Trust Prediction in Bitcoin Alpha Web of Trust Network

LEONARDO GAMBIRASIO MAIALEN LOINAZ SONJA TERVOLA
lgam@kth.se maiala@kth.se tervola@kth.se

May 8, 2024

Abstract

This paper explores the challenge of predicting trust within the Bitcoin Alpha trust network, in which users rate each other's trustworthiness based on their interactions. Using datasets containing historical trust ratings ranging from -10 (complete distrust) to +10 (full trust), the study aims to develop a recommendation system. This system is designed to predict trust scores among users who have not interacted directly. Applying advanced network analysis and graph theory techniques, the model assigns scores to potential new interactions, similar to how traditional recommendation systems suggest products. The primary goal is to predict the probability and quality of future interactions within the network. The results of this research offer insights for future decisions and strategies for participation in cryptocurrency exchanges, improving understanding of trust dynamics in decentralized financial systems.

Contents

1 Introduction

Trust is a vital component in the dynamics of relationships between individuals. With the rise of web-based communities and online social networks, the influence of trust has garnered considerable attention. These platforms often provide unrestricted, open, and sometimes anonymous access, attracting a vast array of users. However, this openness also increases the risk of misconduct, leaving these networks exposed and somewhat vulnerable. Beyond the realm of computer science, trust has been explored within various other disciplines such as sociology, psychology, and economics. The overarching goal in these studies is to develop metrics that can quantify user behavior within specific social environments. Furthermore, mechanisms based on trust are increasingly being utilized to predict future user behaviors.

A cryptocurrency is a digital asset designed to function as a means of electronic payment. Bitcoin is a type of cryptocurrency characterized as an alias. This means that the cryptocurrency within a wallet is not directly linked to individuals, it is associated with one or more specific keys. Consequently, although all transactions are publicly recorded on the blockchain, the identities of Bitcoin owners remain unidentifiable. The key distinction between cryptocurrencies and conventional electronic money lies in their structure: cryptocurrencies operate on a decentralized basis, unlike centralized banking systems that govern both electronic and fiat currencies. Given the anonymous nature of these platforms, that do not reveal the true identities of participants, understanding the existence and the level of trust in such markets is critically important. A recent news by Bloomberg highlighted how over two billion dollars has been lost due to malicious users.

The Bitcoin Alpha web of trust network offers an intriguing perspective on how users perceive one another's trustworthiness. This web of trust is built on users' experiences, where they rate one another based on their interactions, forming a rich and complex network of trust relationships.

1.1 Problem definition

Our project seeks to address the challenge of trust prediction within this network. By leveraging historical trust ratings among users, we aim to develop a model that can predict trust scores between pairs of users who have not directly transacted with each other.

We use available datasets from containing information about how bitcoin users rate each other upon an interaction in exchange platforms. Trust ratings in this network range from -10 (representing complete distrust) to +10 (indicating full confidence). Therefore, higher rating indicates a good experience with the particular user.

We are developing a recommender system designed to predict the trustworthiness of links within a network based on previously observed interactions and temporal data. This system operates by assigning scores to potential link formations between nodes in a graph, analogous to how a traditional recommender system might suggest products to users. The primary objective is to predict the likelihood and quality of new interactions (links) that might occur in future periods.

Our results will provide insights about future decisions and strategies for participation in cryptocurrency exchanges.

2 Related Work

Several studies have explored the application of graph theory and network science to evaluate and predict trust in Bitcoin networks. The integration of link prediction techniques is particularly notable for its ability to anticipate future transactions and relationships based on historical interaction data [1], which emphasized the role of weighted signed networks in predicting transactions within the Bitcoin Over-The-Counter (OTC) network . This approach underscores the potential for using network analysis to not only predict but also to enhance the understanding of trust dynamics in decentralized financial systems.

Moreover, trust in these networks is not isolated to individual interactions but emerges as a characteristic of broader community structures. Previous research conducted by Punčeva [2] demonstrated that trust can be quantified and characterized within communities, using methods such as modularity and clustering coefficients to identify trust-based communities within Bitcoin exchanges. These communities provide a foundation for more reliable and secure transactions, suggesting a direct correlation between network structure and the perceived and actual security of transactions.

Parallel to trust studies, the field of link prediction in social networks has been explored as a method to understand and forecast connections within a network, providing insights into network evolution and community structures [3]. Techniques in link prediction apply to various applications, from social network growth to predicting future interactions in cryptoeconomic systems.

In summary, the related work in this field converges on the significance of understanding and predicting trust in Bitcoin networks. By leveraging advanced network analysis techniques and adapting traditional social network analysis tools, research can provide insights that help enhance the security and reliability of transactions in these complex networks.

3 Method

3.1 Coding language and libraries

The following information is intended to allow proper evaluation of new research built on top of our code by comparing it with our results. For this purpose, all software and hardware details will be defined.

The entire code is written in Python 3.10.1 [4] through Visual Studio Code. The Python libraries that have been used are: Pandas [5], Numpy [6], Scikit-learn [7], NetworkX [8].

Follows the specifications of the machine in which the code was run to obtain the final results.

Brand	Dell
Model	Latitude 5401
Chip	Intel core i7 vPro 9850H
Memory	16 GB

Table 1: Hardware details.

3.2 Dataset

Our primary data source is the Bitcoin Alpha web of trust network dataset [9], a rich repository of interactions within the network. The dataset contains:

- Nodes: A total of 3,783 unique users form the network.
- Edges: There are 24,186 trust relationships captured in the dataset.
- Edge Weights: These weights represent trust ratings, spanning from -10 (distrust) to +10 (trust).
- Temporal Aspect: Each trust rating is timestamped, allowing us to analyze how trust dynamics evolve over time.

The percentage of positive edges is 93%.

3.3 Data preprocessing

Since the dataset is very unbalanced (93% of positive edged), we opted to create a balanced subset of edges to improve our predictions by avoiding overfitting phenomena on positive trust scores.

The new balanced dataset has been created in the following way:

1. All the negative edges have been taken.
2. A random subsample of the positive edges has been taken. The size of this subsample is the same as the number of negative entries.
3. The two sets were put together and shuffled.

With the purpose of allowing a correct evaluation comparing our results to the ground truth, we randomly split the new dataset into a Training set (80%) and a Test set (20%).

We are aware that this subsampling of the original dataset likely altered the structure of the graph, removing important edges that could have connected different communities of nodes together. Future improvements of this research may include finding a better subsampling method that takes into account the graph structure.

Starting from the new Training set we now created the graph using NetworkX. The result is a Directed and Weighted graph with 1502 nodes and 2458 edges.

3.4 Network analysis and preparation

In order to understand which nodes were the most important in the graph, we used the PageRank algorithm (from NetworkX). This algorithm scores each node according to its importance within the network.

For a better understanding we also calculated the trust score mean of every node. The trust score mean is the mean of the trust scores of all the edges where the node is either source or target. Follow the results.

The tables are showing the worst nodes and the best nodes. The first column of the tables is the absolute position of a node based on PageRank score. This means the node in position 0 is the most important within the entire network. Important to notice that the most important node (node 7604) has a very negative trust score mean (the trust score is in range -10 to 10).

The purpose of this analysis is selecting the most important negative node and the most important positive node. We will use this information later while generating features. We selected node 7604 as the ”worst one” and node 1 as the ”best one”.

	node	page_rank	mean	n_edges
0	7604	0.012285	-8.327869	61
1	7603	0.009309	-4.974359	39
2	177	0.008601	-4.149254	67
3	7564	0.007965	-7.176471	34
4	798	0.006808	-3.230769	26

Figure 1: Top five negative nodes

	node	page_rank	mean	n_edges
7	1	0.005547	1.750000	44
8	2	0.005213	0.138889	36
11	16	0.004561	0.684211	19
13	22	0.003996	0.423077	26
17	17	0.003396	2.000000	13

Figure 2: Top five positive nodes

3.5 Graph reduction

After selecting the best and worst node, we will calculate the similarity scores between them and all other nodes in the graph. The algorithm we applied is computationally heavy, so we were forced to reduce the size of the graph.

Instead of randomly selecting the nodes destined to be removed, we made use of the previously calculated PageRank score. In this way we removed the least significant nodes, reducing the damage to the graph structure (compared to the random selection method).

The nodes with PageRank score < 0.0005 has been removed. Since more than 50% of the nodes within the graph had PageRank score < 0.0005, the number of nodes in the reduced graph is 637 (instead of the previous 1502 nodes).

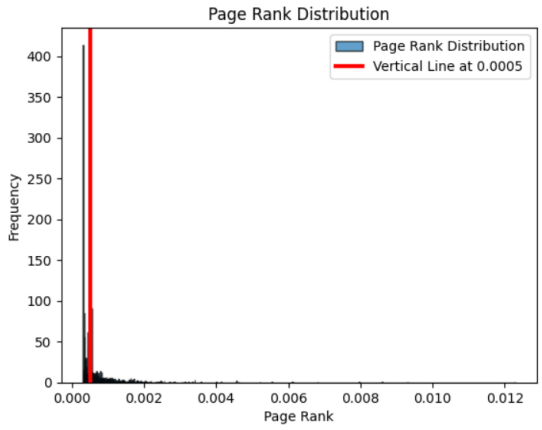

Figure 3: PageRank distribution

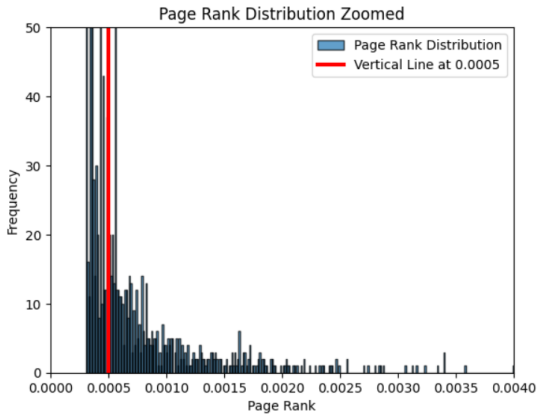

Figure 4: Zoomed PageRank distribution

From the graph have been generated the two new Training set and Test set. Training set became the list of all the reduced graph's edges. Test set is a subsample of the previous test set. Only edges that have at least one node in the graph were retained. Edge weights were obviously maintained.

3.6 Feature generation

From the previously generated graph (the reduced graph), the new training set, and the new test set, we need to generate features that provide information about the network with the purpose of predicting trust score values between two not-yet-connected nodes.

We began to calculate the similarity scores between the nodes in the Training set and the previously selected best and worst node. The similarity algorithm works only if both the source and target nodes are present in

the graph. We are sure that every node in the Training set is present in the graph, since the Training set was created from the graph.

For the purpose of calculating the similarity score we used the SimRank algorithm (from NetworkX).

Since the calculation is rather expensive, we optimized the number of calculations. During the iteration of the Training set (a list of edges: each row consists of source node, destination node, weight), before calculating the score we checked whether the score for that specific node was already present in a specially created dictionary. If it was already calculated, we skipped the operation, otherwise we calculated the score and wrote it into the dictionary.

The output of these operations are two dictionaries in which both use nodes as keys. The values for the first dictionary are the similarity scores between the node and the worst node, and the similarity scores between the node and the best node for the second dictionary.

We then performed the same operations to calculate the similarity scores between the nodes in the Test set and the worst/best node. There is only one difference from the previous calculations: we are not sure that all nodes in the test set are present in the graph. Therefore, before calculating the score we had to make sure that the node was present in the graph, and if it was not, we added the edge described by the specific row of the Test set, calculated the similarity score, and then removed the edge.

The other features we generated starting from the graph are Eigenvector centrality scores, node degrees and PageRank scores. The resulting feature set will be composed of:

- Source node and target node
- Eigenvector centrality score of both source and target node
- Degree of both source and target node
- PageRank score of both source and target node
- SimRank score between source/target nodes and worst node
- SimRank score between source/target nodes and best node

The feature set will look like the following table.

	source	target	eig_cent_source	eig_cent_target	degree_source	degree_target	rank_source	rank_target	sim_worst_source	sim_worst_target	sim_best_source	sim_best_target
0	7363	7502	0.004566	0.026701	2	4	0.000764	0.001083	0.025743	0.032000	0.024126	0.024956
1	7502	125	0.026701	0.038597	4	11	0.001083	0.001942	0.032000	0.031241	0.024956	0.024328
2	7502	7363	0.026701	0.004566	4	2	0.001083	0.000764	0.032000	0.025743	0.024956	0.024126
3	272	7571	0.012534	0.004777	14	3	0.001110	0.000816	0.033478	0.021571	0.019166	0.017209
4	272	7596	0.012534	0.023733	14	5	0.001110	0.002417	0.033478	0.032538	0.019166	0.022391

Figure 5: Firsts rows of the feature set used for training the model

3.7 Training the model

The last step for predicting the trust scores is training a machine learning model. For this research we opted for a Random Forest Classifier (from Scikit-learn).

The model has been trained using 3 different versions of the dataset, with the purpose of comparing the performances using different features. The tested versions are:

- Version 1: Source node and target node, Eigenvector centrality scores and node degrees.
- Version 2: All the previous plus PageRank scores.
- Version 3: All the previous plus SimRank scores (Final version, Figure 5).

The results will be explained in the next section.

4 Results and Discussion

The results can be resumed in the following table.

Feature set version	Exact match	Threshold 1	Threshold 2	Mean squared error
Version 1	0.498	0.619	0.736	18.785
Version 2	0.490	0.606	0.723	21.507
Version 3	0.543	0.647	0.753	17.991

Table 2: Table of results

For evaluating our predictions we used 3 different methods: accuracy (exact matches out of the total number of predictions), accuracy with a threshold t and the Mean Squared Error.

We decided to implement the "precision with threshold" method because we believe that a forecast that is very close to the true value should be considered correct. Given a range of 20 points, a distance of 1-2 points between the predicted value and the actual value is still a good prediction and provides very important insights about our prediction model.

The threshold value indicates the margin given to consider the prediction valid. For example, if the true value is 3 and a threshold of 2 is given, any predicted number between 1 and 5 (+/-2) will be considered a correct prediction.

We will analyze the results for version 3 mentioned before, as it is the most complete one. In the third version, the model is trained using the source and target nodes, Eigenvector centrality scores and degrees, PageRank scores, and SimRank scores. The results (shown in the third row of Table 2) indicate that we have obtained a 54.3% exact match, 64.5% correct predictions with a threshold of 1, and 75% correct predictions with a threshold of 2. Moreover, since the data set is balanced and the number of negative and positive samples is equal, we can assume that the results do not overfit the positive samples.

Regarding other metrics, we have calculated the Mean Square Error (MSE), which is a measure of the average of the squares of the errors. The error is the amount by which the value predicted by a model differs from the actual value.

Before analyzing our results, we should know that the MSE is always a non-negative value, and the closer it is to zero, the better. It penalizes larger errors more severely than smaller ones because the errors are

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Figure 6: MSE formula

squared. This metric is used in regression analysis and other statistical modeling to assess the quality of a model.

With the obtained value of 17.99, as shown in Table 2, we can say that our model has improved compared to the MSE of version 1 (18.78) and version 2 (21.5), since a lower MSE indicates better performance in terms of prediction accuracy.

The MSE is very high. This suggests that our model predicts most of the time the correct result, but when it predicts wrong the wrong value is very far from the actual value.

As a comparison metric, we plotted the correctly predicted values and the wrongly predicted ones for each possible rating from -10 to 10.

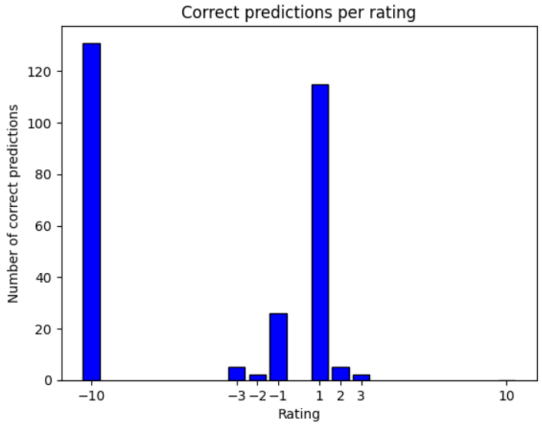

Figure 7: Correct predictions

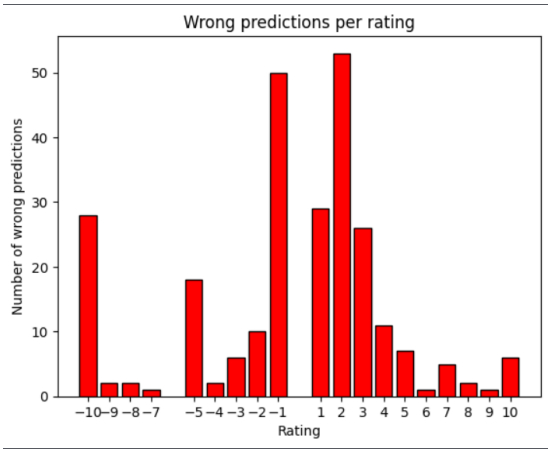

Figure 8: Wrong predictions

It is noticeable that the correct predictions were mostly for really negative ratings. This is due to the high importance of the worst node in our adapted dataset. We used the worst node (node 7604) to calculate the similarity scores between the nodes in the dataset and itself. The worst node is ”first” within the network in terms of PageRank score and it has a trust score mean of -8.32 in 61 edges (information visible in Figure 1). This means that the node is very important within the network and very unreliable, since most of its edges have a score of -10 (the lowest value on the scale). Therefore, being similar to this node means having a high probability of being unreliable. Therefore, our model is quite good in predicting very bad nodes.

Unfortunately, we can not say the same for the best node (node 1). It is important within the network, but it has a trust score mean close to the middle value 0 (1.75 mean score in 44 edges). This means that the node has almost the same amount of positive edges and negative edges. Because of its weak “positivity,” the similarity score between the nodes and itself does not help predictions as much as the worst node.

Finally, as for the incorrect ratings, most of them were obtained among the average ratings, which is not a big problem since our primary goal is to predict the worst/best cases. Also, it is preferable to predict an intermediate wrong value than to predict an extreme wrong value; it is safer.

5 Conclusion

The use of a reduced graph was motivated by computational constraints. However, we were able to achieve promising results in predicting the worst/most malicious users. These findings demonstrate the potential of our approach in identifying problematic nodes within a network.

For future improvements, we aim to use the entire graph and calculate the SimRank algorithm for not only the best/worst node but for the best k nodes and the worst k nodes. This procedure will be very computationally intensive but could provide more comprehensive insights. Additionally, we plan to explore different models and fine-tune them to improve prediction accuracy. This includes trying various machine learning algorithms and ensemble methods.

To enhance computational efficiency, we will investigate methods to optimize the computation of graph-based features such as SimRank. This might involve parallel computing techniques or more efficient algorithms to handle larger datasets. Furthermore, we intend to increase the size and diversity of the dataset to include more significant nodes, both good and bad. A larger dataset with more significant good nodes would allow us to train the model for both types of nodes, potentially improving the predictions for all types of users.

Incorporating additional features that might influence node behavior, such as temporal dynamics, node attributes, and interaction patterns over time, is another area of focus. We also plan to conduct a thorough error analysis to understand the types of errors the model is making and why. This could help in refining the model and improving its robustness.

Validation of the model's predictions in real-world scenarios by collaborating with domain experts will help in assessing the practical applicability and effectiveness of the model. Finally, integrating feedback mechanisms where users can provide input on the model's predictions will be beneficial. This feedback can be used to further refine and improve the model.

By addressing these areas, we can enhance the robustness, accuracy, and applicability of our model, leading to better predictions and more effective identification of malicious nodes in various network scenarios.

6 Contributions

Team member	Contributions
Leonardo Gambirasio	Presentations, Reports, Methodology and Code
Maialen Loinaz	Presentations, Reports
Sonja Tervola	Presentations, Reports

Table 3: Contributions of individual team members

References

[1] Tanevski Oliver et al., "Link prediction on Bitcoin OTC network," *17th International Conference for Informatics and Information Technology*, May 2020.

[2] P. M. et al., "Bitcoin Trust Communities," *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT),*, 2018.

[3] P. W. et al., "Link Prediction in Social Networks: the State-of-the-Art," *Science China: Information Science*, Jan. 2015.

[4] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[5] T. pandas development team, "pandas-dev/pandas: Pandas," feb 2020.

[6] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[8] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[9] S. Kumar, F. Spezzano, V. Subrahmanian, and C. Faloutsos, "Edge weight prediction in weighted signed networks," in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pp. 221–230, IEEE, 2016.