Bitcoin Transactions

Amirali Mrashifar and Vinay Pandya

San Jose State University

May 17, 2022

Contents

Chapter 1

Introduction

1.1 Background

The rise of crypto currencies has led to a great deal of interest and has been used as a means of monetary transactions. However, the many transactions in the crypto currency world are used for illegal activities. More specifically, many criminals are hiding in plain sight as their transactions are illegally conducted. One of the most prominently used cryptocurrencies is Bitcoin. In this project, we have analyzed the bitcoin network. The bitcoin transactions can be labeled into two different categories. We consider illegal transactions as illicit and legal transactions as licit. In this project we have used different machine learning models to predict features of different transactions in the bitcoin network. The goal of this project is to identify ways to fight anti Money laundering schemes in the cryptocurrency world

1.2 Data set

In this project we have used the Elliptic Data Set. This dataset maps Bitcoin transactions to different entities. Furthermore, the Bitcoin Elliptic data set is the largest labeled transaction dataset publicly available in the cryptocurrency world (Weber and Domeniconi, et al.). In this data set there are 203,769 nodes which are Bitcoin transactions. And 234,355 directed edges representing payment flows. In addition, there are also 166 node features in this data set. However, some of these node features are not known to the public.

General Structure of the graph iS shown in Figure 1.1 where the green nodes represent the neighbours of the highlighted node,blue represent "licit" nodes and the red represents the "illicit" nodes. Some statistics about the

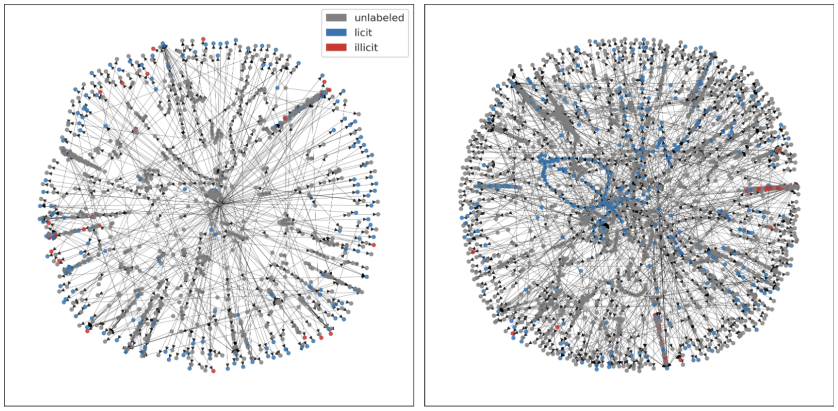

Figure 1.1: Nodes and their representation with their classes

dataset include

- Only 2% of the nodes (4,545) are labeled as "illicit" whereas 21% (42,019) are labelled ""Licit"
- Most of the nodes are not labelled.
- The diameter of the dataset is 1248 whereas the nodes which have zero degrees are the Coinbase transactions which are given to the miners to mine.
- There are 166 features associated with each node on the graph.
- 94 present local information about the transaction, timestamp, output volume and aggregated features such as average BTC spend by the inputs/outputs and average number of incoming (outgoing) transactions associated with the inputs/outputs.
- Average degree of the dataset is $1.128e^{-5}$
- Number of Connected components in the graph is 49
- Number of Connected components are 291(Figure 1.2)

Since the graph is temporal "illicit" nodes do not appear in the beginning of the graph but start to appear as the graph grows and the time passes. As shown in Figure 1.3 most of the "illicit" nodes start to pop up after timestep 10. The maximum number of "illicit" nodes are added at timestep 29 and timestep 42. On looking at the clustering of the "illicit" nodes it seems that they have high betweeness centrality (Table 1.1). Looking at the general

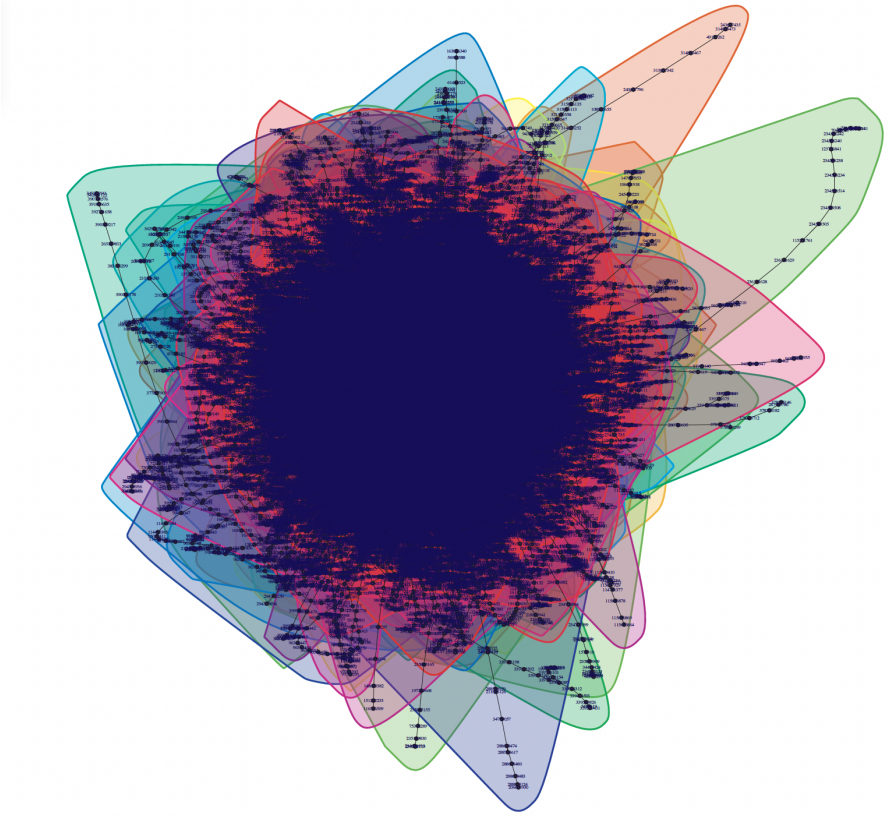

Figure 1.2: Communities for the dataset

Node	Betweeness Centrality
230425980	$2.408385e^{-11}$
5530458	$2.408385e^{-11}$
232022460	$2.408385e^{-11}$

Table 1.1: Nodes with Highest betweenness centralities

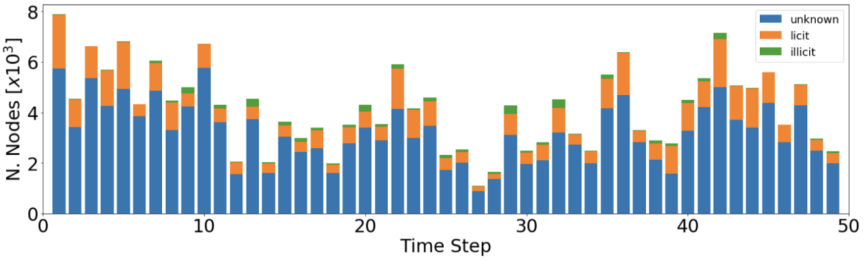

Figure 1.3: Number of Nodes per timestep

structure of the graph does not reveal many things and even the community detection do not convey much information. The General structure of the graph is sown in Figure 1.1 the "illicit" nodes are dispersed between many licit nodes to create obfuscation which suggests that these nodes are involved in some money laundering activities. While the nodes (red nodes) which appear together suggest some sort of terrorist activity. Even with the community detection and other statistics it is difficult to separate the nodes into "illicit" and "licit" as they do not consider the node features nor exploit the structure of the graph.

Chapter 2

Algorithms

The algorithms used in this project are as follows

1. Graph Convolutional Networks
2. Graph Attention Convolution
3. Graph Convolution with Jumping Knowledge
4. GraphSAGE with Jumping Knowledge

2.1 Graph Convolutional Networks

Convolutional neural networks work well with the structured data like arrays and vectors, however they do not incorporate the strutural properties of the graph. Unlike grid like structures like arrays and images, graphs are not euclidean. They are contained in their own subspace and are permutation invariant. In order to adapt the convolutional networks to handle non-euclidean data structures like graph their basic function needs to be modified. Authors in [3] introduced the concept of Graph Convolutions in order to exploit the structural properties of the graph [3].

$$H^{(l+1)} = \sigma(\tilde{D}^{\frac{-1}{2}}\tilde{A}\tilde{D}^{\frac{-1}{2}}H^{(l)}W^{(l)}) \tag{2.1}$$

The equation 2.1 can be broken down as follows

- $\tilde{A}$ is the Adjacency matrix
- $\tilde{D}$ contains degree representations
- $D^{\frac{-1}{2}}A^{\frac{-1}{2}}D^{\frac{-1}{2}}$ is the spectral decomposition which accomodates varying degree distributions.

2.2 Graph Attention Convolution

Attention mechanism is widely popular in machine translation as it helps in focusing on words which are important during translation [1]. In Graph Attenttion convolution we apply the attention mechanism where different neigbhours acquire different weights depending on their significance in prediction. The general operation of attention mechanism is given as follows

$$\alpha_{i,j} = \frac{\exp(LeakyReLU(a^T \|\Theta x_i\| \|\Theta x_j)}{\sum_{k \in N_{(i)\cup(j)}} \exp(LeakyReLU(a^T [\Theta x_i \| \| \Theta x_k]))} \tag{2.2}$$

$$H^{l+1} = \sum_{u \in N} (H_u, H_l + 1) H(l) \tag{2.3}$$

Equation 2.2 calculates attention weights for the k hop neighbourhoods which helps in looking at nodes which provide more information while ignoring others [4].

2.3 Jumping knowledge with Graph Convolution

Instead of applying linear softmax to only the lstGraph Convolutional layer, authors in [5] use an aggregator function to combine the convolutional layers and apply the final feed forward model. The aggregator can be any function

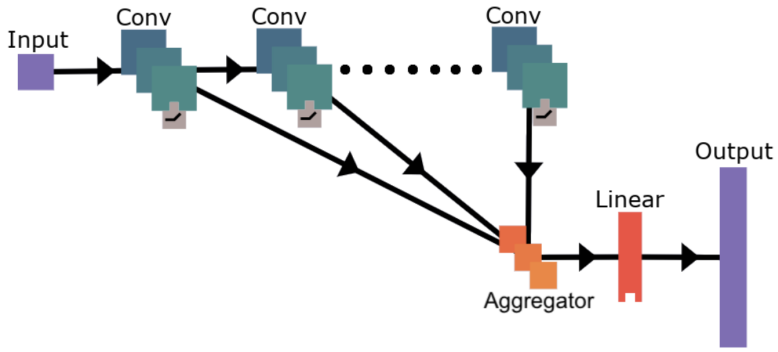

Figure 2.1: JumpingKnowledge architecture [5]

like "add","concat","max-pooling" and "LSTM". Max-pooling selects the most important hidden layer for all the features but is highly unstable.

2.4 GraphSAGE

Most of the methods are transductive *i.e* complete graph needs to be present during training. However since the elliptic dataset is temporal [3] our algorithm should be able to incorporate chenges without the need to retrain. Authors in [2] introduce a mechanism which can incorporate both the temporal structure as well as the node properties of the graph In line 4(Figure 2.2)

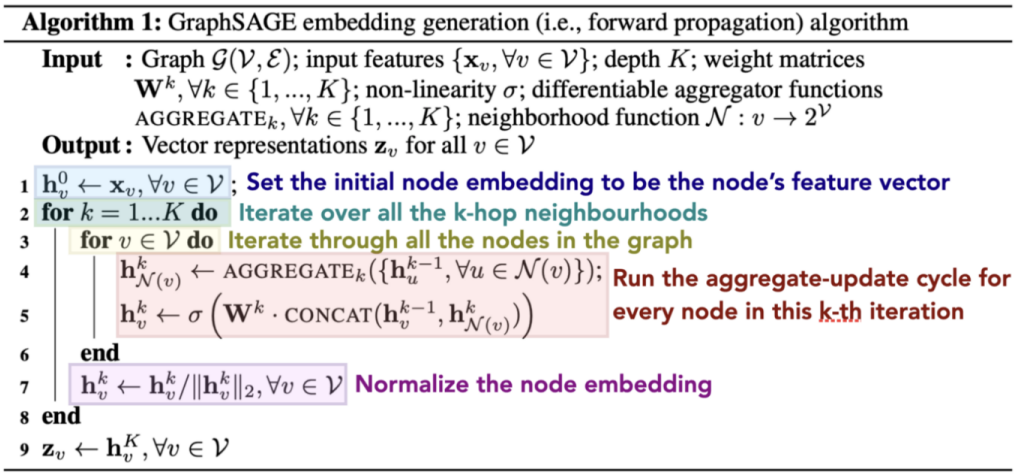

Figure 2.2: GraphSAGE algorithm

a variety of aggregator functions including using max-pool, mean aggregation and even LSTM aggregation. The LSTM aggregation method required the nodes to be shuffled every k-iteration so as to not temporally favour any one node when computing the aggregation. Line 5 is the simple concatenation operation.There is a seperate weight matrix for each operation. Finally the node embeddings are normalized in line 7

2.5 GNNExplainer

GNNExplainer is an explanation tool which provides interpretable explanations for Deep Graph algorithms. The main idea is to generate a minimal subgraph which maximizes mutual information for a prediction [6] The concept is taken from shapely values where we create combinations of nodes and edges and determine their importance. If the absence of a particular node or edge do not affect the prediction then they are discarded from the optimal subgraph.

Chapter 3

Experiments

3.1 GCNConvolution experiments

We use a 2 layer GCNConvolution (Section 2.1) network and apply that on the complete graph the results are as follows

- **Accuracy** 93.5%
- predicts everything as "illicit"
- **f1 Score** 0.11%

looking with GNNExplainer we get Figure 3.2
The model sees feature 118 as the most important(Figure 3.3)

3.2 GATConvolution experiments

We use a 2 layer GATConvolution (Section 2.2) network and apply that on the complete graph the results are as follows

- **Accuracy** 93.5%
- better than GCNConv
- **f1 Score** 0.46%

looking with GNNExplainer we get Figure 3.5
The model sees feature 140 as the most important(Figure 3.6)

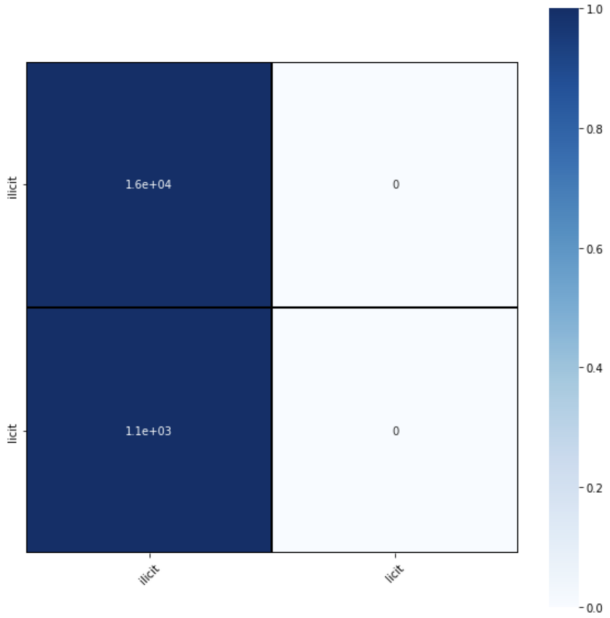

Figure 3.1: Confusion matrix for GCN

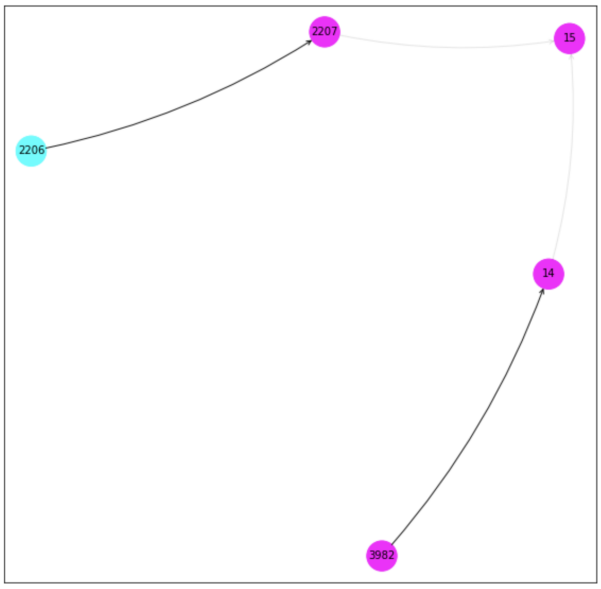

Figure 3.2: SubGraph Generated for GCN model

3.3 JumpingKnowledge experiments

Jumping Knowledge network with concatenation aggregator (Section 2.3) and apply that on the complete graph the results are as follows

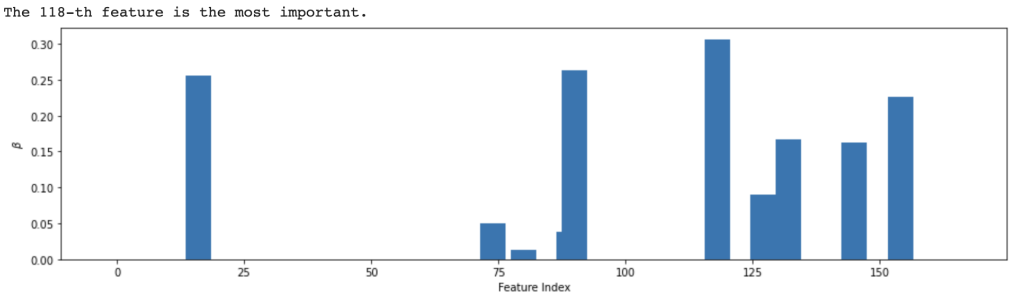

Figure 3.3: Feature Importance GCN

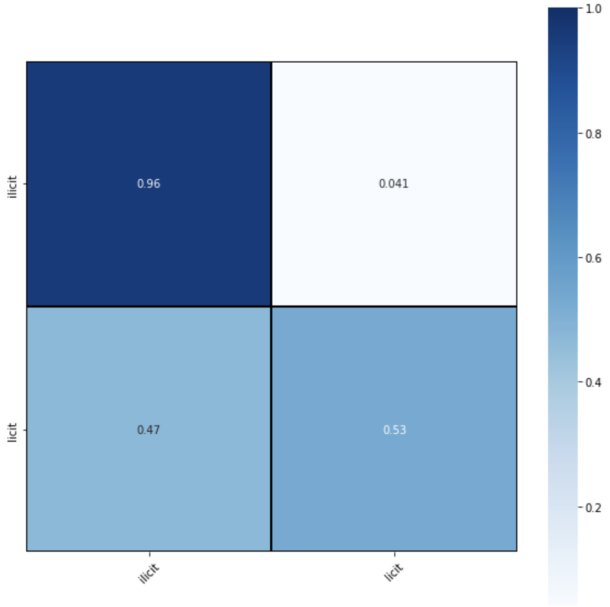

Figure 3.4: Confusion matrix for GATConv

- **Accuracy** 95.35%
- Better than attention network
- **f1-SCORE** 0.77

looking with GNNExplainer we get Figure 3.2
The model sees feature 144 as the most important(Figure 3.9)

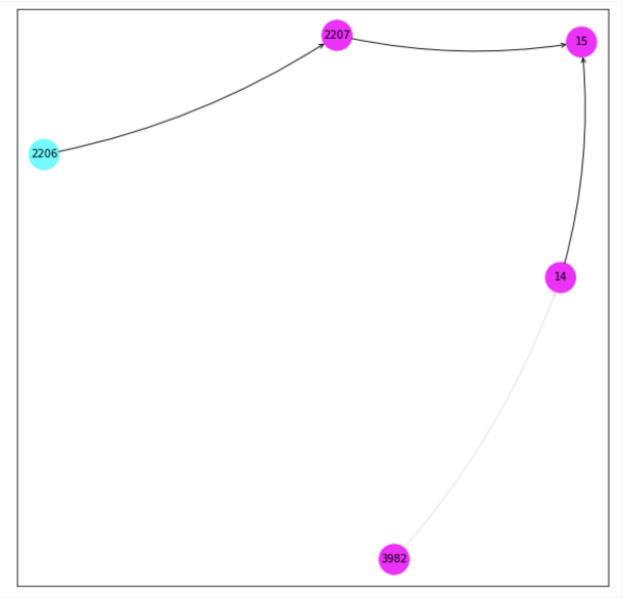

Figure 3.5: SubGraph Generated for GATConv model

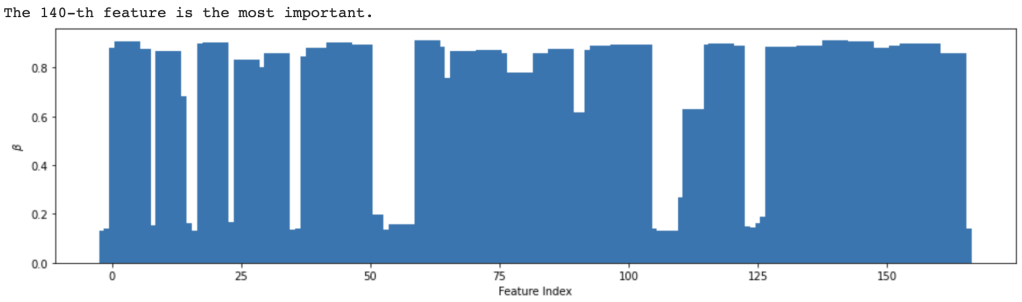

Figure 3.6: Feature Importance GATConv

3.4 GraphSAGE experiments

We use a 3 layer GraphSAGE (Section 2.4) network with jumping knowledge aggregator and apply that on the complete graph the results are as follows

- **Accuracy** 96%
- Predicts reasonably well
- **F1-score** 0.79

looking with GNNExplainer we get Figure 3.11

The model sees feature 157 as the most important(Figure 3.12)

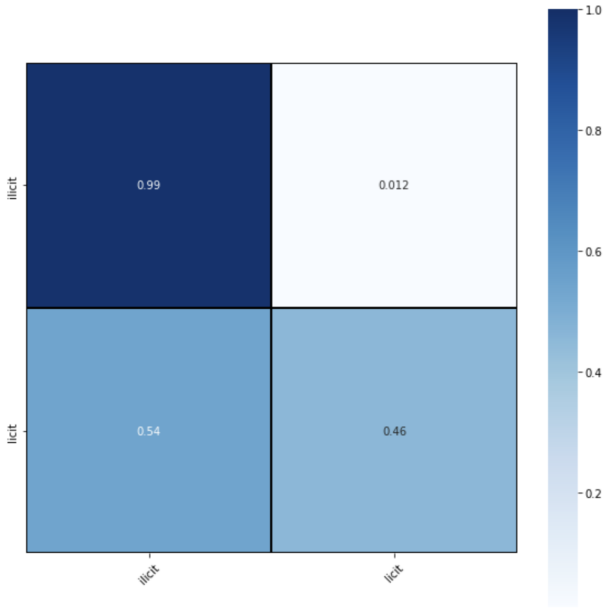

Figure 3.7: Confusion matrix for JumpingKnowledge GCN

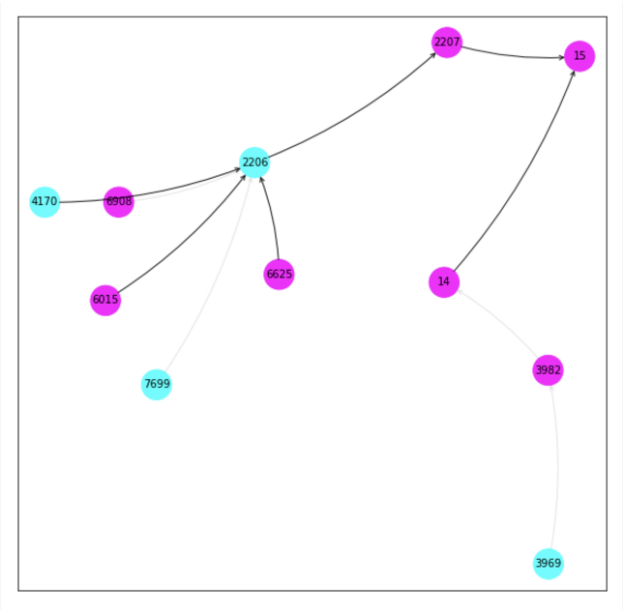

Figure 3.8: SubGraph Generated for Jumping Knowledge model

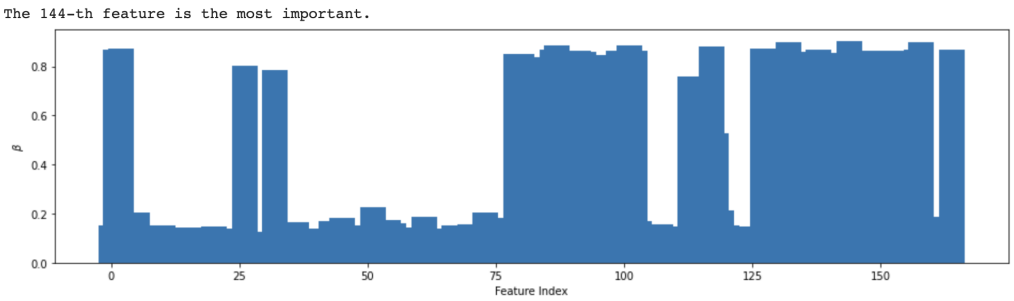

Figure 3.9: Feature Importance Jumping Knowledge

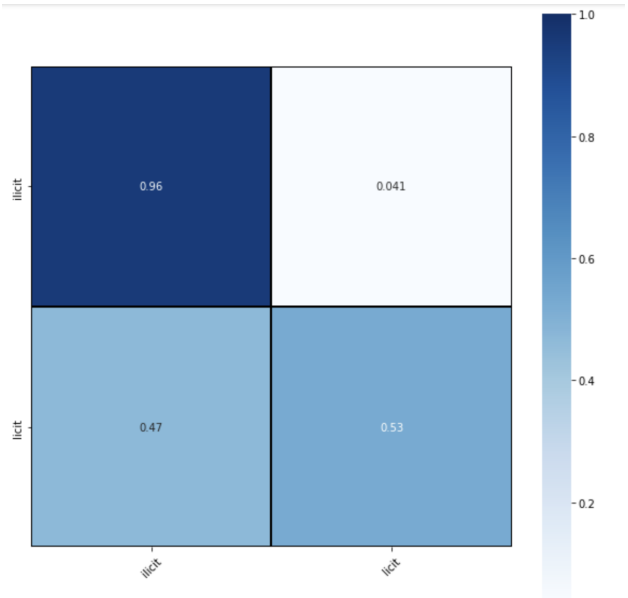

Figure 3.10: Confusion matrix for GraphSAGE

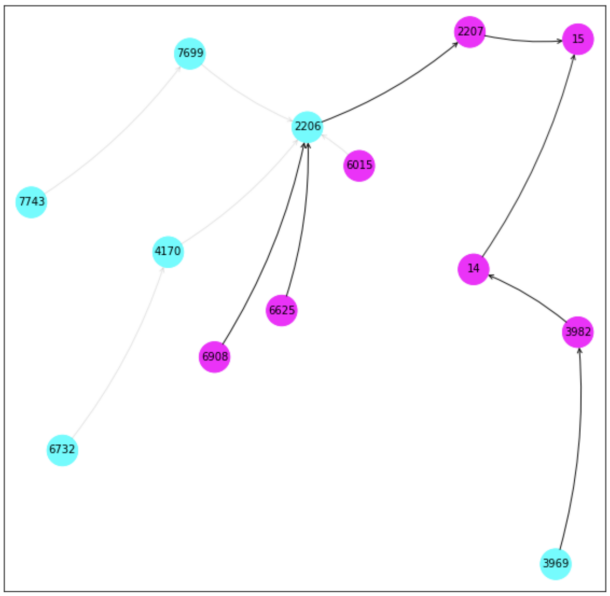

Figure 3.11: SubGraph Generated for GraphSAGE model

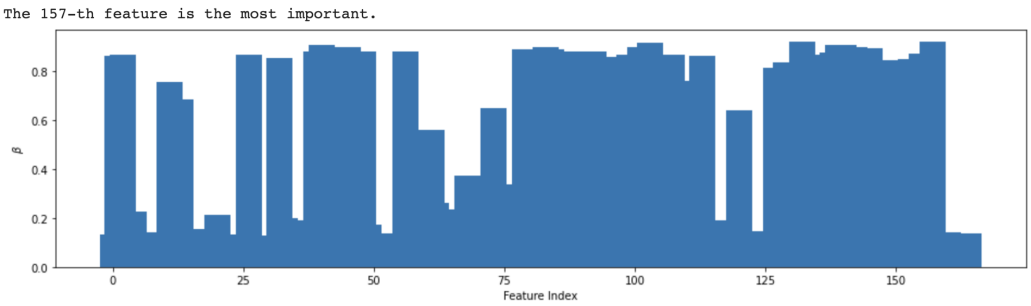

Figure 3.12: Feature Importance GraphSAGE

Chapter 4

Conclusion

Throughout this experiment we had the opportunity to work with the Elliptic Data Set, the largest publicly labelled cryptocurrency data set available. Different neural networks were used to leverage the graph and the structural properties of the graph. However, traditional structures of neural networks had to be modified in our experiment. We used Convolutional Neural Networks (CNN) and Graph Convolutional Networks (GCN) to leverage the graph, however, these networks failed to get us an acceptable F1 score. Later we utilized GNNExplainer and GATConv to leverage the graph which we saw more success, however, the results were not acceptable due to a low F1-score. After, Jumping Knowledge was used, the experiment had its highest F1-score. Later, Jumping Knowledge was combined with GraphSAGE. The results of this combination were the highest recorded in the experiment with an Accuracy of 96% and an F1 score of 0.79. All things considered, GCN alone does not capture the information of the graph, GCN combined with Jumping Knowledge improves the accuracy, and GraphSAGE model works really well with temporal graphs. Hence, it exploits the internal structure of the graph well.

Bibliography

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

[2] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

[3] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2016. DOI: 10.48550/ARXIV.1609.02907. URL: https://arxiv.org/abs/1609.02907.

[4] Petar Veličković et al. *Graph Attention Networks*. 2017. DOI: 10.48550/ARXIV.1710.10903. URL: https://arxiv.org/abs/1710.10903.

[5] Keyulu Xu et al. *Representation Learning on Graphs with Jumping Knowledge Networks*. 2018. DOI: 10.48550/ARXIV.1806.03536. URL: https://arxiv.org/abs/1806.03536.

[6] Rex Ying et al. *GNNExplainer: Generating Explanations for Graph Neural Networks*. 2019. DOI: 10.48550/ARXIV.1903.03894. URL: https://arxiv.org/abs/1903.03894.