Using Machine Learning to predict whether bitcoin transactions are associated with ransomware

Michael Polonio

Introduction

As we progress further into the information age our reliance on technology becomes more prevalent. This has been exploited by cyber criminals whose goal is to extort large amounts of money from businesses and the general public in the form of Bitcoin payments. To their targets they deploy ransomware, or malicious software that encrypts a computers file system and demands payment to be decrypted. Machines can be rendered useless when encrypted, locking out the user. This can be problematic if the computers are running critical infrastructure, or something vital to national security. The payments sent to the attacker for the decryption key are always in the form of cryptocurrency, Bitcoin in particular. In this paper I will use machine learning techniques to produce a model that predicts whether particular bitcoin transactions are being used as a ransomware payment i.e. finding a bitcoin wallet address that belongs to the attacker. Upon finding that an address has indeed been used for malicious intent, action can be taken against that bitcoin address, such as banning it from any future transactions or blacklisting it to prevent further online scams (Chandrasekharuni, 2021).

Problem Statement

Given a bitcoin payment transaction along with the data pertaining to its node in the bitcoin graph, the goal is to predict if the payment is associated with a ransomware payment. The graph features are designed to quantify specific transaction patterns (Akcora et al., UCI, 2020). These predictors will be used in several classification algorithms in order to classify each transaction as either a ransomware payment, or not. Each model will then be compared and evaluated for accuracy.

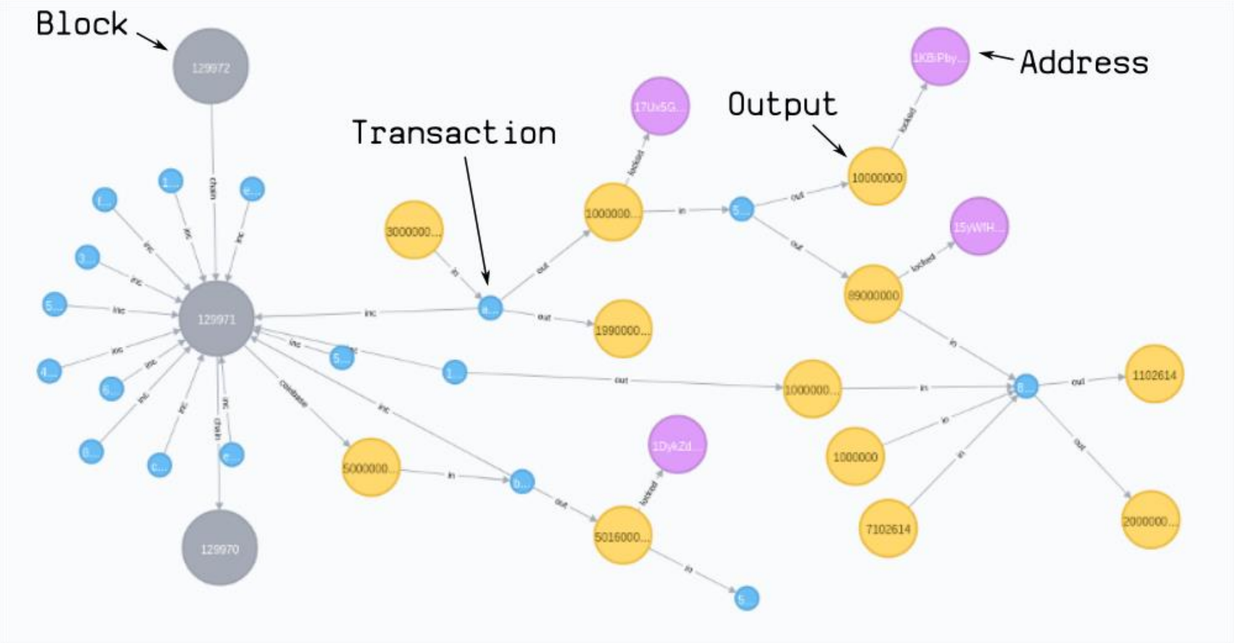


Fig 1: Bitcoin Transaction Graph

Data

The data is publicly available from the UCI Machine Learning Repository (BitcoinHeistRansomwareAddressDataset) and contains 29,16,697 observations. Each observation represents a node in the bitcoin transaction graph and contains the following features:

address: String. Bitcoin address

year: int. Year

day: int. Day of the year. 1 is the first day, 365 is the last

length: int. Length is designed to quantify mixing rounds on Bitcoin, where transactions receive and distribute similar amounts of coins in multiple rounds with newly created addresses to hide the coin origin (Chandrasekharuni, 2021).

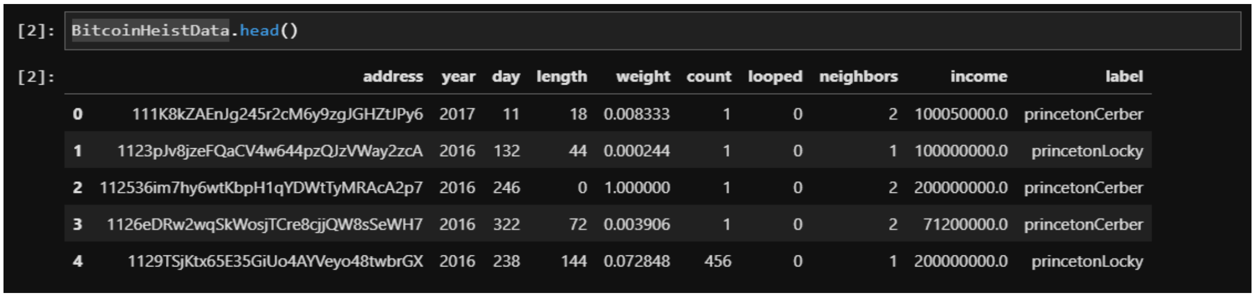
weight: float. Weight quantifies the merge behavior (i.e., the transaction has more input addresses than output addresses), where coins in multiple addresses are each passed through a succession of merging transactions and accumulated in a final address (Chandrasekharuni, 2021).

count: int. Similar to weight, the count feature is designed to quantify the merging pattern. However, the count feature represents information on the number of transactions, whereas the weight feature represents information on the amount of transaction (Chandrasekharuni, 2021).

looped: int. Loop is intended to count how many transactions i) split their coins; ii) move these coins in the network by using different paths and finally, and iii) merge them in a single address. Coins at this final address can then be sold and converted to fiat currency (Chandrasekharuni, 2021).

neighbors: int. Indicates the number of neighbors a transaction had (Chandrasekharuni, 2021).

income: int. Income in terms of Satoshi amount where a Satoshi is the smallest unit of a bitcoin, equivalent to 100 millionth of a bitcoin (Chandrasekharuni, 2021).



```
[2]: BitcoinHeistData.head()
```

```
[2]:
                                   address  year  day  length    weight  count  looped  neighbors       income           label
   0    111K8kZAEnJg245r2cM6y9zgJGHZtJPy6  2017   11      18  0.008333      1       0          2  100050000.0  princetonCerber
   1   1123pJv8jzeFQaCV4w644pzQJzVWay2zcA  2016  132      44  0.000244      1       0          1  100000000.0   princetonLocky
   2  112536im7hy6wtKbpH1qYDWtTyMRAcA2p7  2016  246       0  1.000000      1       0          2  200000000.0  princetonCerber
   3   1126eDRw2wqSkWosjTCre8cjjQW8sSeWH7  2016  322      72  0.003906      1       0          2   71200000.0  princetonCerber
   4    1129TSjKtx65E35GiUo4AYVeyo48twbrGX  2016  238     144  0.072848    456       0          1  200000000.0   princetonLocky
```

Fig 2: First five entries in the dataset

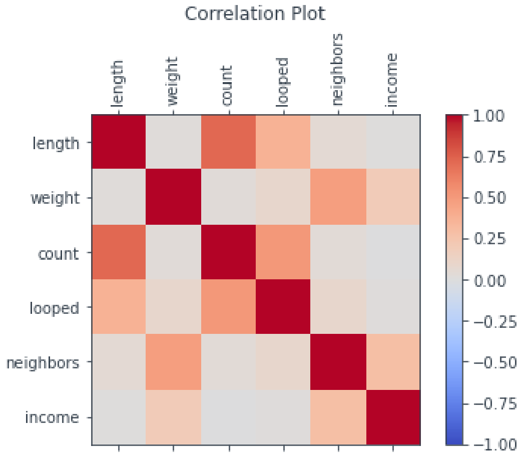
Methods

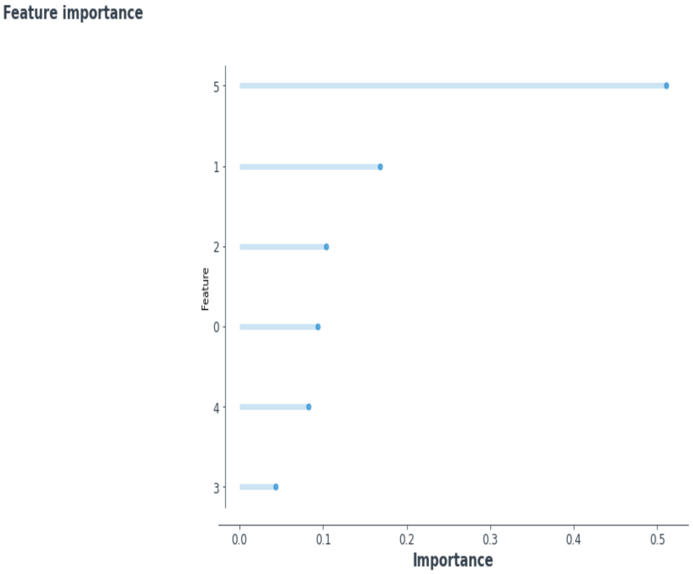
Python and the Scikit-learn machine learning library will be used in this analysis. I suspect the 'year' column may cause overfitting when sampling the data, so it will be dropped. Address string is not needed and will be dropped. The final features for the Decision Tree, Random Forest, and Naïve Bayes Classifier are: length, weight, count, looped, neighbors, and income. The features used for the SVM will be length, weight, neighbors, and income. Turn the problem into binary classification by assigning 0 for non-ransomware and 1 for ransomware to a newly created feature, which is the target variable (ransomware). Assign equal number of ransomware and non-ransomware observations to new dataset (41,413), which is the total number of ransomware transactions in the original dataset. The data was split into training and testing sets with a 70%/30% split. The data was trained using scikit-learns DecisionTreeClassifier, RandomForestClassifier, GaussianNB, and SVC with a polynomial and radial basis function kernel. The data was evaluated using 10-fold cross validation. The models were then fit with the testing data and evaluated by their accuracy, CV score, F1 score, precision and recall. Further evaluation includes the confusion matrix for each model.

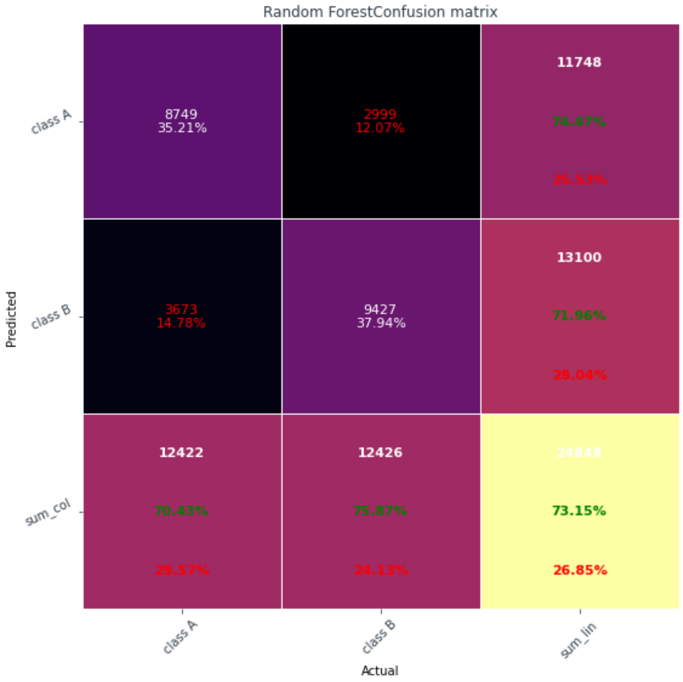
Results

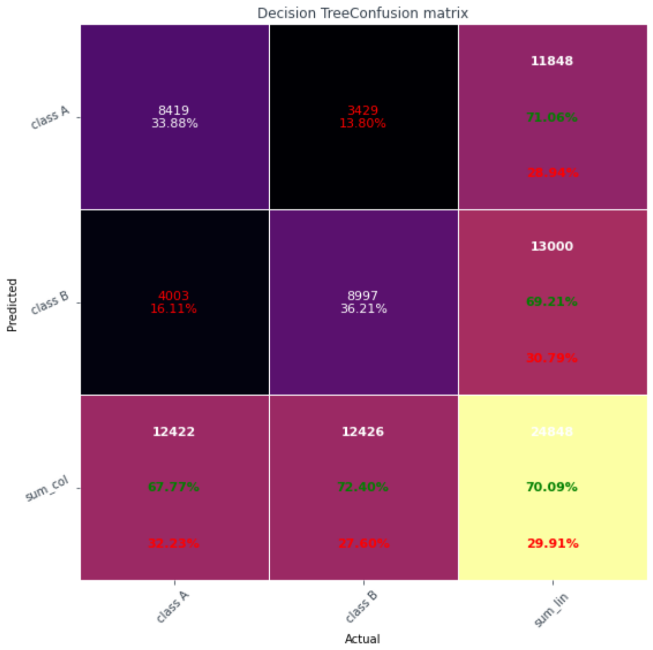
Model Name	Accuracy	CV Score	F1-Score	Precision	Recall
Decision Tree	70%	0.70	0.707	0.69	0.72
Random Forest	73%	0.69	0.740	0.72	0.76
Naive Bayes	51%	0.69	0.669	0.50	N/A
SVM (polynomial kernel)	55%	0.53	N/A	N/A	N/A

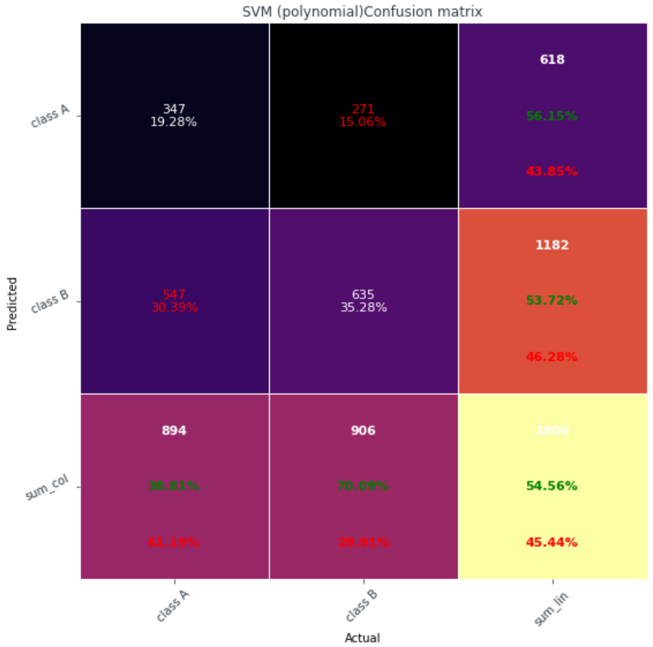
SVM (RBF kernel)	52%	0.53	N/A	N/A	N/A











Conclusion

The results show that all of the models performed with an accuracy better than guessing, some significantly better. The Random Forest (50 Decision Trees) performed the best, predicting ransomware bitcoin transactions with an accuracy of 73%, a cross validation score (10-fold) of 0.70, F1 score of 0.74, Precision of 0.72, and a Recall of 0.76. The Random Forest outperformed all other models in every evaluation metric. The Decision Tree was second with an accuracy of 70%. Next was the SVM (polynomial kernel) at 54%, then SVM (rbf) at 52%, and the Naïve Bayes Classifier at 50%, the same as blind guessing. Overall, I think these results are promising and I believed they can be improved with hyperparameter tuning, but still can be applied to the problem of predicting ransomware bitcoin transactions.

References

Chandrasekharuni, Yogesh (2021). Using AI to detect Bitcoin addresses involved in ransomware transactions. Analytics Vidya. Using AI to detect Bitcoin addresses involved in ransomware transactions | by Yogesh Chandrasekharuni | Analytics Vidhya | Medium

Al-Haija, Q. A., & Alsulami, A. A. (2021). High Performance Classification Model to Identify Ransomware Payments for Heterogeneous Bitcoin Networks. *Electronics*, *10*(17), 2113. Electronics | Free Full-Text | High Performance Classification Model to Identify Ransomware Payments for Heterogeneous Bitcoin Networks (mdpi.com)

[Data] UCI Machine Learning Repository. Accessed 2021. UCI Machine Learning Repository: BitcoinHeistRansomwareAddressDataset Data Set

[Fig 1] Bitcoin Transaction Graph. neo4j-bitcoin.png (1000×521)