-N. and D.P.Á.; writing—original draft preparation, F.A.A.-N. and R.V.-P.; writing—review and editing, F.A.A.-N. and R.V.-P.; visualization, F.A.A.-N. and D.P.Á.; supervision, A.L.S.O., R.V.-P. and L.J.G.V.; project administration, A.L.S.O., R.V.-P. and L.J.G.V.; funding acquisition, F.A.A.-N. and R.V.-P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was carried out with the support of the program “Convocatoria 779 de 2017-Doctorado Nacional Convocatoria para la Formación de Capital Humano de Alto Nivel para el Departamento de Boyacá–2017” and the Universidad del Norte.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

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

## References

1. Carlin, D.; Burgess, J.; O’Kane, P.; Sezer, S. You Could Be Mine(d): The Rise of Cryptojacking. *IEEE Secur. Priv.* **2020**, *18*, 16–22. [[CrossRef](#)]
2. Aponte, F.; Gutierrez, L.; Pineda, M.; Meriño, I.; Salazar, A.; Wightman, P. Cluster-Based Classification of Blockchain Consensus Algorithms. *IEEE Lat. Am. Trans.* **2021**, *19*, 688–696. [[CrossRef](#)]
3. Tayyab, U.e.H.; Khan, F.B.; Durad, M.H.; Khan, A.; Lee, Y.S. A Survey of the Recent Trends in Deep Learning Based Malware Detection. *J. Cybersecur. Priv.* **2022**, *2*, 800–829. [[CrossRef](#)]
4. Wu, M.H.; Lai, Y.J.; Hwang, Y.L.; Chang, T.C.; Hsu, F.H. MinerGuard: A Solution to Detect Browser-Based Cryptocurrency Mining through Machine Learning. *Appl. Sci.* **2022**, *12*, 9838. [[CrossRef](#)]

5. Bijmans, H.L.; Booij, T.M.; Doerr, C. Inadvertently Making Cyber Criminals Rich: A Comprehensive Study of Cryptojacking Campaigns at Internet Scale. In Proceedings of the 28th USENIX Security Symposium (USENIX Security 19), Santa Clara, CA, USA, 14 August 2019; pp. 1627–1644.
6. Aponte-Novoa, F.A.; Orozco, A.L.S.; Villanueva-Polanco, R.; Wightman, P. The 51% Attack on Blockchains: A Mining Behavior Study. *IEEE Access* **2021**, *9*, 140549–140564. [CrossRef]
7. Aponte-Novoa, F.A.; Villanueva-Polanco, R. On Proof-of-Accuracy Consensus Protocols. *Mathematics* **2022**, *10*, 2540. [CrossRef]
8. Tekiner, E.; Acar, A.; Uluagac, A.S.; Kirda, E.; Selcuk, A.A. SoK: Cryptojacking Malware. In Proceedings of the 2021 IEEE European Symposium on Security and Privacy (EuroS P), Virtual, 6–10 September 2021; pp. 120–139. . 2021.00019. [CrossRef]
9. Coinhive—Monero Mining Club. 2022. Available online: <http://web.archive.org/web/20190130232758/https://coinhive.com/> (accessed on 24 May 2022).
10. CryptoLoot—Earn More from Your Traffic. 2022. Available online: <https://crypto-loot.org/> (accessed on 24 May 2022).
11. Cryptojacking Campaign Impacts Nearly 1.500 Websites—Noticias de seguridad—Trend Micro ES. 2022. Available online: <https://www.trendmicro.com/vinfo/es/security/news/cybercrime-and-digital-threats/cryptojacking-campaign-impacts-nearly-1-500-websites> (accessed on 3 June 2022).
12. Cryptojacking: What Is It and What You Need To Know in 2022. 2022. Available online: <https://www.ensighten.com/blog/cryptojacking-what-is-it-and-what-you-need-to-know-in-2022> (accessed on 4 June 2022).
13. Ying, Q.; Yu, Y.; Tian, D.; Jia, X.; Ma, R.; Hu, C. CJSpector: A Novel Cryptojacking Detection Method Using Hardware Trace and Deep Learning. *J. Grid Comput.* **2022**, *20*, 31. [CrossRef]
14. Hernandez-Suarez, A.; Sanchez-Perez, G.; Toscano-Medina, L.K.; Olivares-Mercado, J.; Portillo-Portilo, J.; Avalos, J.G.; García Vilalba, L.J. Detecting Cryptojacking Web Threats: An Approach with Autoencoders and Deep Dense Neural Networks. *Appl. Sci.* **2022**, *12*, 3234. [CrossRef]
15. Naseem, F.; Aris, A.; Babun, L.; Tekiner, E.; Uluagac, S. MINOS: A lightweight real-time cryptojacking detection system. In Proceedings of the 28th Annual Network and Distributed System Security Symposium, NDSS, San Diego, CA, USA, 21–25 February 2021.
16. Om Kumar, C.; Sathia Bham, P.R. Detecting and confronting flash attacks from IoT botnets. *J. Supercomput.* **2019**, *75*, 8312–8338. [CrossRef]
17. Liu, J.; Zhao, Z.; Cui, X.; Wang, Z.; Liu, Q. A novel approach for detecting browser-based silent miner. In Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), Guangzhou, China, 18–21 June 2018; pp. 490–497.
18. Sivaraju, S. An Insight into Deep Learning based Cryptojacking Detection Model. *J. Trends Comput. Sci. Smart Technol.* **2022**, *4*, 175–184. [CrossRef]
19. Petrov, I.; Invernizzi, L.; Bursztein, E. CoinPolice: Detecting Hidden Cryptojacking Attacks with Neural Networks. *arXiv* **2020**, arXiv:2006.10861.
20. YouTube Ads have been Secretly Mining Cryptocurrency | Mashable. 2022. Available online: <https://mashable.com/article/coinhive-youtube-google-doubleclick> (accessed on 19 May 2022).
21. ¿Qué es el Cryptojacking y cómo Funciona? 2022. Available online: <https://www.kaspersky.es/resource-center/definitions/what-is-cryptojacking> (accessed on 19 May 2022).
22. Russian Scientists Arrested for Crypto Mining at Nuclear Lab—CoinDesk. 2021. Available online: <https://www.coindesk.com/markets/2018/02/09/russian-scientists-arrested-for-crypto-mining-at-nuclear-lab/> (accessed on 18 May 2022).
23. A Crypto-Mining Botnet has been Hijacking MSSQL Servers for Almost Two Years | ZDNet. 2022. Available online: <https://www.zdnet.com/article/a-crypto-mining-botnet-has-been-hijacking-mssql-servers-for-almost-two-years/> (accessed on 18 May 2022).
24. Zoomed In: A Look into a Coinminer Bundled with Zoom Installer. 2022. Available online: [https://www.trendmicro.com/en\\_us/research/20/d/zoomed-in-a-look-into-a-coinminer-bundled-with-zoom-installer.html](https://www.trendmicro.com/en_us/research/20/d/zoomed-in-a-look-into-a-coinminer-bundled-with-zoom-installer.html) (accessed on 18 May 2022).
25. Nintendo Switch Game Pulled Over Cryptojacking Concerns. 2022. Available online: <https://bitcoinist.com/nintendo-switch-game-pulled-over-cryptojacking-concerns/> (accessed on 18 May 2022).
26. Over 200,000 MikroTik Routers Compromised in Cryptojacking Campaign. 2022. Available online: <https://www.trendmicro.com/vinfo/ru/security/news/cybercrime-and-digital-threats/over-200-000-mikrotik-routers-compromised-in-cryptojacking-campaign> (accessed on 19 May 2022).
27. Malik, A.W.; Anwar, Z. Do Charging Stations Benefit from Cryptojacking? A Novel Framework for Its Financial Impact Analysis on Electric Vehicles. *Energies* **2022**, *15*, 5773. [CrossRef]
28. Detected Cryptojacking Prompts Microsoft to Remove Eight Free Apps from Microsoft Store. 2022. Available online: <https://cointelegraph.com/news/detected-cryptojacking-prompts-microsoft-to-remove-eight-free-apps-from-microsoft-store> (accessed on 19 May 2022).
29. Several Cryptojacking Apps Found on Microsoft Store | Symantec Blogs. 2022. Available online: <https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/cryptojacking-apps-microsoft-store> (accessed on 19 May 2022).
30. Tesla’s Cloud Hacked, Used to Mine Cryptocurrency. 2022. Available online: <https://gizmodo.com/teslas-cloud-hacked-used-to-mine-cryptocurrency-1823155247> (accessed on 24 May 2022).
31. Bug Hunter Finds Cryptocurrency-Mining Botnet on DOD Network | ZDNet. 2022. Available online: <https://www.zdnet.com/article/bug-hunter-finds-cryptocurrency-mining-botnet-on-dod-network/> (accessed on 19 May 2022).

32. Crypto Mining Botnet Found on Defense Department web Server—Decrypt. 2022. Available online: <https://decrypt.co/18738/crypto-mining-botnet-found-on-defense-department-web-server> (accessed on 19 May 2022).
33. Man Fined \$7.000 for Using Russian Supercomputer to Mine Bitcoin—Decrypt. 2022. Available online: <https://decrypt.co/9751/man-fined-for-using-russian-supercomputer-to-mine-crypto> (accessed on 19 May 2022).
34. Aponte-Novoa, F.A.; Daniel, P.A.; Villanueva-Polanco, R. Detecting Cryptojacking On Web Sites Use Classifiers. 2022. Available online: [https://github.com/faan03/Detecting\\_Cryptojacking\\_On\\_Web\\_Sites\\_Use\\_Classifiers](https://github.com/faan03/Detecting_Cryptojacking_On_Web_Sites_Use_Classifiers) (accessed on 19 May 2022).
35. Arthur, D.; Vassilvitskii, S. k-means++: The advantages of careful seeding. In Proceedings of the SODA '07, New Orleans, LA, USA, 7–9 January 2007.
36. Prashanth, S.K.; Shitharth, S.; Praveen Kumar, B.; Subedha, V.; Sangeetha, K. Optimal Feature Selection Based on Evolutionary Algorithm for Intrusion Detection. *SN Comput. Sci.* **2022**, *3*, 439. [CrossRef]
37. caretSBF: Selection By Filtering (SBF) Helper Functions in Caret: Classification and Regression Training. 2022. Available online: <https://rdrr.io/cran/caret/man/caretSBF.html> (accessed on 3 June 2022).
38. Recursive Feature Elimination (RFE) for Feature Selection in Python. 2022. Available online: <https://machinelearningmastery.com/rfe-feature-selection-in-python/> (accessed on 3 June 2022).
39. Feng, J.; Xu, H.; Mannor, S.; Yan, S. Robust Logistic Regression and Classification. In *Proceedings of the Advances in Neural Information Processing Systems*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K., Eds.; Curran Associates, Inc.: Montreal, QC, Canada, 2014; Volume 27.
40. Suthaharan, S. Chapter 6—A Cognitive Random Forest: An Intra- and Intercognitive Computing for Big Data Classification Under Cune Condition. In *Cognitive Computing: Theory and Applications*; Gudivada, V.N., Raghavan, V.V., Govindaraju, V., Rao, C., Eds.; Elsevier: Amsterdam, The Netherlands, 2016; Volume 35, pp. 207–227. [CrossRef]
41. Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. *Classification and Regression Trees*; Routledge: New York, NY, USA, 2017.
42. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
43. Li, P. Robust LogitBoost and Adaptive Base Class (ABC) LogitBoost. 2012. Available online: <https://doi.org/10.48550/ARXIV.1203.3491> (accessed on 20 October 2022).
44. Richardson, M.; Dominowska, E.; Ragno, R. Predicting Clicks: Estimating the Click-through Rate for New Ads. In Proceedings of the 16th International Conference on World Wide Web, WWW '07, Banff, AB, Canada, 8–12 May 2007; Association for Computing Machinery: New York, NY, USA, 2007; pp. 521–530. [CrossRef]
45. Burges, C.J. From ranknet to lambdarank to lambdamart: An overview. *Learning* **2010**, *11*, 81.
46. Dasarathy, B.V. Nearest neighbor (NN) norms: NN pattern classification techniques. *IEEE Comput. Soc. Tutor.* **1991**, 447.
47. Ripley, B.D. *Pattern Recognition and Neural Networks*; Cambridge University Press: Cambridge, UK, 2007.
48. Dreiseitl, S.; Ohno-Machado, L. Logistic regression and artificial neural network classification models: A methodology review. *J. Biomed. Inform.* **2002**, *35*, 352–359. [CrossRef]
49. Omer, Z.M.; Shareef, H. Comparison of Decision Tree based Ensemble Methods for Prediction of Photovoltaic Maximum Current. *Energy Convers. Manag.* **2022**, *16*, 100333. [CrossRef]
50. Cozza, F.; Guarino, A.; Isernia, F.; Malandrino, D.; Rapuano, A.; Schiavone, R.; Zaccagnino, R. Hybrid and lightweight detection of third party tracking: Design, implementation, and evaluation. *Comput. Netw.* **2020**, *167*, 106993. [CrossRef]
51. Guarino, A.; Malandrino, D.; Zaccagnino, R. An automatic mechanism to provide privacy awareness and control over unwittingly dissemination of online private information. *Comput. Netw.* **2022**, *202*, 108614. [CrossRef]
52. Ikram, M.; Asghar, H.J.; Kâafar, M.A.; Krishnamurthy, B.; Mahanti, A. Towards Seamless Tracking-Free Web: Improved Detection of Trackers via One-class Learning. *arXiv* **2016**, arXiv:1603.06289.
53. Gomes, F.; Correia, M. Cryptojacking detection with cpu usage metrics. In Proceedings of the 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 24–27 November 2020; pp. 1–10.