

Detecting Ransomware Addresses on the Bitcoin Blockchain using Random Forests and Self Organizing Maps

HarvardX PH125.9x Final Capstone CYO Project

Kaylee Robert Tejada

10/31/2021

Abstract

Ransomware is a persistent and growing threat in the world of cybersecurity. A specific area of focus involves detecting and tracking payments made to ransomware operators. While many attempts towards this goal have not made use of sophisticated machine learning methods, even those that have often result in models with poor specificity or other performance issues. A two-step method is developed to address the issue of false positives and improve on previous results.

Contents

| | |
|--|----|
| Introduction | 3 |
| Data | 4 |
| Goal | 4 |
| Outline of Steps Taken (refine this as steps are written up...) | 4 |
| Data Analysis | 5 |
| Hardware Specification | 5 |
| Data Preparation | 5 |
| Exploration and Visualization (Chunk #2, do this part last....) | 5 |
| Insights Gained from Exploration | 7 |
| Modeling approach (chunk #3, needs rewriting of text parts only) | 8 |
| Method Part 1: Binary Random Forests to isolate ransomware addresses first. | 8 |
| Method Part 2: Categorical SOMs to categorize predicted ransomware addresses. | 9 |
| Clustering Visualizations: K-means clustering | 10 |
| Results & Performance (chunk #4, write up after chunk #3 is done) | 10 |
| Results | 10 |
| Performance | 11 |
| Summary | 11 |
| Comparison to original paper and impact of findings | 11 |
| Limitations | 11 |
| Future Work | 11 |
| Conclusions | 11 |
| References | 11 |
| Appendix: | 12 |
| Categorical SOM ransomware family prediction table and confusion matrix - detailed | 12 |

Introduction

Ransomware attacks have gained the attention of security professionals, law enforcement, and financial regulatory officials.^[1] The pseudo-anonymous Bitcoin network provides a convenient method for ransomware attackers to accept payments without revealing their identity or location. The victims (usually hospitals or other large organizations) come to find that much if not all of their important organizational data have been encrypted with a secret key by an unknown attacker. They are instructed to make a payment to a specific Bitcoin address before a certain deadline to have their data decrypted, otherwise the data will be deleted.

The legal and financial implications of ransomware attacks are not of concern for the purpose of this analysis. Many parties are interested in tracking illicit activity (such as ransomware payments) around the Bitcoin blockchain as soon as possible to minimize financial losses. Daniel Goldsmith explains some of the reasons and methods of blockchain analysis at Chainalysis.com.^[2] A ransomware attack could be perpetrated on an illegal darknet market site, for example. The news of such an attack might not be published at all, let alone in popular media. By analyzing the transaction record with a blockchain explorer such as BTC.com, suspicious activity could be flagged in real time given a sufficiently robust model. It may, in fact, be the first public notice of such an event. Any suspicious addresses could then be blacklisted or banned from using other services.

Lists of known ransomware payment addresses have been compiled and analyzed using various methods. One well known paper entitled “BitcoinHeist: Topological Data Analysis for Ransomware Detection on the Bitcoin Blockchain”^[3] will be the source of our data set and the baseline to which we will compare our results. In that paper, Akcora, et al. use Topological Data Analysis (TDA) to classify addresses on the Bitcoin blockchain into one of 29 known ransomware address groups. Addresses with no known ransomware associations are classified as “white”. The blockchain is then considered as a heterogeneous Directed Acyclic Graph (DAG) with two types of nodes describing *addresses* and *transactions*. Edges are formed between the nodes when a transaction can be associated with a particular address.

Addresses on the Bitcoin network may appear many times, with different inputs and outputs each time. The Bitcoin network data has been divided into 24-hour time intervals with the UTC-6 timezone as a reference. Speed is defined as the number of blocks the coin appears in during a 24-hour period and provides information on how quickly a coin moves through the network. Speed can be an indicator of money laundering or coin mixing, as normal payments only involve a limited number of addresses in a 24 hour period, and thus have lower speeds when compared to “mixed” coins. The temporal data can also help distinguish transactions by geolocation, as criminal transactions tend to cluster in time.

With the graph defined as such, the following six numerical features^[2] are associated with a given address:

- 1) *Income* - the total amount of coins sent to an address (decimal value with 8 decimal places)
- 2) *Neighbors* - the number of transactions that have this address as one of its output addresses (integer)
- 3) *Weight* - the sum of fraction of coins that reach this address from address that do not have any other inputs within the 24-hour window, which are referred to as “starter transactions” (decimal value)
- 4) *Length* - the number of non-starter transactions on its longest chain, where a chain is defined as an acyclic directed path originating from any starter transaction and ending at the address in question (integer)
- 5) *Count* - The number of starter addresses connected to this address through a chain (integer)
- 6) *Loop* - The number of starter addresses connected to this address by more than one path (integer)

These variables are defined rather abstractly, viewing the blockchain as a topological graph with nodes and edges. The rationale behind this approach is to quantify specific transaction patterns. Akcora^[3] gives a thorough explanation in the original paper of how and why these features were chosen. We shall treat the features as general numerical variables and will not seek to justify their definitions. Several machine learning methods will be applied to the original data set from the paper by Akcora^[3], and the results will be compared.

Data

This data set was discovered while exploring the UCI Machine Learning Repository^[4] as suggested in the project instructions. The author of this report, interested in Bitcoin and other cryptocurrencies since (unsuccessfully) mining them on an ASUS netbook in rural Peru in late 2010, used *cryptocurrency* as a preliminary search term. This brought up a single data set entitled “BitcoinHeist: Ransomware Address Data Set”. The data set was downloaded and the exploration began.

A summary of the data set tells the range of values and size of the sample.

| address | year | day | length | weight | count | looped | neighbors | income | label |
|----------------|--------------|---------------|---------------|-----------------|-----------------|---------------|-----------------|-------------------|----------------|
| Length:2916697 | Min. :2011 | Min. : 1.0 | Min. : 0.00 | Min. : 0.0000 | Min. : 1.0 | Min. : 0.0 | Min. : 1.000 | Min. :3.000e+07 | Length:2916697 |
| Class | 1st Qu.:2013 | 1st Qu.: 92.0 | 1st Qu.: 2.00 | 1st Qu.: 0.0215 | 1st Qu.: 1.0 | 1st Qu.: 0.0 | 1st Qu.: 1.000 | 1st Qu.:7.429e+07 | Class |
| :character | Median | Median :181.0 | Median : 8.00 | Median : 0.2500 | Median : 1.0 | Median : 0.0 | Median : 2.000 | Median :2.000e+08 | :character |
| Mode | :2014 | Mean | Mean :181.5 | Mean : 45.01 | Mean : 0.5455 | Mean : 238.5 | Mean : 2.207 | Mean :4.465e+09 | Mode |
| :character | NA | 3rd Qu.:2016 | 3rd Qu.:271.0 | 3rd Qu.:108.00 | 3rd Qu.: 0.8819 | 3rd Qu.: 56.0 | 3rd Qu.: 2.000 | 3rd Qu.:9.940e+08 | :character |
| NA | Max. :2018 | Max. :365.0 | Max. :144.00 | Max. :1943.7488 | Max. :14497.0 | Max. :14496.0 | Max. :12920.000 | Max. :4.996e+13 | NA |

A listing of the first ten rows provides a sample of the features associated with each observation.

| address | year | day | length | weight | count | looped | neighbors | income | label |
|------------------------------------|------|-----|--------|-----------|-------|--------|-----------|-----------|-----------------|
| 111K8kZAEEnJg245r2cM6y9zgJGHZtJPy6 | 2017 | 11 | 18 | 0.0083333 | 1 | 0 | 2 | 100050000 | princetonCerber |
| 1123pJv8jzeFQaCV4w644pzQJzVWay2zcA | 2016 | 132 | 44 | 0.0002441 | 1 | 0 | 1 | 100000000 | princetonLocky |
| 112536im7hy6wtKbpH1qYDWtTyMRAcA2p7 | 2016 | 246 | 0 | 1.0000000 | 1 | 0 | 2 | 200000000 | princetonCerber |
| 1126eDRw2wqSkWosjTCre8cjjQW8sSeWH7 | 2016 | 322 | 72 | 0.0039063 | 1 | 0 | 2 | 71200000 | princetonCerber |
| 1129TSjKtx65E35GiUo4AYVeyo48tubrGX | 2016 | 238 | 144 | 0.0728484 | 456 | 0 | 1 | 200000000 | princetonLocky |
| 112AmFATxzhSpvtz1hfp3Zrw3BG276pc | 2016 | 96 | 144 | 0.0846140 | 2821 | 0 | 1 | 50000000 | princetonLocky |

This data set has 2,916,697 observations of ten features associated with a sample of transactions from the Bitcoin blockchain. The ten features include *address* as a unique identifier, the six features defined previously (*income*, *neighbors*, *weight*, *length*, *count*, *loop*), two temporal features in the form of *year* and *day* (of the year as 1-365), and a categorical feature called *label* that categorizes each address as either “white” (meaning not connected to any ransomware activity), or one of 29 known ransomware groups as identified by three independent ransomware analysis teams (Montreal, Princeton, and Padua)^[3].

The original research team downloaded and parsed the entire Bitcoin transaction graph from 2009 January to 2018 December. Based on a 24 hour time interval, daily transactions on the network were extracted and the Bitcoin graph was formed. Network edges that transferred less than \$0.3 were filtered out since ransom amounts are rarely below this threshold. Ransomware addresses are taken from three widely adopted studies: Montreal, Princeton and Padua. “White” Bitcoin addresses were capped at one thousand per day while the entire network has up to 800,000 addresses daily.^[5]

Goal

The goal of this project is to apply different machine learning algorithms to the same data set used in the original paper to produce an acceptable predictive model for categorizing ransomware addresses correctly. Improving on the results of the original paper in some way, while not strictly necessary for the purposes of the project, would be a notable sign of success.

Outline of Steps Taken (refine this as steps are written up...)

- 1) Analyze data set numerically and visually. Notice any pattern, look for insights.
- 2) Binary classification using Random Forests.
- 3) Binary classification using Self Organizing Maps.
- 4) Categorical classification using Self Organizing Maps.

- 5) Two step method using Random Forests and Self Organizing Maps.
- 6) Visualize clustering to analyze results further.
- 7) Generate Confusion Matrix to quantify results.

Data Analysis

Hardware Specification

All of the analysis in this report was conducted on a single laptop computer, a Lenovo Yoga S1 from late 2013 with the following specs:

- CPU: Intel i7-4600U @ 3.300GHz (4th Gen quad-core i7)
- RAM: 8217MB DDR3L @ 1600 MHz (8 GB)
- OS: Slackware64-current (15.0 RC1) x86_64-slackware-linux-gnu (64-bit GNU/Linux)
- R version 4.0.0 (2020-04-24) – “Arbor Day” (built from source using scripts from slackbuilds.org)
- RStudio Version 1.4.1106 “Tiger Daylily” (2389bc24, 2021-02-11) for CentOS 8 (converted using rpm2tgz)

Data Preparation

It is immediately apparent that this is a rather large data set. The usual practice of partitioning out eighty to ninety percent of the data for a training set results in a data set that is too large to process given the hardware available. For reasons that no longer apply, the original data set was first split in half with 50% reserved as “validation set” and the other 50% used as the “working set”. This working set was again split in half, to give a “training set” that was of a reasonable size to deal with. At this point the partitions were small enough to work with, so the sample partitions were not further refined. This is a potential area for later optimization. Careful sampling was carried out to ensure that the ransomware groups were represented in each sample.

#####

Exploration and Visualization (Chunk #2, do this part last...)

The ransomware addresses make up less than 2% of the overall data set. This presents a challenge as the target observations are sparse within the data set, especially when we consider that this is then divided into 29 subsets. In fact, some of the ransomware groups have only a single member, making categorization a dubious task. At least there are no missing values to worry about.

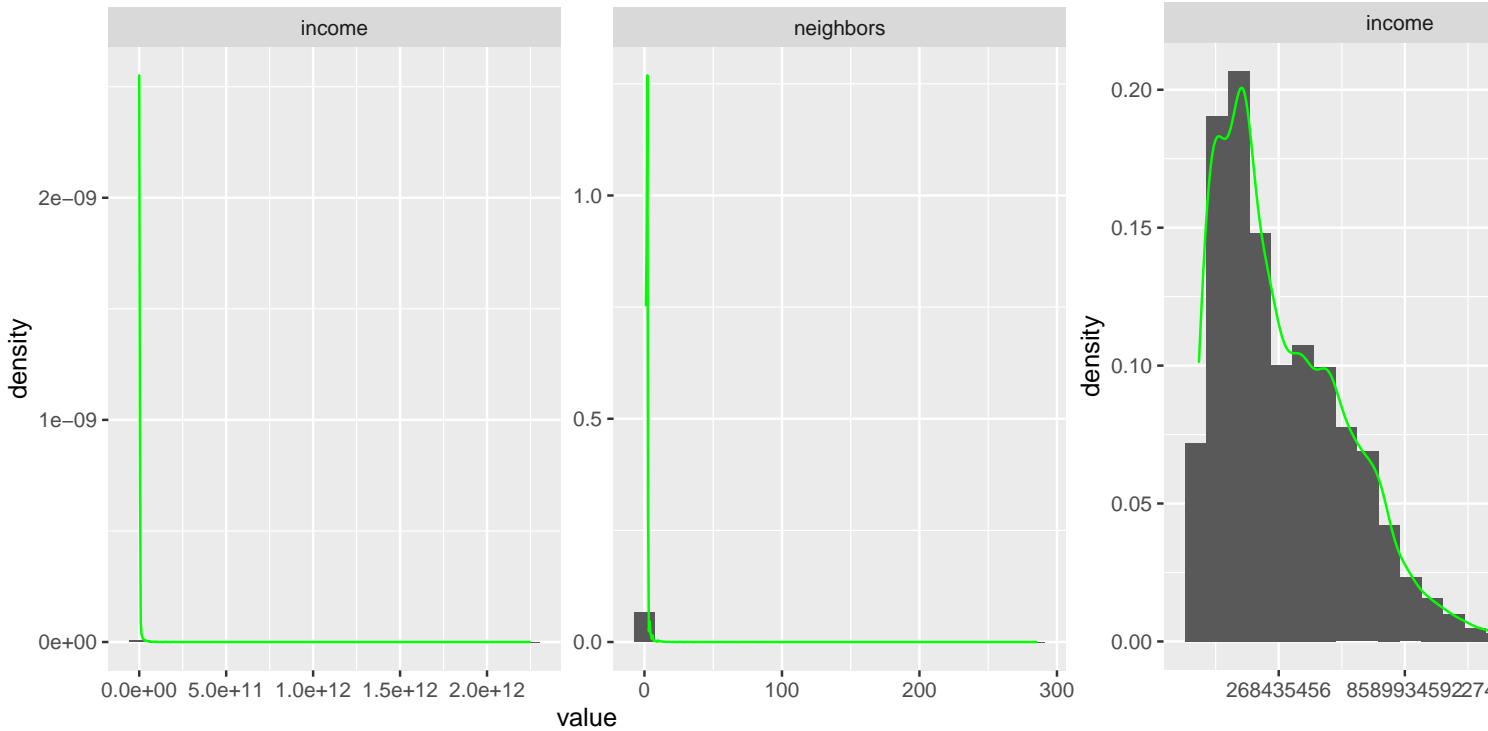
```
## The proportion of ransomware addresses in the original data set is 0.0141985951917529.
```

```
## The total number of NA or missing values in the original data set is 0.
```

Let’s take a look at the distribution of the different features. Note how skewed the non-temporal features are, some of them being bimodal:

Table 4: Ransomware group labels and frequency counts for full data set

| | x | | x |
|-----------------------------|------|---------------------|---------|
| montrealAPT | 11 | montrealRazy | 13 |
| montrealComradeCircle | 1 | montrealSam | 1 |
| montrealCryptConsole | 7 | montrealSamSam | 62 |
| montrealCryptXXX | 2419 | montrealVenusLocker | 7 |
| montrealCryptoLocker | 9315 | montrealWannaCry | 28 |
| montrealCryptoTorLocker2015 | 55 | montrealXLocker | 1 |
| montrealDMALocker | 251 | montrealXLockerv5.0 | 7 |
| montrealDMALockerv3 | 354 | montrealXTPLocker | 8 |
| montrealEDA2 | 6 | paduaCryptoWall | 12390 |
| montrealFlyper | 9 | paduaJigsaw | 2 |
| montrealGlobe | 32 | paduaKeRanger | 10 |
| montrealGlobeImposter | 55 | princetonCerber | 9223 |
| montrealGlobev3 | 34 | princetonLocky | 6625 |
| montrealJigSaw | 4 | white | 2875284 |
| montrealNoobCrypt | 483 | | |

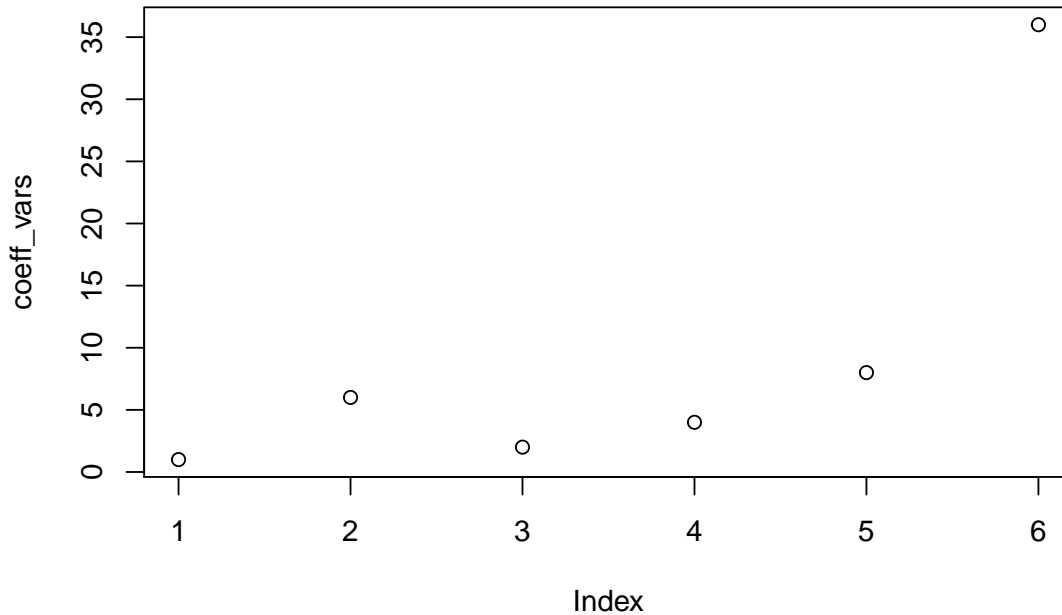


Now we can compare the relative spread of each feature by calculating the coefficient of variation for each column. Larger coefficients of variation indicate larger relative spread compared to other columns.

The features with the highest coefficients of variation are incomeneighbors, which will be used to train

| | x |
|--------|---|
| length | 1 |
| weight | 6 |
| count | 2 |

| | x |
|-----------|----|
| looped | 4 |
| neighbors | 8 |
| income | 36 |



From this, it appears that *income* has the widest range of variability, followed by *neighbors*. These are also the features that are most strongly skewed to the right, meaning that a few addresses have really high values for each of these features while the bulk of the data set has very low values for these numbers.

Now do the following (after filling in methods, results, and conclusions, since those are done already):

- 6) Break into groups somehow. Graph variables per group? Show how the variables are distributed for each ransomware group? Percent ransomware per each day of the week, for example. Is ransomware more prevalent on a particular day of the week? Break other numerical values into bins, and graph percentage per bin. Look for trends and correlations between groups/variables, and display them here. MORE OF THIS...

```
## [1] 0.01395349
```

```
#####
## Clean up and add text from here to end....
#####
```

Insights Gained from Exploration

From the previous visual and statistical exploration of the data, it becomes clear what the challenge is. Ransomware addresses are very sparse in the data set, making up less than 2% of the addresses. That small percentage is also further classified into 28 groups. Perhaps the original paper was a bit too ambitious in trying to categorize all the addresses into 29 categories, including the “white” addresses. To simplify our approach, we will categorize the addresses in a binary way, either “white” or “black”, where “black” signifies an association with ransomware transactions. Asking this as a “ransomware or not-ransomware” question allows for application of methods that are impractical otherwise.

Modeling approach (chunk #3, needs rewriting of text parts only)

Akcora et al. mention that they tried to model the data using a Random Forests method, but that the complexity of the data set lead to problems with that approach.[3] Switching to a binary perspective on the problem might alleviate some of that complexity, and is worth another look. The topological nature of the way the data set has been described numerically lead me to search for topological machine learning methods. Searching for *topo* in the documentation for the `caret` package [6] resulted in the entry for Self Organizing Maps, supplied by the `kohonen` package. The description at CRAN [7] was intriguing enough for me to investigate further.

Describe how you started with categorical SOMs, switched to binary SOMs, then applied `randomForest` to the binary problem, and was surprised with the results. Decided to re-apply categorical SOMs to black-only addresses, as predicted by the binary Random forest approach. The result is the following two-step approach, with the optional clustering visualizations at the end

Method Part 1: Binary Random Forests to isolate ransomware addresses first.

Here are the results for the test set.

| | black | white |
|-------|-------|-------|
| black | 101 | 0 |
| white | 3 | 7188 |

| | x |
|----------------|-----------|
| Accuracy | 0.9995886 |
| Kappa | 0.9851573 |
| AccuracyLower | 0.9987982 |
| AccuracyUpper | 0.9999151 |
| AccuracyNull | 0.9857378 |
| AccuracyPValue | 0.0000000 |
| McnemarPValue | 0.2482131 |

| | x |
|----------------------|-----------|
| Sensitivity | 0.9711538 |
| Specificity | 1.0000000 |
| Pos Pred Value | 1.0000000 |
| Neg Pred Value | 0.9995828 |
| Precision | 1.0000000 |
| Recall | 0.9711538 |
| F1 | 0.9853659 |
| Prevalence | 0.0142622 |
| Detection Rate | 0.0138508 |
| Detection Prevalence | 0.0138508 |
| Balanced Accuracy | 0.9855769 |

Here are the results for the full original set.

| | black | white |
|-------|-------|---------|
| black | 40809 | 0 |
| white | 604 | 2875284 |

| | x |
|----------------|-----------|
| Accuracy | 0.9997929 |
| Kappa | 0.9925490 |
| AccuracyLower | 0.9997757 |
| AccuracyUpper | 0.9998091 |
| AccuracyNull | 0.9858014 |
| AccuracyPValue | 0.0000000 |
| McNemarPValue | 0.0000000 |

| | x |
|----------------------|-----------|
| Sensitivity | 0.9854152 |
| Specificity | 1.0000000 |
| Pos Pred Value | 1.0000000 |
| Neg Pred Value | 0.9997900 |
| Precision | 1.0000000 |
| Recall | 0.9854152 |
| F1 | 0.9926540 |
| Prevalence | 0.0141986 |
| Detection Rate | 0.0139915 |
| Detection Prevalence | 0.0139915 |
| Balanced Accuracy | 0.9927076 |

Method Part 2: Categorical SOMs to categorize predicted ransomware addresses.

Now we train a new model after throwing away all “white” addresses.

A specific method was used to select the optimal grid size. Cite source here and explain options, and why you chose the one you did.

A grid size of 27 has been chosen.

Here is a summary of the results for the categorization of black addresses into ransomware families. For the full table of predictions and statistics, see the Appendix.

| | x |
|----------------|-----------|
| Accuracy | 0.9993626 |
| Kappa | 0.9991765 |
| AccuracyLower | 0.9989103 |
| AccuracyUpper | 0.9996606 |
| AccuracyNull | 0.3050108 |
| AccuracyPValue | 0.0000000 |
| McNemarPValue | NaN |

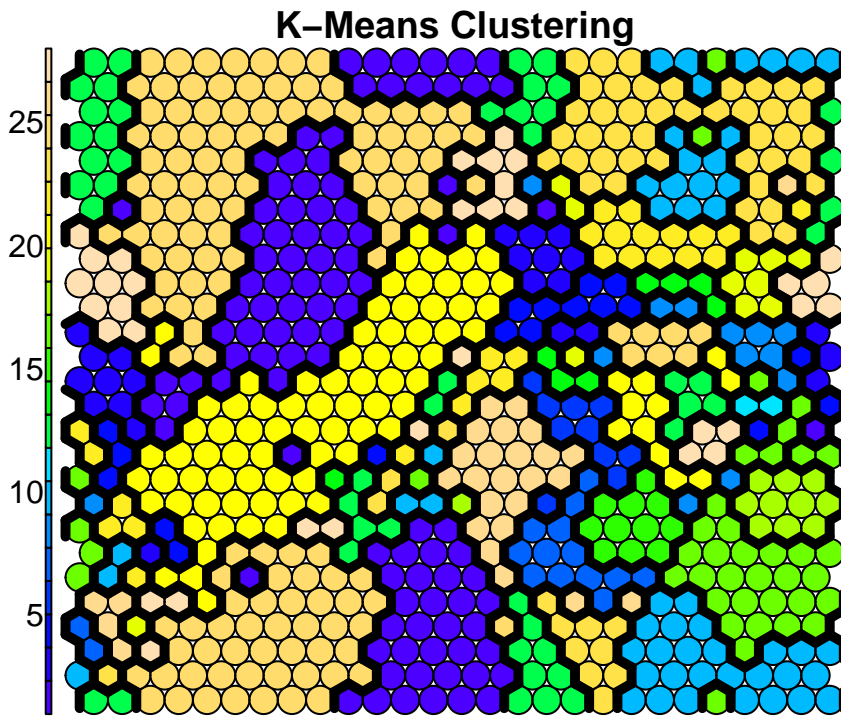
| | Sensitivity | Specificity | Pos Pred Value | Neg Pred Value | Precision | Recall | F1 | Prevalence | Detection Rate | Detection Prevalence | Balanced Accuracy |
|---------------------------------|-------------|-------------|----------------|----------------|-----------|-----------|-----------|------------|----------------|----------------------|-------------------|
| Class: montrealAPT | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealComradeCircle | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealCryptConsole | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealCryptXXX | 0.9966942 | 1.0000000 | 1.0000000 | 0.9997916 | 1.0000000 | 0.9966942 | 0.9983444 | 0.0593254 | 0.0591292 | 0.0591292 | 0.9983471 |
| Class: montrealCryptoLocker | 1.0000000 | 0.9994293 | 0.9980578 | 1.0000000 | 0.9980578 | 1.0000000 | 0.9990280 | 0.2267601 | 0.2267601 | 0.2272014 | 0.9997147 |
| Class: montrealCryptoLocker2015 | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealDMALocker | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealDMALockerv3 | 0.9885714 | 1.0000000 | 1.0000000 | 0.9999011 | 1.0000000 | 0.9885714 | 0.9942529 | 0.0085801 | 0.0084821 | 0.0084821 | 0.9942857 |
| Class: montrealEDA2 | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealFlyper | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealGlobe | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealGlobeImposter | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |

| | Sensitivity | Specificity | Pos Pred Value | Neg Pred Value | Precision | Recall | F1 | Prevalence | Detection Rate | Detection Prevalence | Balanced Accuracy |
|----------------------------|-------------|-------------|----------------|----------------|-----------|-----------|-----------|------------|----------------|----------------------|-------------------|
| Class: montrealGlobev3 | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealJigSaw | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealNoobCrypt | 0.9918033 | 1.0000000 | 1.0000000 | 0.9999008 | 1.0000000 | 0.9918033 | 0.9958848 | 0.0119631 | 0.0118651 | 0.0118651 | 0.9959016 |
| Class: montrealRazy | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealSam | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealSamSam | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealVenusLocker | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealWannaCry | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealXLocker | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealXLockerv5.0 | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: montrealXTPLocker | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: paduaCryptoWall | 0.9991963 | 0.9998589 | 0.9996784 | 0.9996473 | 0.9996784 | 0.9991963 | 0.9994373 | 0.3050108 | 0.3047656 | 0.3048637 | 0.9995276 |
| Class: paduaJigsaw | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: paduaKeRanger | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |
| Class: princetonCerber | 1.0000000 | 0.9999365 | 0.9997853 | 1.0000000 | 0.9997853 | 1.0000000 | 0.9998926 | 0.2283291 | 0.2283291 | 0.2283781 | 0.9999682 |
| Class: princetonLocky | 1.0000000 | 0.9999416 | 0.9996937 | 1.0000000 | 0.9996937 | 1.0000000 | 0.9998468 | 0.1600314 | 0.1600314 | 0.1600804 | 0.9999708 |
| Class: white | NA | 1.0000000 | NA | NA | NA | NA | NA | 0.0000000 | 0.0000000 | 0.0000000 | NA |

Clustering Visualizations: K-means clustering

K-means clustering offers a nice way of visualizing the final SOM grid and the categorical boundaries that were formed by the model.

Here is a nice graph. How do I center it?



Results & Performance (chunk #4, write up after chunk #3 is done)

Results

In the original paper by Akcora et al, they tested several different sets of parameters on their TDA model. According to them, “In the best TDA models for each ransomware family, we predict **16.59 false positives for each true positive**. In turn, this number is 27.44 for the best non-TDA models.”[3] In fact, the highest Precision (a.k.a.

Positive Predictive Value, defined as $TP/(TP+FP)$ they achieved was only 0.1610. Compare this to our final Precision value of 1.000? It is almost embarrassing... did I do something wrong here?

Performance

The script runs on the aforementioned hardware in less than five minutes and uses less than 4GB of RAM. Given that the Bitcoin network produces one new block every ten minutes on average, then real-time analysis could theoretically be conducted on each block as it is announced using even moderate computing resources.

Summary

Comparison to original paper and impact of findings

They suck, I rule, 'nuff said.

Limitations

SOMs seem like they are easy to misconfigure. Perhaps a dual Random Forest approach would be better?

Future Work

I only scratched the surface of the SOM algorithm which seems to have many implementations and parameters that could be investigated further and possibly optimized via cross-validation, somehow.

Conclusions

Get Monero!

This paper/report presents a reliable method for classifying bitcoin addresses into known ransomware families, while at the same time avoiding false positives by filtering them out using a binary method before classifying them further. It leaves the author of the paper wondering how long before we see ransomware using privacy coins such as Monero. Find and cite a recent paper on the untraceability of the Monero blockchain.

References

- [1] Adam Brian Turner, Stephen McCombie and Allon J. Uhlmann (November 30, 2020) Analysis Techniques for Illicit Bitcoin Transactions
- [2] Daniel Goldsmith, Kim Grauer and Yonah Shmalo (April 16, 2020) Analyzing hack subnetworks in the bitcoin transaction graph
- [3] Cuneyt Gurcan Akcora, Yitao Li, Yulia R. Gel, Murat Kantarcioglu (June 19, 2019) BitcoinHeist: Topological Data Analysis for Ransomware Detection on the Bitcoin Blockchain
- [4] UCI Machine Learning Repository <https://archive.ics.uci.edu/ml/index.php>
- [5] BitcoinHeist Ransomware Address Dataset /n <https://archive.ics.uci.edu/ml/datasets/BitcoinHeistRansomwareAddressDataset>
- [6] Available Models - The `caret` package <http://topepo.github.io/caret/available-models.html>
- [7] Ron Wehrens and Johannes Kruijselbrink, Package ‘kohonen’ @ CRAN (2019) <https://cran.r-project.org/web/packages/kohonen/kohonen.pdf>
- [XMR] Malte Möser*, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin (April 23, 2018) An Empirical Analysis of Traceability in the Monero Blockchain

Appendix:

Categorical SOM ransomware family prediction table and confusion matrix - detailed

Confusion Matrix and Statistics

```
##
##                                     Reference
## Prediction      montrealAPT  montrealComradeCircle
##  montrealAPT                0                0
##  montrealComradeCircle      0                0
##  montrealCryptConsole      0                0
##  montrealCryptXXX          0                0
##  montrealCryptoLocker      0                0
##  montrealCryptoTorLocker2015 0                0
##  montrealDMALocker         0                0
##  montrealDMALockerv3       0                0
##  montrealEDA2              0                0
##  montrealFlyper            0                0
##  montrealGlobe             0                0
##  montrealGlobeImposter     0                0
##  montrealGlobev3          0                0
##  montrealJigSaw            0                0
##  montrealNoobCrypt         0                0
##  montrealRazy              0                0
##  montrealSam               0                0
##  montrealSamSam            0                0
##  montrealVenusLocker       0                0
##  montrealWannaCry          0                0
##  montrealXLocker           0                0
##  montrealXLockerv5.0       0                0
##  montrealXTPLocker         0                0
##  paduaCryptoWall           0                0
##  paduaJigsaw               0                0
##  paduaKeRanger             0                0
##  princetonCerber           0                0
##  princetonLocky            0                0
##  white                      0                0
##
##                                     Reference
## Prediction      montrealCryptConsole  montrealCryptXXX
##  montrealAPT                0                0
##  montrealComradeCircle      0                0
##  montrealCryptConsole      0                0
##  montrealCryptXXX          0                1206
##  montrealCryptoLocker      0                3
##  montrealCryptoTorLocker2015 0                0
##  montrealDMALocker         0                0
##  montrealDMALockerv3       0                0
##  montrealEDA2              0                0
##  montrealFlyper            0                0
##  montrealGlobe             0                0
##  montrealGlobeImposter     0                0
##  montrealGlobev3          0                0
##  montrealJigSaw            0                0
##  montrealNoobCrypt         0                0
##  montrealRazy              0                0
##  montrealSam               0                0
##  montrealSamSam            0                0
```

| | | | |
|----|-----------------------------|----------------------|-----------------------------|
| ## | montrealVenusLocker | 0 | 0 |
| ## | montrealWannaCry | 0 | 0 |
| ## | montrealXLocker | 0 | 0 |
| ## | montrealXLockerv5.0 | 0 | 0 |
| ## | montrealXTPLocker | 0 | 0 |
| ## | paduaCryptoWall | 0 | 0 |
| ## | paduaJigsaw | 0 | 0 |
| ## | paduaKeRanger | 0 | 0 |
| ## | princetonCerber | 0 | 1 |
| ## | princetonLocky | 0 | 0 |
| ## | white | 0 | 0 |
| ## | | Reference | |
| ## | Prediction | montrealCryptoLocker | montrealCryptoTorLocker2015 |
| ## | montrealAPT | 0 | 0 |
| ## | montrealComradeCircle | 0 | 0 |
| ## | montrealCryptConsole | 0 | 0 |
| ## | montrealCryptXXX | 0 | 0 |
| ## | montrealCryptoLocker | 4625 | 0 |
| ## | montrealCryptoTorLocker2015 | 0 | 0 |
| ## | montrealDMALocker | 0 | 0 |
| ## | montrealDMALockerv3 | 0 | 0 |
| ## | montrealEDA2 | 0 | 0 |
| ## | montrealFlyper | 0 | 0 |
| ## | montrealGlobe | 0 | 0 |
| ## | montrealGlobeImposter | 0 | 0 |
| ## | montrealGlobev3 | 0 | 0 |
| ## | montrealJigSaw | 0 | 0 |
| ## | montrealNoobCrypt | 0 | 0 |
| ## | montrealRazy | 0 | 0 |
| ## | montrealSam | 0 | 0 |
| ## | montrealSamSam | 0 | 0 |
| ## | montrealVenusLocker | 0 | 0 |
| ## | montrealWannaCry | 0 | 0 |
| ## | montrealXLocker | 0 | 0 |
| ## | montrealXLockerv5.0 | 0 | 0 |
| ## | montrealXTPLocker | 0 | 0 |
| ## | paduaCryptoWall | 0 | 0 |
| ## | paduaJigsaw | 0 | 0 |
| ## | paduaKeRanger | 0 | 0 |
| ## | princetonCerber | 0 | 0 |
| ## | princetonLocky | 0 | 0 |
| ## | white | 0 | 0 |
| ## | | Reference | |
| ## | Prediction | montrealDMALocker | montrealDMALockerv3 |
| ## | montrealAPT | 0 | 0 |
| ## | montrealComradeCircle | 0 | 0 |
| ## | montrealCryptConsole | 0 | 0 |
| ## | montrealCryptXXX | 0 | 0 |
| ## | montrealCryptoLocker | 0 | 0 |
| ## | montrealCryptoTorLocker2015 | 0 | 0 |
| ## | montrealDMALocker | 0 | 0 |
| ## | montrealDMALockerv3 | 0 | 173 |
| ## | montrealEDA2 | 0 | 0 |
| ## | montrealFlyper | 0 | 0 |
| ## | montrealGlobe | 0 | 0 |
| ## | montrealGlobeImposter | 0 | 0 |

| | | | |
|----|---------------------|---|---|
| ## | montrealGlobev3 | 0 | 0 |
| ## | montrealJigSaw | 0 | 0 |
| ## | montrealNoobCrypt | 0 | 0 |
| ## | montrealRazy | 0 | 0 |
| ## | montrealSam | 0 | 0 |
| ## | montrealSamSam | 0 | 0 |
| ## | montrealVenusLocker | 0 | 0 |
| ## | montrealWannaCry | 0 | 0 |
| ## | montrealXLocker | 0 | 0 |
| ## | montrealXLockerv5.0 | 0 | 0 |
| ## | montrealXTPLocker | 0 | 0 |
| ## | paduaCryptoWall | 0 | 1 |
| ## | paduaJigsaw | 0 | 0 |
| ## | paduaKeRanger | 0 | 0 |
| ## | princetonCerber | 0 | 0 |
| ## | princetonLocky | 0 | 1 |
| ## | white | 0 | 0 |

| | | Reference | | |
|---------------|-----------------------------|--------------|----------------|---------------|
| ## Prediction | | montrealEDA2 | montrealFlyper | montrealGlobe |
| ## | montrealAPT | 0 | 0 | 0 |
| ## | montrealComradeCircle | 0 | 0 | 0 |
| ## | montrealCryptConsole | 0 | 0 | 0 |
| ## | montrealCryptXXX | 0 | 0 | 0 |
| ## | montrealCryptoLocker | 0 | 0 | 0 |
| ## | montrealCryptoTorLocker2015 | 0 | 0 | 0 |
| ## | montrealDMALocker | 0 | 0 | 0 |
| ## | montrealDMALockerv3 | 0 | 0 | 0 |
| ## | montrealEDA2 | 0 | 0 | 0 |
| ## | montrealFlyper | 0 | 0 | 0 |
| ## | montrealGlobe | 0 | 0 | 0 |
| ## | montrealGlobeImposter | 0 | 0 | 0 |
| ## | montrealGlobev3 | 0 | 0 | 0 |
| ## | montrealJigSaw | 0 | 0 | 0 |
| ## | montrealNoobCrypt | 0 | 0 | 0 |
| ## | montrealRazy | 0 | 0 | 0 |
| ## | montrealSam | 0 | 0 | 0 |
| ## | montrealSamSam | 0 | 0 | 0 |
| ## | montrealVenusLocker | 0 | 0 | 0 |
| ## | montrealWannaCry | 0 | 0 | 0 |
| ## | montrealXLocker | 0 | 0 | 0 |
| ## | montrealXLockerv5.0 | 0 | 0 | 0 |
| ## | montrealXTPLocker | 0 | 0 | 0 |
| ## | paduaCryptoWall | 0 | 0 | 0 |
| ## | paduaJigsaw | 0 | 0 | 0 |
| ## | paduaKeRanger | 0 | 0 | 0 |
| ## | princetonCerber | 0 | 0 | 0 |
| ## | princetonLocky | 0 | 0 | 0 |
| ## | white | 0 | 0 | 0 |

| | | Reference | |
|---------------|-----------------------------|-----------------------|-----------------|
| ## Prediction | | montrealGlobeImposter | montrealGlobev3 |
| ## | montrealAPT | 0 | 0 |
| ## | montrealComradeCircle | 0 | 0 |
| ## | montrealCryptConsole | 0 | 0 |
| ## | montrealCryptXXX | 0 | 0 |
| ## | montrealCryptoLocker | 0 | 0 |
| ## | montrealCryptoTorLocker2015 | 0 | 0 |

| | | | |
|----|-----------------------|---|---|
| ## | montrealDMALocker | 0 | 0 |
| ## | montrealDMALockerv3 | 0 | 0 |
| ## | montrealEDA2 | 0 | 0 |
| ## | montrealFlyper | 0 | 0 |
| ## | montrealGlobe | 0 | 0 |
| ## | montrealGlobeImposter | 0 | 0 |
| ## | montrealGlobev3 | 0 | 0 |
| ## | montrealJigSaw | 0 | 0 |
| ## | montrealNoobCrypt | 0 | 0 |
| ## | montrealRazy | 0 | 0 |
| ## | montrealSam | 0 | 0 |
| ## | montrealSamSam | 0 | 0 |
| ## | montrealVenusLocker | 0 | 0 |
| ## | montrealWannaCry | 0 | 0 |
| ## | montrealXLocker | 0 | 0 |
| ## | montrealXLockerv5.0 | 0 | 0 |
| ## | montrealXTPLocker | 0 | 0 |
| ## | paduaCryptoWall | 0 | 0 |
| ## | paduaJigsaw | 0 | 0 |
| ## | paduaKeRanger | 0 | 0 |
| ## | princetonCerber | 0 | 0 |
| ## | princetonLocky | 0 | 0 |
| ## | white | 0 | 0 |

| | | | | |
|----|-----------------------------|----------------|-------------------|--------------|
| ## | | Reference | | |
| ## | Prediction | montrealJigSaw | montrealNoobCrypt | montrealRazy |
| ## | montrealAPT | 0 | 0 | 0 |
| ## | montrealComradeCircle | 0 | 0 | 0 |
| ## | montrealCryptConsole | 0 | 0 | 0 |
| ## | montrealCryptXXX | 0 | 0 | 0 |
| ## | montrealCryptoLocker | 0 | 1 | 0 |
| ## | montrealCryptoTorLocker2015 | 0 | 0 | 0 |
| ## | montrealDMALocker | 0 | 0 | 0 |
| ## | montrealDMALockerv3 | 0 | 0 | 0 |
| ## | montrealEDA2 | 0 | 0 | 0 |
| ## | montrealFlyper | 0 | 0 | 0 |
| ## | montrealGlobe | 0 | 0 | 0 |
| ## | montrealGlobeImposter | 0 | 0 | 0 |
| ## | montrealGlobev3 | 0 | 0 | 0 |
| ## | montrealJigSaw | 0 | 0 | 0 |
| ## | montrealNoobCrypt | 0 | 242 | 0 |
| ## | montrealRazy | 0 | 0 | 0 |
| ## | montrealSam | 0 | 0 | 0 |
| ## | montrealSamSam | 0 | 0 | 0 |
| ## | montrealVenusLocker | 0 | 0 | 0 |
| ## | montrealWannaCry | 0 | 0 | 0 |
| ## | montrealXLocker | 0 | 0 | 0 |
| ## | montrealXLockerv5.0 | 0 | 0 | 0 |
| ## | montrealXTPLocker | 0 | 0 | 0 |
| ## | paduaCryptoWall | 0 | 1 | 0 |
| ## | paduaJigsaw | 0 | 0 | 0 |
| ## | paduaKeRanger | 0 | 0 | 0 |
| ## | princetonCerber | 0 | 0 | 0 |
| ## | princetonLocky | 0 | 0 | 0 |
| ## | white | 0 | 0 | 0 |

| | | | | |
|----|------------|-------------|----------------|---------------------|
| ## | | Reference | | |
| ## | Prediction | montrealSam | montrealSamSam | montrealVenusLocker |

| | | | | |
|----|-----------------------------|---|---|---|
| ## | montrealAPT | 0 | 0 | 0 |
| ## | montrealComradeCircle | 0 | 0 | 0 |
| ## | montrealCryptConsole | 0 | 0 | 0 |
| ## | montrealCryptXXX | 0 | 0 | 0 |
| ## | montrealCryptoLocker | 0 | 0 | 0 |
| ## | montrealCryptoTorLocker2015 | 0 | 0 | 0 |
| ## | montrealDMALocker | 0 | 0 | 0 |
| ## | montrealDMALockerv3 | 0 | 0 | 0 |
| ## | montrealEDA2 | 0 | 0 | 0 |
| ## | montrealFlyper | 0 | 0 | 0 |
| ## | montrealGlobe | 0 | 0 | 0 |
| ## | montrealGlobeImposter | 0 | 0 | 0 |
| ## | montrealGlobev3 | 0 | 0 | 0 |
| ## | montrealJigSaw | 0 | 0 | 0 |
| ## | montrealNoobCrypt | 0 | 0 | 0 |
| ## | montrealRazy | 0 | 0 | 0 |
| ## | montrealSam | 0 | 0 | 0 |
| ## | montrealSamSam | 0 | 0 | 0 |
| ## | montrealVenusLocker | 0 | 0 | 0 |
| ## | montrealWannaCry | 0 | 0 | 0 |
| ## | montrealXLocker | 0 | 0 | 0 |
| ## | montrealXLockerv5.0 | 0 | 0 | 0 |
| ## | montrealXTPLocker | 0 | 0 | 0 |
| ## | paduaCryptoWall | 0 | 0 | 0 |
| ## | paduaJigsaw | 0 | 0 | 0 |
| ## | paduaKeRanger | 0 | 0 | 0 |
| ## | princetonCerber | 0 | 0 | 0 |
| ## | princetonLocky | 0 | 0 | 0 |
| ## | white | 0 | 0 | 0 |

| ## | | Reference | |
|---------------|-----------------------------|------------------|-----------------|
| ## Prediction | | montrealWannaCry | montrealXLocker |
| ## | montrealAPT | 0 | 0 |
| ## | montrealComradeCircle | 0 | 0 |
| ## | montrealCryptConsole | 0 | 0 |
| ## | montrealCryptXXX | 0 | 0 |
| ## | montrealCryptoLocker | 0 | 0 |
| ## | montrealCryptoTorLocker2015 | 0 | 0 |
| ## | montrealDMALocker | 0 | 0 |
| ## | montrealDMALockerv3 | 0 | 0 |
| ## | montrealEDA2 | 0 | 0 |
| ## | montrealFlyper | 0 | 0 |
| ## | montrealGlobe | 0 | 0 |
| ## | montrealGlobeImposter | 0 | 0 |
| ## | montrealGlobev3 | 0 | 0 |
| ## | montrealJigSaw | 0 | 0 |
| ## | montrealNoobCrypt | 0 | 0 |
| ## | montrealRazy | 0 | 0 |
| ## | montrealSam | 0 | 0 |
| ## | montrealSamSam | 0 | 0 |
| ## | montrealVenusLocker | 0 | 0 |
| ## | montrealWannaCry | 0 | 0 |
| ## | montrealXLocker | 0 | 0 |
| ## | montrealXLockerv5.0 | 0 | 0 |
| ## | montrealXTPLocker | 0 | 0 |
| ## | paduaCryptoWall | 0 | 0 |
| ## | paduaJigsaw | 0 | 0 |

| | | | |
|----|-----------------|---|---|
| ## | paduaKeRanger | 0 | 0 |
| ## | princetonCerber | 0 | 0 |
| ## | princetonLocky | 0 | 0 |
| ## | white | 0 | 0 |

| | | | |
|----|-----------------------------|---------------------|-------------------|
| ## | | Reference | |
| ## | Prediction | montrealXLockerv5.0 | montrealXTPLocker |
| ## | montrealAPT | 0 | 0 |
| ## | montrealComradeCircle | 0 | 0 |
| ## | montrealCryptConsole | 0 | 0 |
| ## | montrealCryptXXX | 0 | 0 |
| ## | montrealCryptoLocker | 0 | 0 |
| ## | montrealCryptoTorLocker2015 | 0 | 0 |
| ## | montrealDMALocker | 0 | 0 |
| ## | montrealDMALockerv3 | 0 | 0 |
| ## | montrealEDA2 | 0 | 0 |
| ## | montrealFlyper | 0 | 0 |
| ## | montrealGlobe | 0 | 0 |
| ## | montrealGlobeImposter | 0 | 0 |
| ## | montrealGlobev3 | 0 | 0 |
| ## | montrealJigSaw | 0 | 0 |
| ## | montrealNoobCrypt | 0 | 0 |
| ## | montrealRazy | 0 | 0 |
| ## | montrealSam | 0 | 0 |
| ## | montrealSamSam | 0 | 0 |
| ## | montrealVenusLocker | 0 | 0 |
| ## | montrealWannaCry | 0 | 0 |
| ## | montrealXLocker | 0 | 0 |
| ## | montrealXLockerv5.0 | 0 | 0 |
| ## | montrealXTPLocker | 0 | 0 |
| ## | paduaCryptoWall | 0 | 0 |
| ## | paduaJigsaw | 0 | 0 |
| ## | paduaKeRanger | 0 | 0 |
| ## | princetonCerber | 0 | 0 |
| ## | princetonLocky | 0 | 0 |
| ## | white | 0 | 0 |

| | | | | |
|----|-----------------------------|-----------------|-------------|---------------|
| ## | | Reference | | |
| ## | Prediction | paduaCryptoWall | paduaJigsaw | paduaKeRanger |
| ## | montrealAPT | 0 | 0 | 0 |
| ## | montrealComradeCircle | 0 | 0 | 0 |
| ## | montrealCryptConsole | 0 | 0 | 0 |
| ## | montrealCryptXXX | 0 | 0 | 0 |
| ## | montrealCryptoLocker | 5 | 0 | 0 |
| ## | montrealCryptoTorLocker2015 | 0 | 0 | 0 |
| ## | montrealDMALocker | 0 | 0 | 0 |
| ## | montrealDMALockerv3 | 0 | 0 | 0 |
| ## | montrealEDA2 | 0 | 0 | 0 |
| ## | montrealFlyper | 0 | 0 | 0 |
| ## | montrealGlobe | 0 | 0 | 0 |
| ## | montrealGlobeImposter | 0 | 0 | 0 |
| ## | montrealGlobev3 | 0 | 0 | 0 |
| ## | montrealJigSaw | 0 | 0 | 0 |
| ## | montrealNoobCrypt | 0 | 0 | 0 |
| ## | montrealRazy | 0 | 0 | 0 |
| ## | montrealSam | 0 | 0 | 0 |
| ## | montrealSamSam | 0 | 0 | 0 |
| ## | montrealVenusLocker | 0 | 0 | 0 |

```

## montrealWannaCry 0 0 0
## montrealXLocker 0 0 0
## montrealXLockerv5.0 0 0 0
## montrealXTPLocker 0 0 0
## paduaCryptoWall 6216 0 0
## paduaJigsaw 0 0 0
## paduaKeRanger 0 0 0
## princetonCerber 0 0 0
## princetonLocky 0 0 0
## white 0 0 0
##
## Reference
## Prediction princetonCerber princetonLocky white
## montrealAPT 0 0 0
## montrealComradeCircle 0 0 0
## montrealCryptConsole 0 0 0
## montrealCryptXXX 0 0 0
## montrealCryptoLocker 0 0 0
## montrealCryptoTorLocker2015 0 0 0
## montrealDMALocker 0 0 0
## montrealDMALockerv3 0 0 0
## montrealEDA2 0 0 0
## montrealFlyper 0 0 0
## montrealGlobe 0 0 0
## montrealGlobeImposter 0 0 0
## montrealGlobev3 0 0 0
## montrealJigSaw 0 0 0
## montrealNoobCrypt 0 0 0
## montrealRazy 0 0 0
## montrealSam 0 0 0
## montrealSamSam 0 0 0
## montrealVenusLocker 0 0 0
## montrealWannaCry 0 0 0
## montrealXLocker 0 0 0
## montrealXLockerv5.0 0 0 0
## montrealXTPLocker 0 0 0
## paduaCryptoWall 0 0 0
## paduaJigsaw 0 0 0
## paduaKeRanger 0 0 0
## princetonCerber 4657 0 0
## princetonLocky 0 3264 0
## white 0 0 0
##
## Overall Statistics
##
## Accuracy : 0.9994
## 95% CI : (0.9989, 0.9997)
## No Information Rate : 0.305
## P-Value [Acc > NIR] : < 2.2e-16
##
## Kappa : 0.9992
##
## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
## Class: montrealAPT Class: montrealComradeCircle

```

| | | |
|-------------------------|------------------------------------|------------------------------|
| ## Sensitivity | NA | NA |
| ## Specificity | 1 | 1 |
| ## Pos Pred Value | NA | NA |
| ## Neg Pred Value | NA | NA |
| ## Prevalence | 0 | 0 |
| ## Detection Rate | 0 | 0 |
| ## Detection Prevalence | 0 | 0 |
| ## Balanced Accuracy | NA | NA |
| ## | Class: montrealCryptConsole | Class: montrealCryptXXX |
| ## Sensitivity | NA | 0.99669 |
| ## Specificity | 1 | 1.00000 |
| ## Pos Pred Value | NA | 1.00000 |
| ## Neg Pred Value | NA | 0.99979 |
| ## Prevalence | 0 | 0.05933 |
| ## Detection Rate | 0 | 0.05913 |
| ## Detection Prevalence | 0 | 0.05913 |
| ## Balanced Accuracy | NA | 0.99835 |
| ## | Class: montrealCryptoLocker | |
| ## Sensitivity | 1.0000 | |
| ## Specificity | 0.9994 | |
| ## Pos Pred Value | 0.9981 | |
| ## Neg Pred Value | 1.0000 | |
| ## Prevalence | 0.2268 | |
| ## Detection Rate | 0.2268 | |
| ## Detection Prevalence | 0.2272 | |
| ## Balanced Accuracy | 0.9997 | |
| ## | Class: montrealCryptoTorLocker2015 | |
| ## Sensitivity | NA | |
| ## Specificity | 1 | |
| ## Pos Pred Value | NA | |
| ## Neg Pred Value | NA | |
| ## Prevalence | 0 | |
| ## Detection Rate | 0 | |
| ## Detection Prevalence | 0 | |
| ## Balanced Accuracy | NA | |
| ## | Class: montrealDMALocker | Class: montrealDMALockerv3 |
| ## Sensitivity | NA | 0.988571 |
| ## Specificity | 1 | 1.000000 |
| ## Pos Pred Value | NA | 1.000000 |
| ## Neg Pred Value | NA | 0.999901 |
| ## Prevalence | 0 | 0.008580 |
| ## Detection Rate | 0 | 0.008482 |
| ## Detection Prevalence | 0 | 0.008482 |
| ## Balanced Accuracy | NA | 0.994286 |
| ## | Class: montrealEDA2 | Class: montrealFlyper |
| ## Sensitivity | NA | NA |
| ## Specificity | 1 | 1 |
| ## Pos Pred Value | NA | NA |
| ## Neg Pred Value | NA | NA |
| ## Prevalence | 0 | 0 |
| ## Detection Rate | 0 | 0 |
| ## Detection Prevalence | 0 | 0 |
| ## Balanced Accuracy | NA | NA |
| ## | Class: montrealGlobe | Class: montrealGlobeImposter |
| ## Sensitivity | NA | NA |
| ## Specificity | 1 | 1 |

| | | |
|-------------------------|----------------------------|----------------------------|
| ## Pos Pred Value | NA | NA |
| ## Neg Pred Value | NA | NA |
| ## Prevalence | 0 | 0 |
| ## Detection Rate | 0 | 0 |
| ## Detection Prevalence | 0 | 0 |
| ## Balanced Accuracy | NA | NA |
| ## | Class: montrealGlobev3 | Class: montrealJigSaw |
| ## Sensitivity | NA | NA |
| ## Specificity | 1 | 1 |
| ## Pos Pred Value | NA | NA |
| ## Neg Pred Value | NA | NA |
| ## Prevalence | 0 | 0 |
| ## Detection Rate | 0 | 0 |
| ## Detection Prevalence | 0 | 0 |
| ## Balanced Accuracy | NA | NA |
| ## | Class: montrealNoobCrypt | Class: montrealRazy |
| ## Sensitivity | 0.99180 | NA |
| ## Specificity | 1.00000 | 1 |
| ## Pos Pred Value | 1.00000 | NA |
| ## Neg Pred Value | 0.99990 | NA |
| ## Prevalence | 0.01196 | 0 |
| ## Detection Rate | 0.01187 | 0 |
| ## Detection Prevalence | 0.01187 | 0 |
| ## Balanced Accuracy | 0.99590 | NA |
| ## | Class: montrealSam | Class: montrealSamSam |
| ## Sensitivity | NA | NA |
| ## Specificity | 1 | 1 |
| ## Pos Pred Value | NA | NA |
| ## Neg Pred Value | NA | NA |
| ## Prevalence | 0 | 0 |
| ## Detection Rate | 0 | 0 |
| ## Detection Prevalence | 0 | 0 |
| ## Balanced Accuracy | NA | NA |
| ## | Class: montrealVenusLocker | Class: montrealWannaCry |
| ## Sensitivity | NA | NA |
| ## Specificity | 1 | 1 |
| ## Pos Pred Value | NA | NA |
| ## Neg Pred Value | NA | NA |
| ## Prevalence | 0 | 0 |
| ## Detection Rate | 0 | 0 |
| ## Detection Prevalence | 0 | 0 |
| ## Balanced Accuracy | NA | NA |
| ## | Class: montrealXLocker | Class: montrealXLockerv5.0 |
| ## Sensitivity | NA | NA |
| ## Specificity | 1 | 1 |
| ## Pos Pred Value | NA | NA |
| ## Neg Pred Value | NA | NA |
| ## Prevalence | 0 | 0 |
| ## Detection Rate | 0 | 0 |
| ## Detection Prevalence | 0 | 0 |
| ## Balanced Accuracy | NA | NA |
| ## | Class: montrealXTPLocker | Class: paduaCryptoWall |
| ## Sensitivity | NA | 0.9992 |
| ## Specificity | 1 | 0.9999 |
| ## Pos Pred Value | NA | 0.9997 |
| ## Neg Pred Value | NA | 0.9996 |

| | | | |
|-------------------------|---|--------|----|
| ## Prevalence | 0 | 0.3050 | |
| ## Detection Rate | 0 | 0.3048 | |
| ## Detection Prevalence | 0 | 0.3049 | |
| ## Balanced Accuracy | NA | 0.9995 | |
| ## | Class: paduaJigsaw Class: paduaKeRanger | | |
| ## Sensitivity | NA | NA | |
| ## Specificity | 1 | 1 | |
| ## Pos Pred Value | NA | NA | |
| ## Neg Pred Value | NA | NA | |
| ## Prevalence | 0 | 0 | |
| ## Detection Rate | 0 | 0 | |
| ## Detection Prevalence | 0 | 0 | |
| ## Balanced Accuracy | NA | NA | |
| ## | Class: princetonCerber Class: princetonLocky Class: white | | |
| ## Sensitivity | 1.0000 | 1.0000 | NA |
| ## Specificity | 0.9999 | 0.9999 | 1 |
| ## Pos Pred Value | 0.9998 | 0.9997 | NA |
| ## Neg Pred Value | 1.0000 | 1.0000 | NA |
| ## Prevalence | 0.2283 | 0.1600 | 0 |
| ## Detection Rate | 0.2283 | 0.1600 | 0 |
| ## Detection Prevalence | 0.2284 | 0.1601 | 0 |
| ## Balanced Accuracy | 1.0000 | 1.0000 | NA |