Hang Miao
miaohang@gmail.com
(848) 203-8144

Python Code Link:
https://github.com/HangMiaoEcon/MachineLearning/tree/main/Bitcoin_RansomWare_Detection

Summary of work

The dataset contains nearly 3 million on-chain Bitcoin transactions, each comes with six graph-based features summarized from the given paper as well as time stamp and ransom classification ground truth. The task is to implement supervised classifier(s) that could identify ransom-related addresses in the future, and to compare with what's reported in the paper (mainly the TDA Mapper as well as four other clustering/classification models). Ideally, I should be training and testing the data in a timed setting so as to evaluate my models' capability of predicting the future (such as training on a year and testing on the successive year), however, for the purpose of just demonstrating problem solving and due to limited time, I only trained my models on a random draw of 20% of the data and tested on the rest 80%. More comprehensive evaluation could be applied given my scripts if it's actually needed.

In addition to the approaches already covered by the paper (TDA, DBSCAN, K-means, XGBT and Random Forest) I trained an enhanced **XGBoost** model and **Deep Neural Network** with the following argument respectively. XGBoost as an ensemble method gained its popularity in recent machine learning competitions. DNN gain an edge on image recognition, NLP, AI etc.

As the result, shown in the table below, for the most represented ransom families, the enhanced XGBT model was achieving better performance for some of the top ransom families compared to the paper in most aspect (precision, recall and F1-score). On the contrary, Deep Neural Network performed poorly and can rarely detect any ransom families in my preliminary test (results not shown). The reason is that the idiosyncratic property of the dataset. Unbalanced and sparsity of the malware target families and only 6 dimensional features is not enough to train a decent neural network compared with a typical picture at least 10,000 pixels.

		Precision	Recall	F1-score
CryptXXX	**enhanced XGBT**	0.671	0.321	0.434
	TDA.35|.35	0.030	0.221	0.053
	COSINE	0.027	0.491	0.052
Cerber	**enhanced XGBT**	0.627	0.173	0.271
	TDA.5|.35	0.035	0.289	0.062
	XGBT	0.033	0.181	0.056
CryptoLocker	**enhanced XGBT**	0.588	0.005	0.011
	TDA.65|.65	0.043	0.674	0.081
	DBSCAN.15	0.021	0.760	0.042
CryptoWall	**enhanced XGBT**	0.742	0.027	0.052
	TDA.8|.65	0.066	0.583	0.118
	DBSCAN.2	0.037	0.478	0.069
Locky	**enhanced XGBT**	0.653	0.050	0.093
	TDA.8|.5	0.161	0.900	0.273
	COSINE	0.054	0.375	0.095
white	**enhanced XGBT**	0.987	0.999	0.993

Discussion

1) One of the biggest challenges in detecting ransom-related addresses is that ransom addresses were buried among huge number of white addresses, casting a difficult trade-off to identify them with both good sensitivity and specificity. Yet a nature of unsupervised on-chain behavior is to not have too much labor-heavy censorship to scrutinize the suspicious accounts, hence it's probably best to have more stringent criteria in calling an address random-related rather than identifying lots of suspicious accounts (*i.e.,* we should value specificity more than sensitivity).

2) One important hypothesis, as well as foundation of the given paper, is that random-related transactions indeed have different patterns than normal transactions so that we can identify them. Another hypothesis suggested is that different ransom families might have varying patterns for us to distinguish them, which may or may not be true. Nevertheless, I'm hereby following the logic in the paper and tested my models for each random family separately, though a better approach might be to identify random vs. white addresses first assuming all ransom-related transactions share similar patterns, then to group the ransom-related into various clusters. This is especially important for some of the short-lived ransom families or some newly established ransom families that barely have much data, it's almost impossible to identify them among all, yet under the assumption that we can first identify any ransom-related addresses, then we might be able to hit two birds with one stone: identifying ransom-related addresses, and to classify them into known or unknown families.

3) About the six graph features utilized by the paper, my opinion is that while the features are making sense in providing potential transaction patterns, other features could have been included, such as but not limited to diversity of the downstream neighbors for an address (upstream neighbors are already accounted for in the "neighbors" feature), balance of an address within a small time window (given an assumption that random intermediate address might clear out its income in a relatively shorter time frame), etc. Actually, for most modern classifiers, it's ok or better to have more features, even when redundant or dependent, rather than limiting to the six given features.

4) To make the model work in a setting closer to real-time, it's probably important to update the model with more training data as we got them, and to assign heavier weights for the more recent transactions. One of the benefits for the enhanced XGBT model proposed here, compare to other models, is its ability to be scaled-up to really work with "real-time" identification needs.