Detection of Circular Trade

Fraud Analytics CS6890 - Assignment 1

M Anand Krishna
M.Tech CSE
IIT Hyderabad
Hyderabad, India
cs22mtech14003@iith.ac.in

Akash K S
M.Tech CSE
IIT Hyderabad
Hyderabad, India
cs22mtech11012@iith.ac.in

K Manish
M.Tech CSE
IIT Hyderabad
Hyderabad, India
cs22mtech11008@iith.ac.in

A SivaSai
M.Tech CSE
IIT Hyderabad
Hyderabad, India
cs22mtech11013@iith.ac.in

***Abstract*—Fraud detection is a crucial task in many industries, such as finance, insurance, and e-commerce. In this report, Multi-graph is made using the representation of the dataset. We convert the multi-graph into a directed graph with edge weights, which allows us to capture the strength of relationships between entities in the dataset.The amount of fraudulence is calculated by a function which will find suspicion score based on the 2-cycles and 3-cycle count and trading amount between a pair of traders.This quantity is stored at locations we obtain by performing hash function on nodes of corresponding cycles. We will then construct a undirected graph with edge weights based on suspicion score and number of 2-cycles and 3-cycles between nodes. We then use node2vec to learn embeddings for the nodes in the undirected graph and apply DBSCAN algorithm to identify dense regions of nodes, which can indicate fraudulent behavior.**

***Index Terms*—DBScan,node2vec**

I. Introduction

Fraudulent behavior can result in significant financial losses and damage to a company's reputation. Therefore, there is a pressing need for accurate and efficient fraud detection methods. In recent years, graph-based methods have emerged as a promising approach for fraud detection due to their ability to capture complex relationships between entities in a dataset. In our report, we attempt to construct a undirected graph, with trading data given to us, which can capture necessary information required to find circular trading. We will explain how this graph was constructed in detail in further sections of this report. We will then apply node2vec on the constructed undirected graph to learn node embeddings and apply DBSCAN agorithm to identify dense regions of nodes that can indicate fraudulent behavior.

II. Problem Statement

The problem addressed here is the detection of Circular Trading. Circular trading is a fraudulent activity where a group of individuals or companies trade stocks or other securities among themselves to create a false appearance of trading activity and inflate the value of the securities. We will cluster the sets of dealers who are engaging in circular trading.

III. Description of the Dataset

The dataset given is iron dealers data which contains information about invoices between sellers and buyers and the value of transaction. Each row in the dataset represents one invoice and contains the following information
Seller ID: The unique identifier for the seller
Buyer ID: The unique identifier for the buyer
Value: The value of the invoice, which represents the amount of money exchanged between the seller and the buyer for the goods or services provided.

This data can be used to identify potential cases of circular trading.

IV. Algorithm used

Algorithm 1 Find_Suspicion_Score()

Require: Nodes, edge weights of 2-cycles or 3-cycles
Ensure: Maximum $suspicion\_score$ if circular trading exists in cycle

Let $weights \leftarrow edge\,weights$ be the list of edge weights
Let $Threshold$ be the threshold to decide suspicion.
Let $norm\_weights$ be the list to store normalized wights.
for each $weight_i$ in $weights$ **do**
    $norm\_weight[i] = \frac{weight_i}{\sum_j weight_j}$
end for
$std\_dev = Standard\_Deviation(norm\_weights)$
if $(1 - std) \geq Threshold$ **then**
    $suspicion\_score = (1 - std) * avg(weights)$
else
    $suspicion\_score = -1$
end if
Return $suspicion\_score$

Here, we normalize the weights of edges in cycle and find standard deviation of normalized weights. If standard deviation is less, it means there is more chance for circular trading. So, we will compare (1 - std) with some threshold value which is a hyperparmeter. If it is more than threshold, we will multiply (1-std) with average of weights. This will gives maximum suspicion score, indicating high chance of circular trading.

Else, we will just give suspicion score of -1, indicating there is no circular trading.

Algorithm 2 Detection of Circular Trading

Require: Dataset $data$ with $BuyerID$, $SellerID$ and $valueoftransaction$.

Ensure: Clusters with Dealer IDs engaged in Circular Trading.

1: Load dataset $D = \{seller\_id, buyer\_id, value\} \leftarrow iron\_dealers\_data.csv$
2: Convert D in to a multi graph G^1 with nodes as $seller\_id$'s and $buyer\_id$'s and edges with weight as $value$
3: Find all 2-cycles on G^1 and store them in the list L_2.
4: For every cycle in L_2 find a $suspicion\_score$ using $Find\_Suspicion\_Score()$
5: Find all 3-cycles on G^1 and store the in the list L_3.
6: For every cycle in L_3 find a $suspicion\_score$ using $Find\_Suspicion\_Score()$
7: Create a hash function which takes two nodes and gives unique index for every pair.
8: For every pair of nodes in L_2, find an index using $hash(node1, node2)$
9: Store $suspicion\_score$ in that index's location. If that location already have some value, add new value to it.
10: Repeat $Step\,8$ and $Step\,9$ for L_3.
11: Create an undirected graph G^2 from $seller\_id$ and $buyer\_id$ values of D with edge values of -1.
12: For every node pair (u, v) in G^2, perform $hash(u, v)$.
13: If index is present in hash table, take value at that index and make it weight of (u, v). Else, the weight remains as -1.
14: Perform $Node2Vec$ on G^2 and obtain embeddings.
15: Perform $DBSCAN$ on embeddings and find the clusters. These clusters contains dealer's IDs engaged in circular trading.
16: Return the clusters.

V. Implementation Details

In this section, we provide an overview of the implementation details, steps to run, and hyperparameters used in the algorithm used to detect circular trading.

A. Implementation Details

The algorithm is implemented using Python programming language. Key libraries used for the implementation are:

- `pandas` for data manipulation and loading.
- `numpy` for numerical computations.
- `scikit-learn` for data preprocessing, clustering, and distance computations.
- `pyea` for using Genetic Algorithm Optimizer.

B. Steps to Run

The following steps outline the process to run the Detection of Circular Trading using Node2Vec and DBScan in a Jupyter Notebook

1) Install the required Python libraries, such as pandas, numpy, and scikit-learn, Node2Vec and DBScan if not already installed. You can use the command '`pip install pandas numpy scikit-learn`' for installation.
2) Download the input dataset `iron_dealers_data.csv` and save it in a suitable directory.
3) Open the provided `Assignment1.ipynb` file.
4) In the Jupyter Notebook, Change the file path in the following code block to match the location of the `iron_dealers_data.csv` file on your system, and execute the code block to load the dataset
5) Execute the remaining code blocks in the Jupyter Notebook sequentially, as they appear in the notebook, to perform the Circular Trading detection.
6) Observe the output for clusters with dealer IDs which is displayed as a scatter plot of clusters.

C. Hyperparameters

The following hyperparameters are used in various algorithms used to detect circular trading:

- Threshold, $Threshold$ = 0.6: To decide on the suspicion of fraud in 2-cycles or 3-cycles.
- Dimensionality of resulting node embeddings in Node2Vec $dimensionality = 4$.
- $walk\_length = 10$: The length of each random walk during sampling in Node2Vec.
- $num\_walks = 200$: The number of random walks to perform per node during sampling in Node2Vec.
- $p = 1$: The likelihood of returning to the previous node in the walk.
- $q = 0.5$: The likelihood of moving to a node that is far away from the previous node in the walk
- $window\_size = 10$:the maximum distance between the nodes that are used to predict the embedding of a node.
- $min\_count = 1$: This is the minimum count of nodes that are required for a node to be included in the model.
- $eps = 0.8$: the maximum distance between two points for them to be considered part of the same cluster.
- $min\_samples = 5$: the minimum number of points required for a group of points to be considered a cluster.

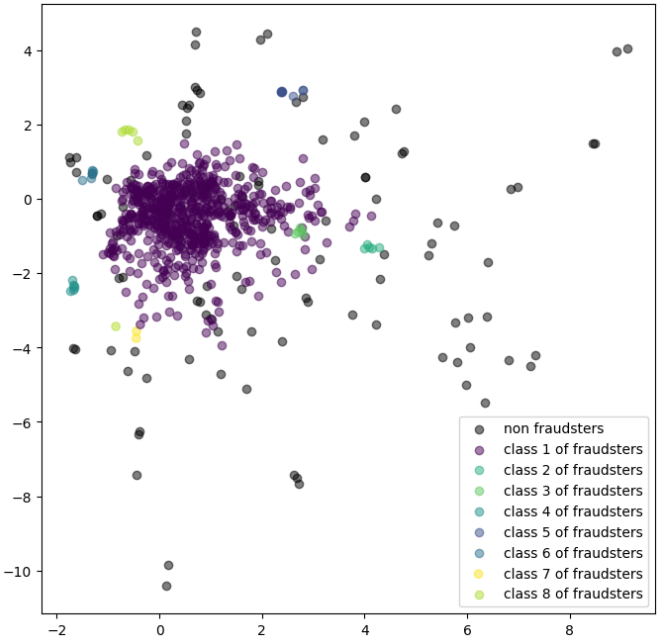

Fig. 1. Scatter Plot after applying DBScan algorithm

VI. Results

See the Fig. 1 which contains clusters obtained after applying DBScan algorithm on embeddings of undirected graph G^2. Fraud activity is directly propotional to the opacity of the colour of plots