Fraud Detection using Graph Features and Machine Learning

Aakash Rastogi
CIDSE, MCS
Arizona State University
Tempe Arizona USA
arastog9@asu.edu

Amruta Tapadiya
CIDSE, MCS
Arizona State University
Tempe Arizona USA
atapadiy@asu.edu

Kshitij Kashettiwar
CIDSE, MCS
Arizona State University
Tempe Arizona USA
kkashett@asu.edu

Pratik Barhate
CIDSE, MSCS
Arizona State University
Tempe Arizona USA
pbarhate@asu.edu

Richa Nagda
CIDSE, MCS
Arizona State University
Tempe Arizona USA
rjnagda@asu.edu

Vrushabh Jambhulkar
CIDSE, MCS
Arizona State University
Tempe Arizona USA
vjambhul@asu.edu

ABSTRACT

Nowadays, online frauds in banks are increasing manifold, due to which banks and customers have to face excessive financial losses. Traditional machine learning techniques fail to identify these fraudulent activities due to the sparse amount of fraudulent training data and the use of simplistic transaction features. Graph databases, on the other hand, can provide us with a lot more effective features that capture community relations between the nodes. These additional features combined with the existing Machine Learning can help us in analyzing the common patterns in the fraudulent transactions. In this project, we have used DeepWalk to generate feature vectors for the customer and merchant nodes in the graph database. Using these features as input to the classification model and sampling techniques like SMOTE and Self-paced Ensemble we achieved an increase in accuracy of 58% over the base model using basic features. Moreover, applying clustering algorithms like OPTICS, birch, and KMeans on these features resulted in distinguishing patterns.

Keywords - Deepwalk, graph features, fraudulent transactions, classification, clustering, sampling

1. INTRODUCTION

The goal of this project is to identify fraudulent transactions as they incur a huge amount of loss to the banks. The BankSim dataset is used which contains 5,94,643 records in total out of which 5,87,443 are normal payments and 7,200 fraudulent transactions. The merchants and customers present in the dataset generate a graph where the nodes represent these merchants and customers while the edges between them represent the transaction. Initially, some preprocessing is done on the data to get rid of biases and make the data consistent. In addition to this, since the data is skewed, sampling techniques like SMOTE and Self-paced Ensemble are used. Moreover, graph-based features are extracted so that they can be used while performing classification. The goal behind this was to improve the accuracy of the classification algorithms. Clustering techniques were also applied to the dataset to gain insights with respect to the fraud data points present in the dataset.

2. PROBLEM STATEMENT

The problem is to detect fraudulent activity in the bank transactions with the help of the Machine Learning algorithms and graph databases. Fraud detection in bank transactions using graph databases and machine learning algorithms is very similar to what is explained in [5]. Our goal is to detect all the fraud transactions with the least number of incorrect fraud classification. To achieve this, extracting different features from the dataset and training the Machine Learning algorithms based on the given features and the newly extracted features on the training set is helpful. Moreover, the Graph databases assist us in detecting the suspicious links and the complex fraud rings in between the nodes to detect all the fraud transactions present in the dataset.

3. RELATED WORKS

A major issue currently in dealing with graph databases is community representation. The challenging aspect here is dealing with overlapping data and clustering the community on the basis of it. [1] mentions a method in which a vector representation of each node in a graph is obtained by considering the structural role of the node in the graph. Here interaction of nodes with each other is divided into different personas which considers graph network behavior. For generating vector representation random walks from a node are taken to extract co-occurrences of nodes which captures the likelihood of a node to occur with other nodes. This method can be applied in transactions as well, where random walks can be taken from each node to generate multiple node embeddings of each node for each persona of the node.

In their paper [2], Perozzi et al. have introduced a novel technique called DeepWalk, to learn node embeddings that capture contextual information from a graph. These node embeddings encode relations between neighboring nodes in a continuous vector space which can be easily fed to classification or other statistical models. The advantages of using local exploration are the ability to use multiple random walkers to explore different parts of the network simultaneously and account for local changes in the network without having to relearn the entire representation model. The authors of this paper claim that the node representations obtained by DeepWalk can outperform all the base models, when labeled data is sparse, using about 60% less training data.

DeepWalk [2] has multiple parameters, and tuning those parameters manually may take time. [3] proposes to learn these hyper-parameters by using an attention model, such that during the training step the attention model directs the representation model towards network properties which are more important for classifying the nodes into target classes. The upstream objective of classification is coupled with the attention model improving the overall accuracy of the task over minimal available data.

4. DATASETS

BankSim dataset [4] consists of synthetic data generated from an agent-based bank payment simulator. This was extracted using a sample of aggregate transactional data provided by a bank in Spain. Statistical and a Social Network Analysis (SNA) of relations between merchants

and customers were used to develop and calibrate the data-generation model. The dataset contains both fraudulent and non-fraudulent transactions which are defined by properties such as "Customer", "Merchant", "Amount", "Category", "Fraud". The dataset contains 5,94,643 records in total out of which 5,87,443 are normal payments and 7,200 fraudulent transactions.

We will be using this dataset to generate a graph database and run G-SQL queries to build the graph representation learning model. Our dataset contains only 0.012% fraudulent transactions so to improve the classification algorithm we will make use of graph databases and extract graph-based features from them. Finally, we plan on resampling this data so as to counter it's non-uniform distribution.

5. SYSTEM ARCHITECTURE AND ALGORITHMS

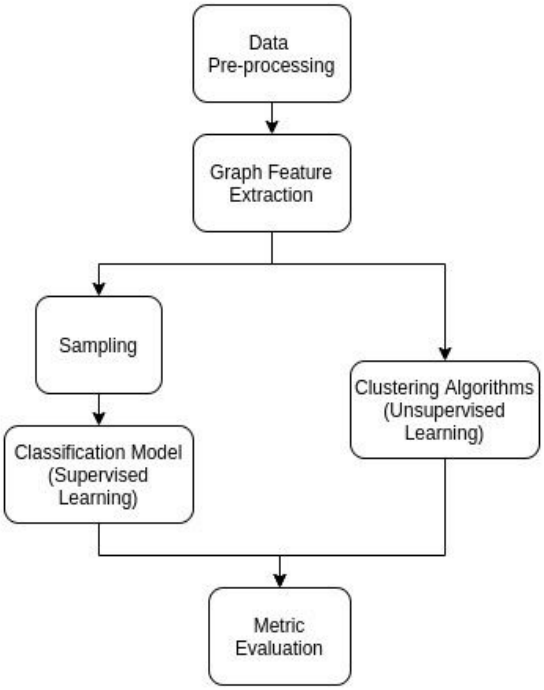

Figure 1. System Architecture

5.1. Data Pre-processing

The data we are using is structured data with nine feature columns and a target column. As shown in Figure 1, we start by pre-processing the data. We observed data is majorly clean with the following ten columns - *step, customer, age, gender, zipcodeOri, merchant, zipMerchant, category, amount, fraud*. We conducted some basic analysis to count the number of distinct values in those columns and found out that the columns *zipcodeOri* and *zipMerchant* have only one value across all rows, hence, we discard those two columns along with the *gender* column. We discard the *gender* column so that the trained models are not biased if the underlying data has some hidden patterns among the historical data.

Among the remaining columns, six are categorical features, except for the *amount* column. We converted the column into categorical features, namely *amount_cat* by dividing it into three ranges - i) *below_median* ii) *above_mean* and iii) *in_between*, with mean and median values being 37.89 and 26.9 respectively. Data Pre-processing steps are shown in Figure 2.

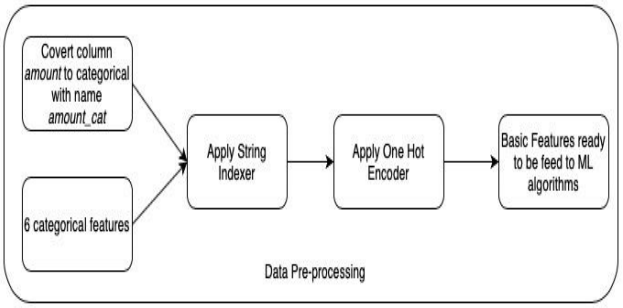

Figure 2: Data Pre-processing

All the machine learning algorithms take numerical values as input. Hence, we applied One Hot Encoding over all the columns.

5.2. Graph Feature Extraction

a. DeepWalk

Node embeddings capture the context information about its surrounding nodes by representing every node as a fixed-length vector. We used the DeepWalk algorithm [2] to learn node embeddings from graph representations. Random Walk is a technique to extract node sequences of fixed-length from the graph by traversing it. After generating node-sequences, we fed them to the Word2Vec model to get node embeddings as a 100-length feature vector. The vector representations learned are used in place of the customer and merchant columns as features to the classification model

b. Persona Classification

Persona classification deals with creating multiple personas of every node thus giving multiple node embeddings to represent a single node. In fraud detection, the graph contains directed edges between customers and merchants as shown in Figure 3.

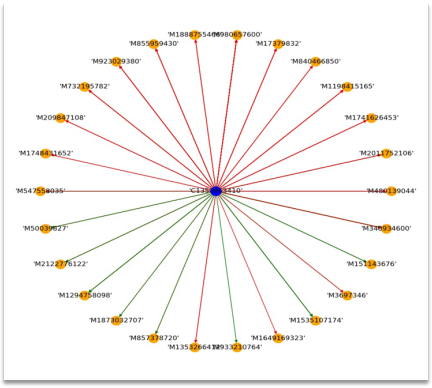

Figure 3. Directed Transaction Graph

Persona classification is useful for considering the overlapping participation of customers in various transactions. For each participation, a persona is created and a random walk is generated in it. But in dealing with transactions, the graph is a directed graph with the longest path length as one. So after creating personas for each node in a transaction, we obtain the same initial adjacency list. Therefore, persona classification is not suitable for fraudulent transaction classification.

5.3. Sampling

a. SMOTE

SMOTE [10] stands for Synthetic Minority Over-Sampling Technique. It is used to deal with the skewed class problem for our use case. The algorithm tries to form boundaries around the existing minority class instances and then synthesizes the new minority class points from the hyperplane formed by the boundaries.

Overall SMOTE is an oversampling technique that might affect the performance of the model for correctly identifying the majority classes and may increase the false classification of majority class into minority class.

b. Self-paced Ensemble

The Self-Paced Ensemble Model which is introduced in paper [6] is used for a highly skewed dataset. This model is making use of bagging and boosting techniques. Bagging is the technique in which the data is divided into multiple subsets from the original dataset, different models run on these subsets in parallel and the final predictions are

made by combining the predictions from all the models. Boosting is the sequential process in which first a base model is run over the subset from the original dataset. After these other models are run over the same subset to correct the errors of the previous model.

The Self-Paced Ensemble model makes use of bagging techniques such as SMOTEBagging and Underbagging and the boosting techniques such as SMOTEBoost and RUSBoost. All the features of the dataset are converted into the One Hot Vector, are passed through these bagging and boosting algorithms, and then the precision, recall, and F1 score are calculated for the dataset.

The dataset on which we worked was a highly undistributed dataset. There were two categories in the dataset: Fraudulent data and non-fraudulent data and the data associated with fraudulent activities was just 0.012% which made the data highly imbalanced and this model worked perfectly well on these kinds of the dataset.

5.4. Classification

As all the seven feature columns used as categorical, we planned on using tree-based models. We did end up using an implementation of Gradient Boosted Trees, namely, the XGBoost Classifier model.

Node Embeddings generated using Deepwalk along with *step*, *age*, *category*, and *amount_cat* were fed to the XGBoost model for training. The classification was done on sampled and unsampled data which helped in comparing the performance of sampling algorithms.

5.5. Clustering

a. OPTICS

OPTICS (Ordering Points To Identify the Clustering Structure) is a density-based clustering algorithm which is a generalized DBSCAN method that does not require us to provide a single ‘eps’ value but a range of values, allowing variable density extraction [8]. OPTICS builds a reachability graph to find cluster neighbors [8] which reduces the runtime of the algorithm drastically as compared to DBSCAN.

We ran OPTICS on the dataset to extract cluster relationships between the fraudulent and non-fraudulent data points. In this density-based clustering, we aimed to find if the fraud data points belong to a dense cluster or are marked as noise points. This will provide us knowledge about the nature of these transactions if they were outliers or not.

b. Birch

Birch is a clustering model that creates a Cluster Feature Tree [9] for the data points. As the algorithm creates the tree, it forms substructures called the Cluster Feature Nodes. The leaves of this tree structure divide the input set into subclusters which are then fed to the global clusterer. The advantage of using Birch is that we do not need to set the number of clusters exclusively, the algorithm uses the subclusters to determine and label the clusters.

c. KMeans

Kmeans is an unsupervised machine learning algorithm that forms clusters based on certain criteria. The goal is to minimize the sum of squares within a given cluster so that similar data points are clustered together. Since it is an unsupervised algorithm we need to specify the value of K i.e. is the number of clusters [7].

We performed KMeans on the dataset to gain insights specifically about the behavior of fraud data. The main goal while performing this algorithm was to see if the fraud data points lie in a specific cluster. If so, then this would mean that the fraud data points have certain features that distinguish it from the non-fraud data points.

6. EVALUATIONS

6.1 Classification

For all the results shown below Table 1, the XGBoost Classification model was used.

Type of Model	Accuracy (%)
Basic Features	40.22
Basic Feature + SMOTE	98.32
Basic Features + Self Paced Ensemble	97.15
Basic features + DeepWalk	64.92
Basic Features + DeepWalk + SMOTE	98.31

Table 1. Performance of different models

6.2 Clustering

The results of the three clustering algorithms performed are evaluated. Plots are generated for each of them to get a better understanding of the results. The green data points represent the normal transaction while the red data points represent the fraud transactions.

a. OPTICS

For the implementation of OPTICS, we set the *min_samples* parameter as *50* and tested the model on a subset data of 10000 samples in which there were 113 fraudulent transactions. With this configuration, we found that all the fraud transactions were marked as noise points as shown in Figure 4, however along with the fraud transaction many non-fraud transactions were also marked as noise points which made it result inconclusive.

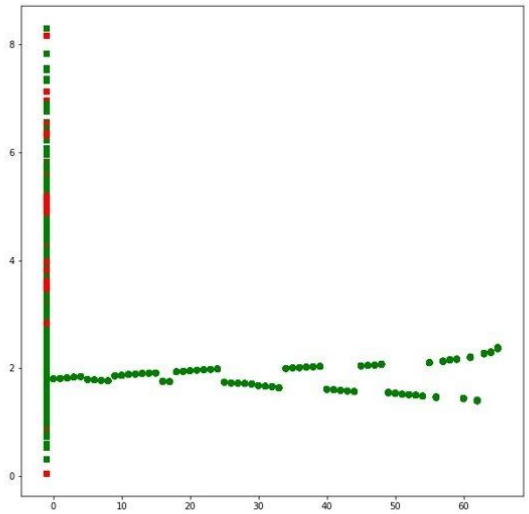

Figure 4. OPTICS

b. Birch

After we ran Birch on our dataset, it resulted in 9 different clusters by setting the *n_clusters* parameter equal to *None*

so that it generates the clusters automatically. From Figure 5, it can be seen that the fraud data points were clustered in 3 clusters only, but the overall accuracy was 82%.

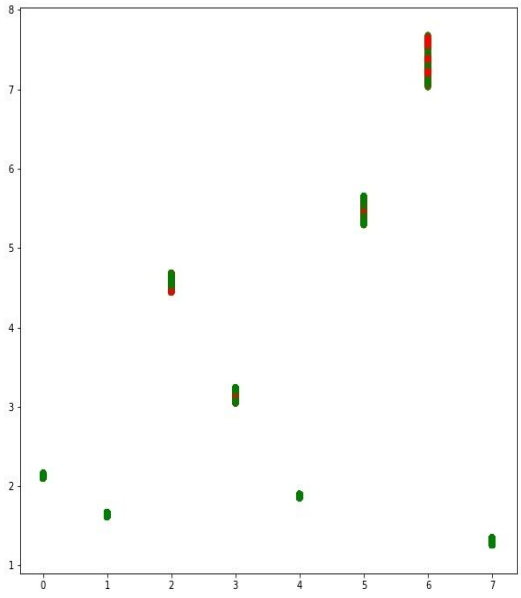

Figure 5. Birch

c. KMeans

KMeans is an unsupervised clustering algorithm, this means that we need to provide the value of K while performing clustering. For choosing the effective value of K the elbow curve was plotted. After plotting the curve it was noted that although the elbow point, as shown in Figure 6, was not clear, a range of points can be used for effectively performing clustering.

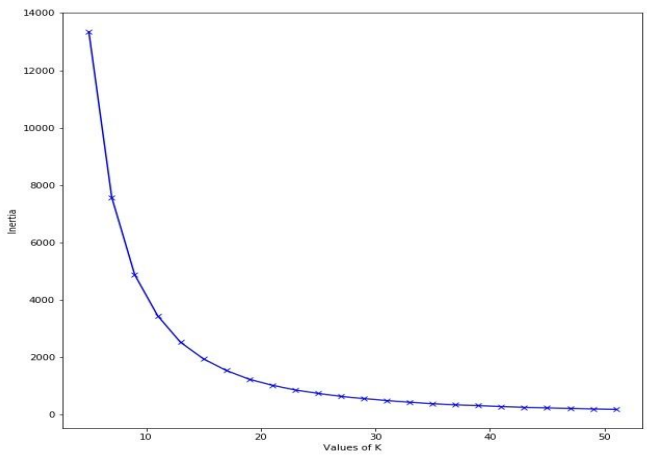

Figure 6. Elbow curve

Initially, we used K=10 to perform clustering. As seen in Figure 7 it can be observed that the fraud data points do not clearly fall in certain clusters. Looking at this result we tried with K=15. The results for this were significantly better. Figure 8 represents this output, the fraud data points lie in two clusters. Only 1.42% valid transactions out of the total data fell into one of these 2 clusters. 98.58% of the total data was clustered accurately.

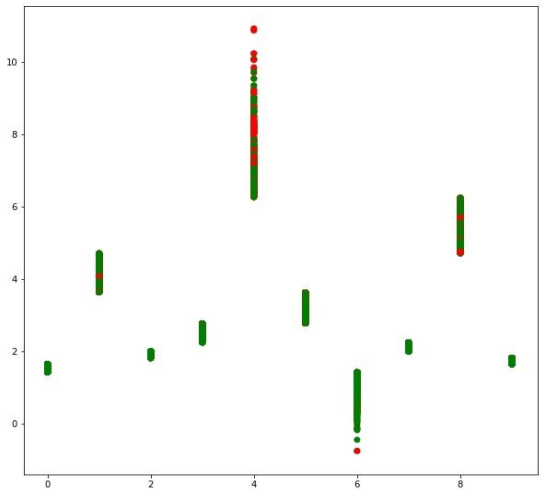

Figure 7. K Means with 10 clusters

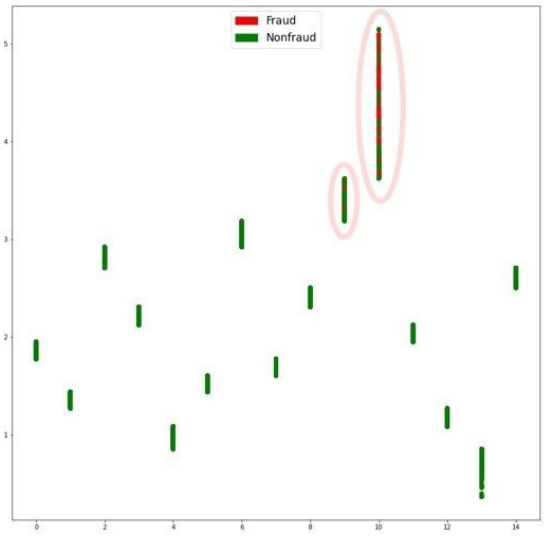

Figure 8. KMeans with 15 clusters

7. UI INTERFACE DESIGNS

Jupyter notebook was used for the majority of the implementation and analysis of the models. The interface for the notebooks can be accessed from a web-browser at *http://localhost:8888* which is the default web address, after executing the command - *jupyter notebook*, on your system.Sample Execution of notebook with results can be seen in Figure 9.

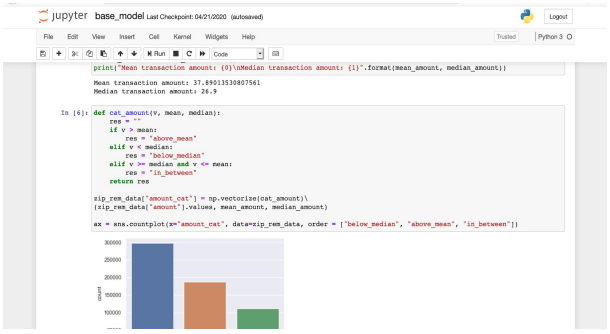

Figure 9. Sample running notebook Web UI

8. DIVISION OF WORK WITH TEAM MEMBERS CONTRIBUTION

Table 2 shows the detailed contribution of each member in the Group.

Member	**Task**	**Description**
Kshitij, Vrushabh	Visualization and graph plots	Converting the given data into the graph format
Pratik, Richa, Amruta	Data Preparation	Pre-processing of the given data
Amruta, Richa	Persona classification and	Implementation of Persona classification.
Pratik, Aakash	Sampling-Self Paced Ensemble	Implementation of Self-Paced Ensemble method for handling the skewed data.
Pratik, Aakash	Sampling SMOTE	Handling the skewed classes issue using SMOTE.
Richa, Amruta	DeepWalk	Implementing DeepWalk algorithm
Kshitij, Vrushabh	Clustering	To extract patterns and apply unsupervised techniques

Table 2. Group Members' Contribution

9. CONCLUSION

We applied different Machine Learning techniques and checked the results based on precision, recall, and F1 score and were able to detect almost all the fraudulent transactions. Various Sampling Techniques such as SMOTE and Self-Paced Ensemble were applied over the existing dataset features and over the graph generated features and the performance was evaluated. Moreover, Deep Walk was also applied with and without Sampling techniques and the results were assessed. The clustering results were visualized using DBScan and K-means clustering with 15 cluster centers. Overall, there was a significant improvement in performance using Sampling techniques.

REFERENCES

[1] Epasto, Alessandro, and Bryan Perozzi. "Is a single embedding enough? learning node representations that capture multiple social contexts." In The World Wide Web Conference, pp. 394-404. 2019.

[2] Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 701-710. 2014.

[3] Abu-El-Haija, Sami, Bryan Perozzi, Rami Al-Rfou, and Alexander A. Alemi. "Watch your step: Learning node embeddings via graph attention." In Advances in Neural Information Processing Systems, pp. 9180-9190. 2018.

[4] Kaggle.com. (2020). Synthetic data from a financial payment system. [online] Available at: https://www.kaggle.com/ntnu-testimon/banksim1 [Accessed 22 Feb. 2020].

[5] dzone.com. (2020). Finding Needles in a Haystack With Graph Databases and Machine Learning - DZone AI. [online] Available at: https://dzone.com/articles/finding-needles-in-a-haystack-with-graph-databases [Accessed 22 Feb. 2020].

[6] Zhining Liu, Wei Cao, Zhifeng Gao, Jiang Bian, Hechang Chen, Yi Chang, Tie Yan Liu. Self-paced Ensemble for Highly Imbalanced Massive Data Classification. September 2019

[7] Scikit-learn.org. 2020. 2.3. Clustering — Scikit-Learn 0.22.2 Documentation. [online] Available at: https://scikit-learn.org/stable/modules/clustering.html#k-means [Accessed 1 May 2020].

[8] Scikit-learn.org. 2020. 2.3. Clustering — Scikit-Learn 0.22.2 Documentation. [online] Available at: https://scikit-learn.org/stable/modules/clustering.html#optics [Accessed 1 May 2020].

[9] Scikit-learn.org. 2020. 2.3. Clustering — Scikit-Learn 0.22.2 Documentation. [online] Available at: https://scikit-learn.org/stable/modules/clustering.html#birch [Accessed 1 May 2020].

[10] Brownlee, Jason. SMOTE Oversampling for Imbalanced Classification with Python https://machinelearningmastery.com. January 2017