Bitcoin Trust Factor Mining using Graph Neural Networks

Shriram Balaji Athreya
sa84@illinois.edu
University of Illinois - Urbana-Champaign
Urbana, Illinois, USA

Sindhu Inuganti
vsi3@illinois.edu
University of Illinois - Urbana-Champaign
Urbana, Illinois, USA

ABSTRACT

Blockchain Technology has accelerated the usage of digital currencies and cryptocurrency trading. Blockchain technology protects data from any manipulation or theft of trading data. Bitcoin is the largest cryptocurrency chain built with over 200k transactions per day. Bitcoin Trading via Exchanges takes high transaction fees from the traders due to which OTC (Over The Counter) trading has become popular. OTC Trading provides higher profits. The risk with OTC Trading is counterpart fraud. This paper discusses the solution to improve OTC Trading Credibility through the use of data mining techniques. We aim to perform inductive trust prediction using the trade data available to us in the form of a transaction graph.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; *Supervised learning by classification*; *Supervised learning by regression.*

KEYWORDS

graph data mining, large networks, neural networks, trust factor detection, big data

ACM Reference Format:
Shriram Balaji Athreya and Sindhu Inuganti. 2022. Bitcoin Trust Factor Mining using Graph Neural Networks. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXX XXX.XXXXXXX

1 INTRODUCTION

In the era of digital currencies and crypto trading, Bitcoin transactions are emerging as an innovative way of payment network. Backed by cryptography and blockchain, a rising technology for decentralized data ledger supporting digital transactions, bitcoin trading is made possible without the need for a central authority. With this advantage, bitcoins gained a reputation as a prominent anonymous form of payment, free from any tracking leading to enhanced confidentiality of information and security. Consequently, privacy and anonymity have become increasingly necessary and dangerous as well [1]. There is a dramatic rise in cybercrimes causing the destruction of data, theft of intellectual property, fraudulent transactions, theft of financial data, reputation damage, and many more, inflicting damages totaling 6 Trillion USD globally in 2021. It is implicated that cyber attacks could also potentially affect the economy of a city, state, or entire country, forcing an exponential growth in the costs for cyber-security ventures. There is an immediate need for innovative solutions for preventing OTC trading fraud. All traders, especially beginners will benefit from refraining to perform transactions with entities that have low reliability.

Conference'17, July 2017, Washington, DC, USA
2022. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

But the main challenge of preserving digital anonymity still stands. This project addresses this challenge of determining the trust factor [13] of an anonymous entity with minimal information about the user with data mining techniques using neural networks. Without any identity or PII of a user/trader, we attempt to form respective personas of traders based on their activity on the bitcoin trading platform. Bitcoin OTC (Over-The-Counter) is one such framework that enables trade marketing between 2 participants with masked identities but collects the ratings of every such transaction, ranging between -10 and 10. These ratings help in forming an individual's persona with additional factors like total ratings, and a total number of positive and negative ratings given and received, further establishing the reliability of the user that facilitates the decision to make the transaction or not. [8]

In the terms of Deep Learning and Neural Networks, this problem can be mined over a graph where users/traders are the nodes, and transactions between them are seen as edges. Nodes are characterized based on multiple features derived from ratings given and received. Nodes further contribute to determining the Trust factor of transactions [3]. This project deals with the node (user) trust factor prediction using ratings provided in the graph ranging from -1 to +1, which makes it a Weighted Signed Network(WSN). WSNs are an implementation of GNNs [11], here used for predicting the node labels (trust factor).

Previous work [10] on edge weight prediction on WSNs has been published related to social networking and bitcoin networks to determine the positive/negative reviews, tweets, ratings, or relations. But this paper attempt to assess the trustfulness and trustworthiness of nodes, which are believed to be interdependent on each other and require to be calculated recursively to find each factor. Here, trustfulness is the characterization of a user based on the reviews received, and trustworthiness accounts for the credibility of the ratings given by the user. For example, ratings given by an unreliable user or a user with a less trustingness score, get a lesser trustworthiness score, leading to a lesser weight for the edge, meaning that the transaction might not be safe enough. Thus, these two factors can be calculated and used for training the GNNs on WSNs for a recursive edge weight prediction model.

2 LITERATURE SURVEY

One approach that has been used for predicting fraudulent user behavior in OTC Bitcoin transactions is the use of machine learning algorithms. These algorithms are able to learn from data and identify patterns that may indicate fraudulent behavior. There has been a usage of machine learning algorithms to predict fraudulent user behavior in OTC Bitcoin transactions by analyzing user-generated content on Bitcoin forums.

Another approach that has been used for predicting fraudulent user behavior in OTC Bitcoin transactions is the use of network analysis techniques. These techniques involve analyzing the relationships between users in a network and can be used to identify patterns that may indicate fraudulent behavior. For example, [16] used network analysis to predict fraudulent user behavior in OTC Bitcoin transactions by analyzing the connections between users. The analysis was able to accurately identify fraudulent behavior and could be used to improve the security of OTC Bitcoin transactions.

In addition to these approaches, there has also been research on using data visualization techniques for predicting fraudulent user behavior in OTC Bitcoin transactions. These techniques involve creating visual representations of data, such as graphs and charts, in order to identify patterns that may indicate fraudulent behavior. For example, [14] used data visualization to predict fraudulent user behavior in OTC Bitcoin transactions by creating graphs of user relationships on Bitcoin forums. The analysis was able to identify patterns that were indicative of fraudulent behavior and could be used to improve the security of OTC Bitcoin transactions.

Overall, the research on using data mining techniques for predicting fraudulent user behavior in OTC Bitcoin transactions has demonstrated the potential of these methods for improving the security of these transactions. By using machine learning algorithms, network analysis, and data visualization, it is possible to identify patterns that may indicate fraudulent behavior and take appropriate action to prevent these transactions from occurring.

The bitcoin OTC data was used for predicting a fraudulent user in Rating Platforms [7]. This classification helps determine fake ratings which are nothing but the reliability of a rating. Three interdependent metrics are calculated: fairness of a user, reliability of a rating, and goodness of a product. Computing these scores for all users, ratings, and products is achieved using the Rev2 algorithm by combining network and behavior properties. It characterizes each user and product by Fairness(F(u)), Goodness(G(p)), and reliability(R(u,p)) measures.

Another implementation of this data set is Weighted Sign prediction [9] where the signs of the weights of edges represent the trust/distrust factor. This model completely focuses on characterizing the node based on 2 metrics called Fairness and Goodness, which are used recursively. These metrics in turn act as features for the regression model to predict the trust factor. However, it can be inferred here that the node or a user's trust factor is identified by only their historic transactions and ratings. But, implementing a novel approach to establishing the trust factor takes all the nodes that are connected to the user into consideration.

Our approach is designed using the key idea from both of these papers, where we are also estimating the trust factor of a user using both the ratings given(credibility) and the ratings received(goodness of a user). This eliminates the possibility that bad/good reviews given by fraudulent users affect the trust factor of the user and relatively computes the credibility of the rating.

3 DATA EXPLORATION

3.1 Dataset

The dataset used for this project is a real-time dataset generated by the trades made over the Bitcoin platform called Bitcoin OTC. It represents the who-trust-whom network with nodes being the users and directed edges pointing in the trade direction. Whenever a trade is established between users, they rate the other user on a scale from -10 to 10, and sometimes reviews in the form of text are also posted. Here, we deal with 36,104 such transactions, from which a Directed graph can be built. Apart from the transactional data, we also have node-specific data where the number of positive ratings and the number of negative ratings given and received are aggregated for every user in the network. We leverage these aggregations representing 5980 users to build the User specific persona and calculate the trust factor for a user which ranges from 0 to 1, we then convert these trust into binary labels 0 and 1 with a threshold of 0.55.

3.2 Data Pre-processing

Before we begin analyzing the network and user data, it is important to ensure that the data is "clean", which means that the network data or the node's data should be noise free. We define noise in transactional data when the features of a transaction have inconsistent data. eg:- rating column having values < 1 or >10. Such noisy transactions and the transactions for which important features like rating, sender, or receiver are missing are identified to make up to 1.6% of the transactions and are dropped from the network. Here, it is important to note that the "Notes" feature which represents the reviews written by the sender/receiver is allowed to have missing values because not all users who are involved in the transactions write reviews. As part of cleaning the data, we also deal with duplicate data. Here, it is possible for repeating transactions but with different timestamps. Also, in this step, we look for redundant information and drop such features to make sure that the model does not learn the same features twice. Once such feature aggregates the total rating per user, which denotes the difference between positive ratings and negative ratings. We consider each of these features(No. Of positive ratings and No. of Negative ratings) individually, so the aggregation is redundant and we remove it. Further, the data contains duplicate nodes and edges, we remove those duplicate nodes and edges from the graph. Further, there are nodes that do not have any transactions and edges for which there are no nodes in the graph, we also remove such nodes and edges from the graph.

3.3 Exporatory Data Analysis

It is a crucial step to spend a significant amount of time understanding the data and how each feature is correlated to the other. The report suggests that all the transactions are made in the year 2013 and almost equally distributed among all the months. Rating for a transaction can range from -10 to 10, indicating negative to positive reviews intuitively. We have notes/reviews for 89% of the data, which can be used for extracting sentiment and the review's sentiment can be compared with the rating. Out of 35k+ transactions, a significant amount of ratings are given and received by a single user called, "sturles" (2%), followed by "GlooBoy" and "BCB" (1% each). However, these do not contribute to outlier data as all of them are legit transactions. Further, the ratings are distributed unequally over the range -10 and 10, the maximum of the ratings given are in the values 1,2,-10 which can lead to data imbalance problems in the graph.

3.4 Feature Transformation

For training a model, it is established that more learnings and patterns can be recognized from numerical features. So, as part of feature transformation, we transform the "notes" column i.e. review in the form of text as 0 or 1 indicating the sentiment of the review. For this, we build an Unsupervised Sentiment Analysis model to cluster the reviews as positive and negative reviews using the K-plus-plus clustering algorithm[5]. Further, we utilize Label Encoding and One-Hot Encoding to convert all non-numeric categorical columns into numeric columns.

Algorithm 1 Unsupervised Sentiment Analysis

```
  Input: Id(int), Review(str)
  Output: Id(int), Sentiment(0 or 1)
  reviews
          Step 1: Text PreprocessingClean noisy characters that
          do not add semantic meaning Remove stop words - "not"
          ("not" is significant here for determining the sentiment)
          Word Lemmatization
    Step 2: Word Embedding
          (1) (2) (3) (1)  Word2Vec to embed each word into numer-
          ical features
    Step 3: K++ Clustering
          (1) Initialize k=2 for 2 clusters (positive and negative)
          (2) Seeding clusters initially with positive(eg. Excellent,
          good, safe, legit) and negative words(eg. Bad, worst,
          scam etc)
          (3) Arbitrarily choose one of the word embeddings from
          seed data as each centroid
          (4) Repeat Until Convergence Criteria is met
                • Assign each item to a cluster that has the cluster
                  centroid
                • Calculate the new mean for each cluster
    Step 4: Transform the "Review" feature in the form of sen-
    tences to an array of cluster assignments for each word Step
    5: Calculate "Positivity Score" and "Negativity Score" based
    on the cluster assignments of words and assign "Positive" or
    "Negative" labels to sentences or 0  1
```

3.5 Feature Extraction

All Graph Neural Network models expect node features and a graph network as input. Nodes and edges represent a graph network. We represent this network in the form of an Adjacency matrix representing the transactions. We also build a feature matrix with Nodes and respective historic data like No. of positive ratings given, no. of negative ratings given, etc. One key feature for a node would be the "Trust Factor" which is derived using a novel approach to considering ratings, the credibility of ratings, and other factors into consideration. This computation is discussed below in detail. With this step, we conclude that the data is ready for the model in the Proposed Approach.

4 PROPOSED SOLUTION

4.1 Trust Factor Equation

The given data set is a graph of nodes and edges, we derived node characteristics like total rating, the total number of positive ratings given, the total number of the negative rating given, the total number of positive ratings received the total number of negative ratings received during the feature extraction phase. The graph neural network also needs node labels (trust factor) to create the supervised node classification model. We try 2 approaches that give us different trust factor results and use the labeling function that provides the best distribution of trust factors between 0 and 1. For labeling the data we create different tagging functions that aggregate all the ratings received and sent.

Algorithm 2 Trust Factor Using Timestamp and Sentiment

TF = F(s_rating, r_rating, timestamp, sentiment)
WHERE -

- TF = Trust Factor
- s_rating = sent rating
- r_rating = received rating
- timestamp = timestamp of sent or received rating
- sentiment = sentiment of review

TF = Credibility(s_rating,timestamp,sentiment) + Goodness(r_rating,timestamp,sentiment)

$$Credibility = AVG(\frac{s\_rating * timestamp\_rating}{current\_time} * (1 + \alpha * sentiment))$$

$$Goodness = AVG(\frac{r\_rating * timestamp\_rating}{current\_time} * (1 + \alpha * sentiment))$$

Algorithm 3 Trust Factor Using Rating

TF = F(s_rating, total_rating, no_positive, no_negative)
WHERE -

- TF = Trust Factor
- s_rating = sent rating
- total_rating = total node rating
- no_positive = number of positive ratings received
- no_negative = number of negative ratings received

TF = Credibility(s_rating) + Goodness(total_rating, no_positive, no_negative)

$$Credibility = AVG(s\_rating)$$

$$Goodness = total\_rating/(no\_positive + no\_negative + 1)$$

We normalize the calculated trust factor within a range of 0 to 1 after calculating for each node. We come to the conclusion that, the second tagging method provides uniform distribution across values between 0 and 1, we thus utilize the second tagging metric for classification. Once we find out the trust factor labels, we then convert them into binary labels 0 and 1 with a threshold of 0.55.

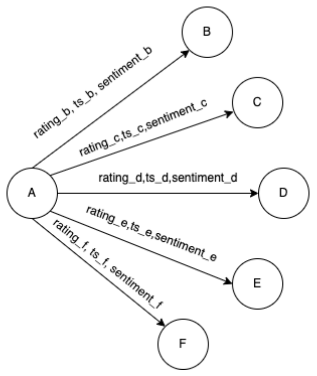
A
B
C
D
E
F
rating_b, ts_b, sentiment_b
rating_c,ts_c,sentiment_c
rating_d,ts_d,sentiment_d
rating_e,ts_e,sentiment_e
rating_f, ts_f, sentiment_f

Figure 1: Credibility Graph

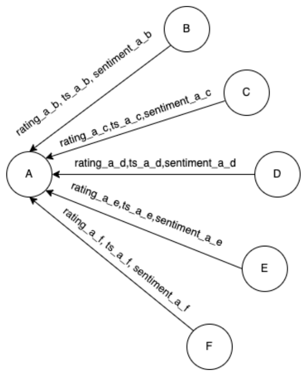
A
B
C
D
E
F
rating_a_b, ts_a_b, sentiment_a_b
rating_a_c,ts_a_c,sentiment_a_c
rating_a_d,ts_a_d,sentiment_a_d
rating_a_e,ts_a_e,sentiment_a_e
rating_a_f, ts_a_f, sentiment_a_f

Figure 2: Goodness Graph

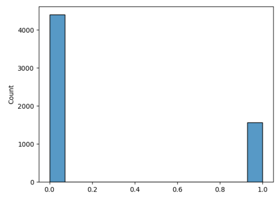

Figure 3: TF Algo 1 Distribution

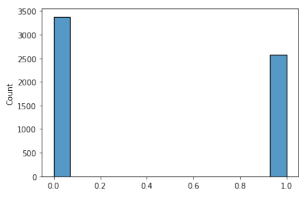

Figure 4: TF Algo 2 Distribution

4.2 Approach

Once we calculate the trust-factor label for each node, we add this to the node characteristics as labels for the node. This trust-factor calculation takes into consideration all the ratings for transactions made by the node and does not consider the long path of transient relations between nodes that have not made any transactions yet. Hence, we utilize the computation from Graph Neural Networks to calculate this transient relation between all the nodes.

4.3 Baseline Model: Graph Convolution Network

For the baseline model, we plan on implementing Graph Convolution Network[6] for the node classification. In recent times, Neural networks have gained a lot of popularity in solving real-world challenges even with high dimensional unstructured data. One of the popular neural network implementations is the Convolutional neural network, which is widely used on image data. CNNs are proven to be very efficient when the data is structured (Euclidean) and computes the set of weights, commonly known as filters or kernels, to learn the features from the neighboring cells. Whereas Graph neural networks are used when the data is unstructured or non-Euclidean, ideally when weights are determined based on all the input vectors. In our use case, one can say that it would be appropriate to use the GNNs because we have a network of nodes. But, in detail, our problem of node classification is based on edge weights and trust factors of the neighboring nodes only I.e. connected nodes only. (Note that the connection between nodes here represents nothing but the transaction). This is where the Graph Convolutional Network comes in. The equation for learning and updating weights in GCNs is very similar to the CNNs, but with slight changes to accommodate the graphical data. Using GCNs has shown revolutionary results in message passing and signal processing. The idea behind GCNs is quite simple. First, each node aggregates the information about the features based on all the connected nodes, and second, the resultant vector is passed through the dense neural network layers.[5] More on the model layers and hyper-parameters will be discussed during the next phase of the report.

5 METHODS

We developed multiple Graph Neural Networks using state-of-art algorithms using the dataset and proposed trust-factor evaluations. We compare each of these models and evaluate them in terms of various metrics like accuracy, f1 score, ROC curve, class distribution, etc. For all the methods, we experimented with various combinations of the activation functions from tanh, relu, and sigmoid. The reported model results are the best results each of the models could give based on the hyper-parameters, number of epochs, number of channels, and activation function.

5.1 Graph Convolutional network

As discussed in the previous section, Graph Convolutional Networks[6] is an efficient variant of convolutional neural networks which operate directly on the graphs by scaling linearly in the number of graph edges. The hidden layers are represented in terms of both the neighborhood of the nodes and node attributes. We developed a 2-layer GCN for node classification on the graph with a symmetric adjacency matrix A(here, binary). By definition, the graph convolution layer computes the updated node attribute matrix as follows:

$$\mathbf{X}' = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{W} + \mathbf{b}$$

This GCNConv model serves as the baseline model throughout the rest of the discussions and evaluations in the paper.

5.2 Graph Attention Networks

GATConv[15] is another graph neural network architecture that leverages masked self-attentional layers to address the shortcomings of spectral-based graph neural networks. By stacking 2 layers of GAT with a GCN, we are able to attend to the nodes' neighborhoods' features by implicitly specifying different weights on different nodes in a neighborhood. In other words, by performing masked attention, the model allows every user to attend to every other user, dropping all structural information. This approach in turn helps in a new node classification for which edges are not present in the network solving the Inductive prediction challenge. The coefficients computed by the attention mechanism are expressed as:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\overrightarrow{\mathbf{a}}^T\left[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j\right]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\overrightarrow{\mathbf{a}}^T\left[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k\right]\right)\right)}$$

The convolution is computed using the above attention coefficient (α) to weight the adjacency matrix instead of using the normalized Laplacian:

$$\mathbf{X}' = \alpha X W + b$$

5.3 Gated Graph Sequence Neural Networks

GatedGraphConv[12] modifies Graph Neural Networks with the adoption of Gated Recurrent Units and other optimization techniques. This model is especially favorable to inductive biases relative to sequential graph networks. To make batch predictions, GatedGraphConv is run over each graph simultaneously and is aggregated across the batch of graphs. These node annotations outputs are updated for each graph independently so that the training process can be decomposed to a single output single graph training. With the input of a sparse adjacency matrix and an additional parameter of n_layers, i.e. the number of iterations with the GRU cell, This layer computes $\mathbf{x}'_i = \mathbf{h}_i^{(L)}$ where:

$$\begin{aligned}
\mathbf{h}_i^{(0)} &= \mathbf{x}_i \| \mathbf{0} \\
\mathbf{m}_i^{(l)} &= \sum_{j\in\mathcal{N}(i)} \mathbf{h}_j^{(l-1)}\mathbf{W} \\
\mathbf{h}_i^{(l)} &= \text{GRU}\left(\mathbf{m}_i^{(l)}, \mathbf{h}_i^{(l-1)}\right)
\end{aligned}$$

where GRU is a gated recurrent unit cell. The results from GGS-NNs show that they have desirable inductive biases even with intrinsic graphs, like the current use case of the bitcoin network.

5.4 Topology Adaptive Graph Convolutional Network

Based on the fact that GCNs require approximation to the convolution to alleviate the computational complexity, resulting in the performance loss, TAGConv[2] is defined by a set of K-localized fixed-size learnable filters to perform convolutions on graphs. This model is very much consistent with the convolution in CNNs with properties like local feature extraction and weight sharing but varies as it needs only polynomials of the adjacency matrix on the vertex domain, making it more efficient with less complex computations. However, the output is still similar to CNNs, where the weighted sum of the convolutional results is computed by the size-k filter sliding on the graph following a fixed order of indices. Also, TAGCN is known to leverage information at a farther distance and scales well even for large-scale graphs like the network in the current bitcoin OTC dataset. This graph coverage can be restricted with a parameter 'k', where the layer has a k-hop neighborhood for each node. This layer of computer,

$$\mathbf{Z} = \sum_{k=0}^{K}\mathbf{D}^{-1/2}\mathbf{A}^k\mathbf{D}^{-1/2}\mathbf{X}\mathbf{W}^{(k)}$$

Since TAGCN doesn't require to the approximation of the convolution, it is expected to exhibit better performance, making the computations also easier in each epoch.

5.5 Graph Isomorphism Network

With a series of experiments to check if Graph Neural Networks are actually efficient or not, it has become evident that GNNs provide very little understanding of the representational properties and learning. Graph Isomorphism is an analyzing framework for visualizing GNNs graph structures, which cannot be achieved by models like GCN[6] and GraphSAGE[4]. The GINConv[17], as a framework, represents the set of node attributes of the nodes' neighbors as a multiset, and the GNNs are nothing but the aggregations on these functions. This model is based on the Weisfeiler-Lehman test, where the graph isomorphism problem asks whether two graphs are topologically identical. In this context, two graphs are termed as non-isometric if, at some iteration, the labels of the nodes between the two graphs differ. To model a GINConv on this basis, injective multiset functions for the neighborhood aggregation, i.e. deep multisets, using which we obtain the maximum discriminate power among GNNs. This layer computes for each node i:

$$\mathbf{x}'_i = \text{MLP}\left((1+\epsilon)\cdot\mathbf{x}_i + \sum_{j\in\mathcal{N}(i)}\mathbf{x}_j\right)$$

The node embeddings learned by the GINConv can be utilized directly for the tasks like node classification in the proposed use case, using a simple 'readout' function, that produces the embedding of the entire graph.

5.6 General Graph Neural Network

Studying the general design space of GNNs, the Cartesian product of the different design dimensions, General GNNs[18] is a more generalized approach for novel graphs, which allows insights to be distilled from a huge number of model task combinations. Design space is generally defined by crucial aspects like how message passing works within each layer, how the layer is organized in a network, and finally the training configurations. The proposed model GraphGym facilitates designing the GNN on the above parameters with additional enhancements like supporting all connectivity patterns across layers e.g. Attention, custom split on the graph, and reliability on reproducibility. The generalized layer definition is given as follows,

$$\mathbf{x}_i' = Agg(Act(Droput(BN(x_j W + b))), j\epsilon N(i))$$

where Agg is an aggregation function for the messages, Act is an activation function, Dropout applies dropout to the node features, and BN applies batch normalization to the node features.

6 MODEL PARAMETERS

A combined training framework for the defined models has been developed and these models, all of them being different variants of GNNs, still share certain common parameters. All the parameters may not be applicable for all the models, but such an initialization helps in setting the default values for each neural network. They are, channels in the first layer 1024, and the number of epochs is set to 200. However, all the models might not necessarily train for 200 epochs if they converge towards optimization earlier. We can prevent this with a parameter called Early Stopping Patience, set to 10, which means the model training is terminated after 10 consecutive non-increasing values of loss. However, the convergence is also likely to depend mostly on the learning rate, here alpha = 0.001. Since the number of classes is 2, i.e binary classification, some GNN variants very rapidly learn all the features, and sometimes the anomalies also, lead to the infamous overfitting problem. This is avoided with the regularization parameter called Dropout[5], where the dropout rate, 0.5 represents the fraction of random neurons to be zeroed out in each epoch. Finally, an l2-regularization rate of $5e-4$, also called ridge regularization, where the coefficient acts as the penalty term to the loss function is also added.

Model	Activation	Aggregate	Parameters
TAG	tanh	mean	3,724,288
GCS	tanh	sum	1,064,960
GCN	linear	sum	532,480
General	tanh	sum	1,065,992
GIN	tanh + sigmoid	sum	2,907,139
Gated	tanh	sum	1,049,088

7 MODEL TRAINING GRAPHS

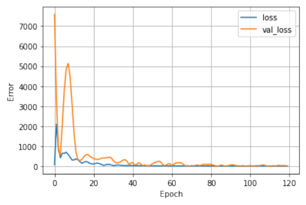

Figure 5: TAG Training Graph

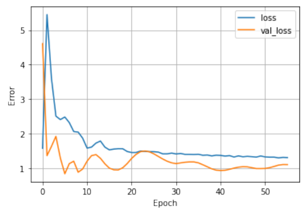

Figure 6: GCS Training Graph

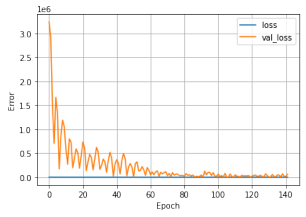

Figure 7: GCN Training Graph

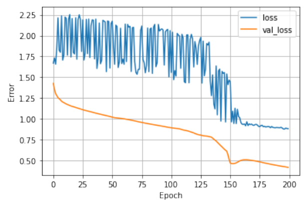

Figure 8: General Training Graph

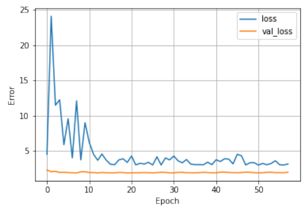

Figure 9: GIN Training Graph

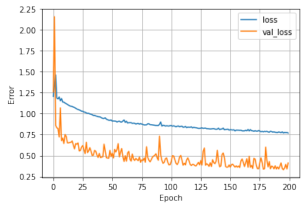

Figure 10: GatedGraphConv Training Graph

8 MODEL EVALUATION AND RESULTS

8.1 Evaluation Metrics

The binary classification problem is being addressed by the Graph Neural Networks, whose training and validation accuracy are reported by the model itself. However, the model performance is well assessed based on its performance against unseen data, i.e. test data. In the following subsection, the performance of each of the GNN variants is reported in terms of Accuracy and F1-score which are primary measures of classification accuracy. These values are calculated using the confusion matrix representation of the classification

results[5]. Confusion Matrix is given by,
1. True Positives(TP) – The actual value was positive and the model predicted a positive value.
2. True Negative(TN) - The actual value was negative and the model predicted a negative value.
3. False Positives(FP) - The actual value was negative but the model predicted a positive value.
4. False Negatives(FN) - The actual value was positive but the model predicted a negative value.
With the above values, a confusion matrix is represented, below

		Actual Values	
Predicted Values		Positive Trust	Negative Trust
	Positive Trust	TP	FP
	Negative Trust	FN	TN

Using the above values, the F1 score combines the precision and recall of a classifier into a single metric by taking their harmonic mean.

$$F_1 = \frac{2}{\frac{1}{\text{recall}} \times \frac{1}{\text{precision}}} = 2\times\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}$$

Similarly, Accuracy is defined as the ratio of correct classifications, represented by,

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

8.2 TAG Model

F1 Score	0.6520497311827957
Accuracy Score	0.6520497311827957

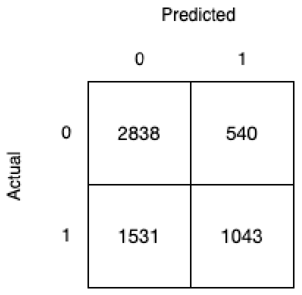

Figure 11: TAG Confusion Matrix

8.3 GCS Model

F1 Score	0.6560819892473119
Accuracy Score	0.6560819892473119

8.4 GCN Model

F1 Score	0.4324596774193548
Accuracy Score	0.4324596774193548

8.5 General Conv Model

F1 Score	0.5036962365591398
Accuracy Score	0.5036962365591398

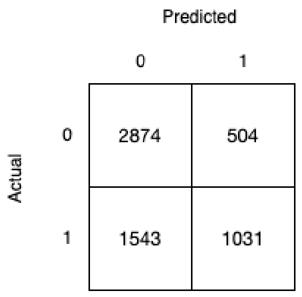

Figure 12: GCS Confusion Matrix

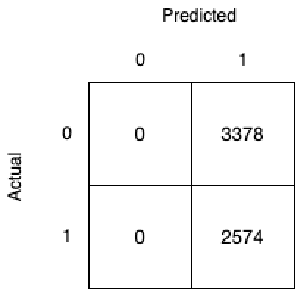

Figure 13: GCN Confusion Matrix

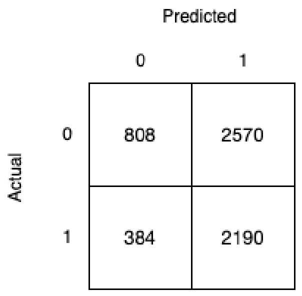

Figure 14: General Conv Confusion Matrix

8.6 GIN Model

F1 Score	0.5675403225806451
Accuracy Score	0.5675403225806451

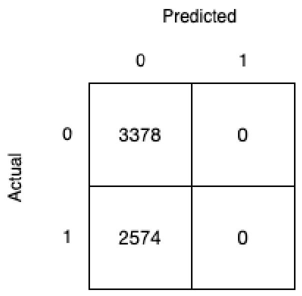

Figure 15: GIN Confusion Matrix

8.7 Gated Conv Model

F1 Score	0.6434811827956989
Accuracy Score	0.6434811827956989

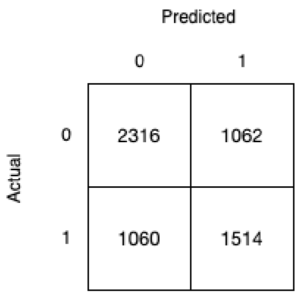
Predicted
0
1
Actual
0
1
2316
1062
1060
1514

Figure 16: Gated Conv Confusion Matrix

9 TEAM AND WORK

9.1 Shriram Balaji Athreya

- Literature Survey
- Data Cleaning and Pre-processing
- GNN Model Training and Validation
- Graph Plot with Trust Factor Distribution and Train/Test with epochs
- Documentation

9.2 Sindhu Inuganti

- Literature Survey
- Data Engineering
- Unsupervised Sentiment Analysis For Reviews
- GNN Model Training and Validation
- GNN Model Hyper-parameter tuning
- Documentation

10 PROJECT SCHEDULE

(1) Literature Survey (Sept 2, 2022 - Sept 24, 2022)
(2) Data Engineering (Sept 25, 2022 - October 10, 2022)
(3) Feature extraction and Transformation (October 11, 2022 - Oct 20, 2022)
(4) GNN Model Creation, Training, and Validation (Oct 21, 2022 - Nov 15, 2022)
(5) GNN Model Hyper-Parameter tuning (Nov 15, 2022 - Nov 30, 2022)
(6) Final Paper Documentation (Dec 1, 2022 - Dec 12, 2022)

11 CONCLUSION

- This submission has been done as part of the final project report.
- Literature Survey has been performed for the problem.
- Unsupervised problem has been converted to a supervised node classification problem by utilizing labeling functions to initialize the trust factors with a value.
- Feature Extraction and Data Transformation is completed.
- Training, Validation, and Testing of Baseline, as well as multiple other models, have been done on the node classification problem.
- These models learn generalizable features for nodes given their previous transactions and further predict the trust factors for unseen nodes and transactions.
- Model Evaluation has been performed for all the selected models which shows their performance over the given graph and edges.
- This solution can be further used in any financial transaction graph system and extended for further study.
- Future Scope of this project is to try further multiple labeling techniques for the graph and develop a robust graph neural network as a hybrid from the selected models to improve the prediction score for the model.
- Any boosting or stacking mechanisms to further improvise the accuracy has not been performed and this along with grid-search hyperparameter tuning can be part of the future scope.
- Further, we plan to implement multi-class classification and also regression models to predict the multi-labeled trust factor and trust probabilities of the models.

REFERENCES

[1] Steven David Brown. 2016. Cryptocurrency and criminality: the bitcoin opportunity. *The Police Journal*, 89, 4, 327–339.
[2] Jian Du, Shanghang Zhang, Guanhang Wu, Jose M. F. Moura, and Soummya Kar. 2017. Topology adaptive graph convolutional networks. (2017). DOI: 10.48550/ARXIV.1710.10370.
[3] Thomas DuBois, Jennifer Golbeck, and Aravind Srinivasan. 2011. Predicting trust and distrust in social networks. In *2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*. IEEE, 418–424.
[4] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. (2017). DOI: 10.48550/ARXIV.1706.02216.
[5] Jian Pei Jiawei Han and Hanghang Tong. 2022. *Data Mining: Concepts and Techniques (4th ed)*. Morgan Kaufmann.
[6] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. (2016). DOI: 10.48550/ARXIV.1609.02907.
[7] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and V.S. Subrahmanian. 2018. Rev2: fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (WSDM '18). Association for Computing Machinery, Marina Del Rey, CA, USA, 333–341. ISBN: 9781450355810. DOI: 10.1145/3159652.3159729.
[8] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. 2018. Rev2: fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 333–341.
[9] Srijan Kumar, Francesca Spezzano, V. Subrahmanian, and Christos Faloutsos. 2016. Edge weight prediction in weighted signed networks. In (Dec. 2016), 221–230. DOI: 10.1109/ICDM.2016.0033.
[10] Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. 2016. Edge weight prediction in weighted signed networks. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 221–230.
[11] Xiaoxiao Li et al. 2020. Explain graph neural networks to understand weighted graph features in node classification. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer, 57–76.
[12] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. (2015). DOI: 10.48550/ARXIV.1511.05493.
[13] Yi Qian and Sibel Adali. 2014. Foundations of trust and distrust in networks: extended structural balance theory. *ACM Transactions on the Web (TWEB)*, 8, 3, 1–33.
[14] Jeyakumar Samantha Tharani, Eugene Yougarajah Andrew Charles, Zhé Hóu, Marimuthu Palaniswami, and Vallipuram Muthukkumarasamy. 2021. Graph based visualisation techniques for analysis of blockchain transactions. In *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, 427–430. DOI: 10.1109/LCN52139.2021.9524878.

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. (2017). DOI: 10.48550/ARXIV.1710.10903.

[16] Jiajing Wu, Jieli Liu, Yijing Zhao, and Zibin Zheng. 2020. Analysis of cryptocurrency transactions from a network perspective: an overview. *CoRR*, abs/2011.09318. https://arxiv.org/abs/2011.09318 arXiv: 2011.09318.

[17] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? (2018). DOI: 10.48550/ARXIV.1810.00826.

[18] Jiaxuan You, Rex Ying, and Jure Leskovec. 2020. Design space for graph neural networks. (2020). DOI: 10.48550/ARXIV.2011.08843.