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

HarvardX Final Capstone CYO Project

Kaylee Robert Tejeda

10/31/2021

Abstract

Ransomware is a persisent 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

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 The news of such an attack might not be published at all, let alone in popular media.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 :character	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
Mode :character	Median :2014	Median :181.0	Median : 8.00	Median : 0.2500	Median : 1.0	Median : 0.0	Median : 2.000	Median :2.000e+08	Mode :character
NA	Mean :2014	Mean :181.5	Mean : 45.01	Mean : 0.5455	Mean : 721.6	Mean : 238.5	Mean : 2.207	Mean :4.465e+09	NA
NA	3rd Qu.:2016	3rd Qu.:271.0	3rd Qu.:108.00	3rd Qu.: 0.8819	3rd Qu.: 56.0	3rd Qu.: 0.0	3rd Qu.: 2.000	3rd Qu.:9.940e+08	NA
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
111K8kZAEnJg245r2cM6y9zgJGHZtJPy6	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
1129TSjKtx65E35GiUo4AYVeyo48twbrGX	2016	238	144	0.0728484	456	0	1	200000000	princetonLocky
112AmFATxzhuSpvtz1hfpa3Zrw3BG276pc	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 transfered less than Ƀ0.3 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 (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.

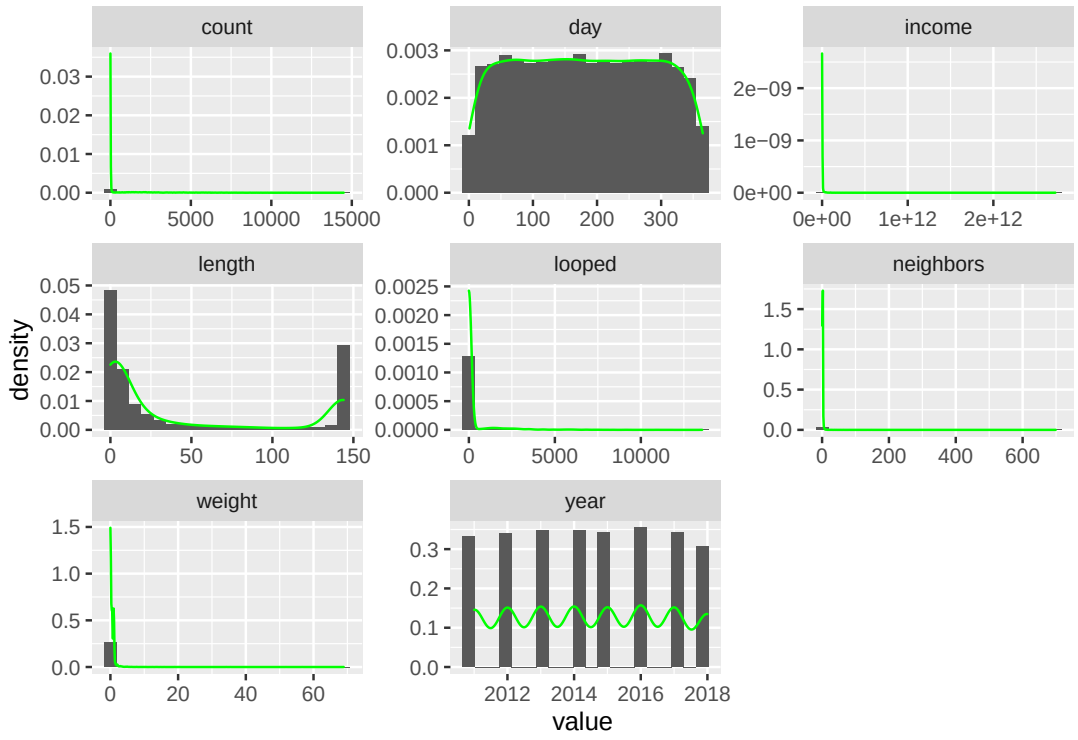
```
## The proportion of ransomware addresses in the original data set is 0.0141985951917529.
```

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

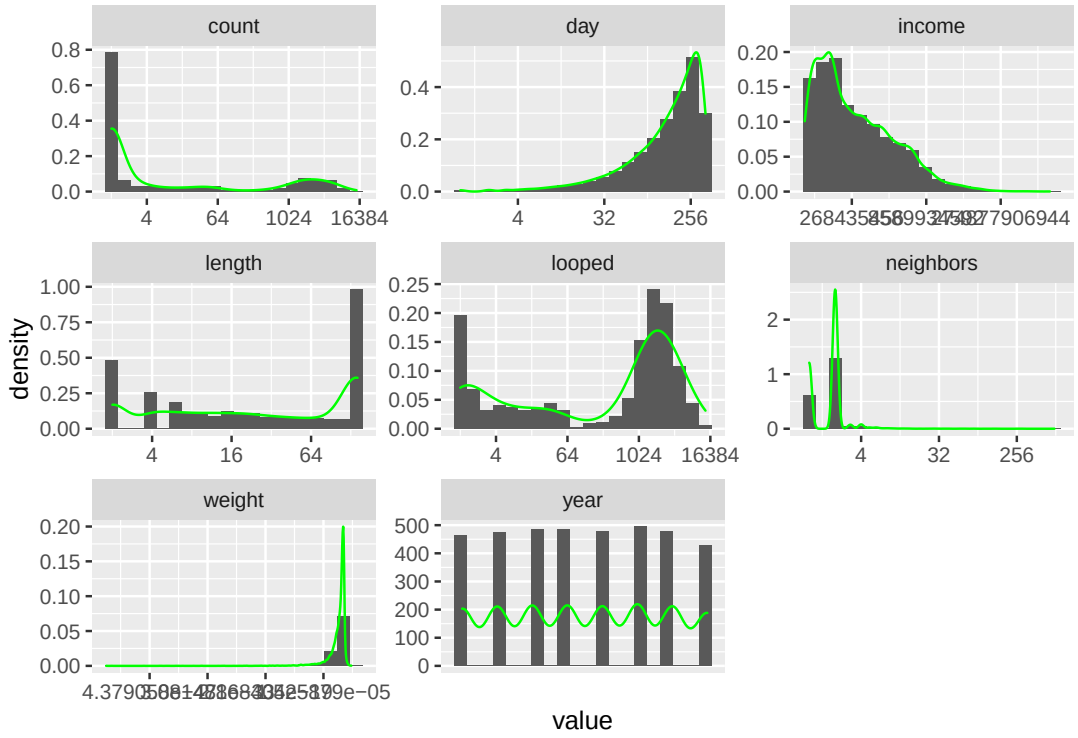
Let's take a look at the distribution of the different features. Note how skewed the non-temporal features are, some of them being bimodal:

Table 3: 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		

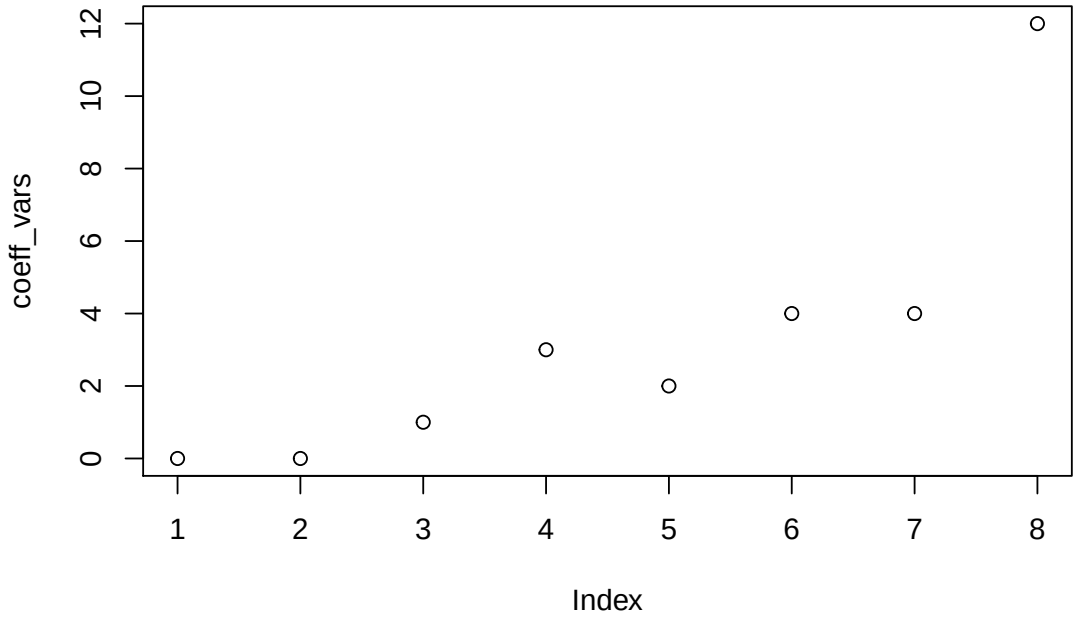



```
## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 8033 rows containing non-finite values (stat_bin).

## Warning: Removed 8033 rows containing non-finite values (stat_density).
```



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.

	x
year	0
day	0
length	1
weight	3
count	2
looped	4
neighbors	4
income	12



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] 0.01539101
```

7) Principle Component Analysis can go here. See "Interlinkages of Malaysian Banking Systems" for an example of detailed PCA. Is it exploratory analysis, or is it a predictive method? I was under the assumption that it is a form of analysis, but the paper mentioned extends it to a form of predictive modeling. How to do this *right* (?!?!)

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.

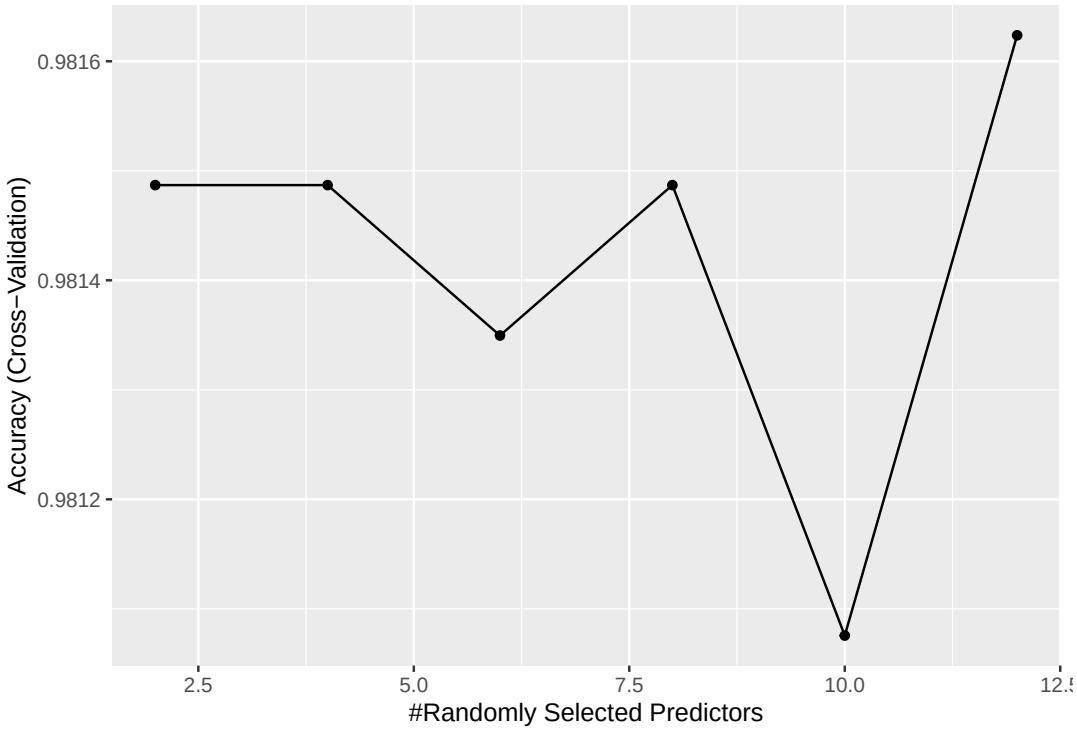
Modeling approach

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.

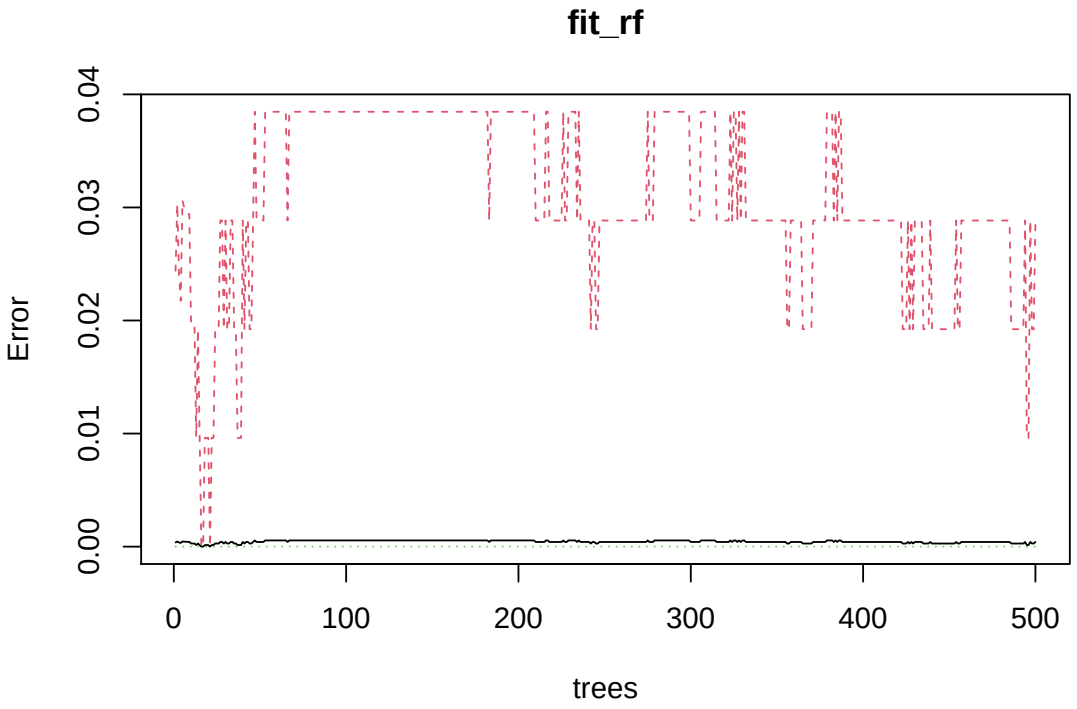
Method 1: Binary Random Forests

Using the `randomForest` library, we train a model on our training set and test against the "black/white" categorization on our test set.

We can see that the results are quite good against the smaller test set and the larger validation set.




```
##   mtry
## 6   12
```




```
## Confusion Matrix for test set:

## Confusion Matrix and Statistics
##
##           Reference
## Prediction black white
```

```
##      black   104     0
##      white     0  7188
##
##                Accuracy : 1
##                  95% CI : (0.9995, 1)
##     No Information Rate : 0.9857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.00000
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 1.00000
##              Prevalence : 0.01426
##          Detection Rate : 0.01426
##    Detection Prevalence : 0.01426
##       Balanced Accuracy : 1.00000
##
##        'Positive' Class : black
##
```

```
## Confusion Matrix for validation set:
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   black   white
##      black   20441       0
##      white     266 1437642
##
##                Accuracy : 0.9998
##                  95% CI : (0.9998, 0.9998)
##     No Information Rate : 0.9858
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9934
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.98715
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 0.99982
##              Prevalence : 0.01420
##          Detection Rate : 0.01402
##    Detection Prevalence : 0.01402
##       Balanced Accuracy : 0.99358
##
##        'Positive' Class : black
##
```

Method 2: Binary SOMs

If we ask the same question to a more sophisticated and topological approach, how good is the model? Mention how the original paper was topological in nature, an how this lead to the investigation of SOMs. Repeat the binary "b/w" approach using SOMs. This accuracy is still pretty good, but not *as* good as the random forest method. Point out how SOMs are really used for classification into *many* groups. This leads to an Insight! (see above) What if we first *isolate* the "black" addresses using Random Forest, and then categorize the black only subset ($< 2\%$) using categorical SOMs. This leads to a 2-part system...

Note to self: I don't even use this part in the final script. Should I leave it out of the paper too?

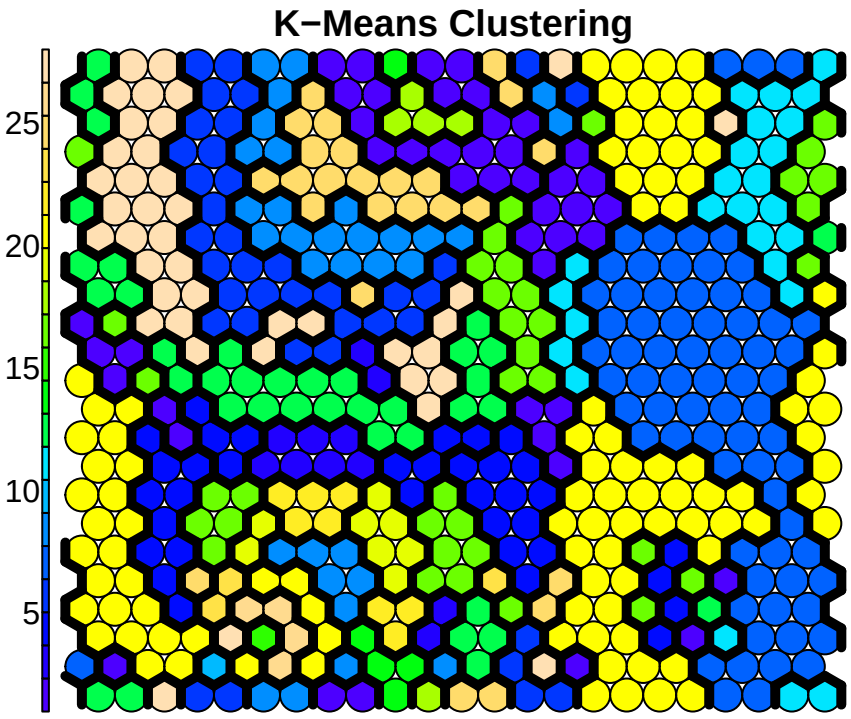
Method 3: Categorical SOMs

Describe categorical SOM work here, show results. This is where the pretty colored hex-graphs show up.

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9942040      0.9925795      0.9925456      0.9955742      0.2989760
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9938662      0.9921468      0.9927062      0.9948841      0.2990099
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
## [1] 28
```



Final Method: Combined Methods 1 and 3

Using the results from Random Forest, isolate the black addresses first, and then run that subset through an SOM algorithm. Compare final results to original paper. These go in a "results" section. (below)

Results & Performance

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 embarassing... did I do something wrong here?

Performance

The overall script takes X hours and X minutes to run on the aforementioned hardware. This could be optimized, but given that it is an eight year old laptop, this is not too unreasonable. It takes me longer to compile LibreOffice.

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 he 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 know 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 untracability 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 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 Kruisselbrink, 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

```
## 182.614 sec elapsed
```