Detecting Anomalous Business Ownership Structures with Graph Neural Networks

Final Project

Dominic Thorn

Abstract – Illegitimate and erroneous declaration of business ownership is a problem for governments and financial institutions worldwide, resulting in tax evasion, fraud, and the general facilitation of criminal activity. We compare the performance of Graph Neural Networks and a gradient-boosted tree ensemble in identifying simulated anomalous owners in a public dataset of UK businesses and shareholders. Both GraphSAGE (Hamilton et al., 2018) and kGNN (Morris et al., 2021) models outperform the gradient-boosted tree ensemble, with kGNN outperforming GraphSAGE on this task. We propose a set of recommendations for practitioners interested in applying GNNs to fraud detection tasks and discuss avenues for further work.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Background

In October 2021, The International Consortium of Investigative Journalists (ICIJ) revealed the findings of their Pandora Papers investigation. Through examination of approximately 12 million confidential business records, the ICIJ found evidence implicating thousands of individuals and businesses in efforts to conceal the ownership of companies and assets around the world (ICIJ, 2021). The intentions behind this secrecy varied from legitimate privacy concerns to criminal activities, including money laundering, tax evasion, and fraud (European Union Agency for Law Enforcement Cooperation, 2021).

To put these numbers in perspective, a 2019 study by the European Commission estimated that a total USD 7.8 trillion was held offshore as of 2016. The share of this attributed to the European Union (EU) was USD 1.6 trillion, which corresponds to an estimated tax revenue loss to the EU of EUR 46 billion (European Commission. Directorate General for Taxation and Customs Union., 2019).

The ease with which information can be concealed or not declared makes it difficult to identify the beneficiaries of a company. Further complications are introduced by the interconnected nature of businesses and individuals, as well as the ingenuity of criminals in masking illicit activities. These difficulties place significant strain on the resources of law enforcement agencies and financial institutions (Steven M., 2019).

In April 2016, the United Kingdom made it mandatory for businesses to keep a register of People with Significant Control. This includes people owning over 25% of the company's shares (*Keeping Your People with Significant Control (PSC) Register*, 2016). Ownership data is curated and processed by the Open Ownership organisation for public scrutiny and research (*Open Ownership*, 2022). It is the data provided by Open Ownership that forms the basis of this study. Details of suspicious or illegitimate business owners are not available because of the sensitive nature of such records. As part of our experimental methods, we propose a method for the simulation of anomalous ownership structures.

To model the complex network of global business ownership, it is necessary to represent companies, people, and their relationships in a graph structure. With data in this format, it is possible to consider the features of an entity's local neighbourhood, as well as the entity's characteristics, when investigating potential fraud. Anomaly detection algorithms that operate on graph structures remain at the frontier of machine learning research.

To the best of the author's knowledge, there is no published research studying the effectiveness of graph anomaly detection techniques on business ownership networks. The following proposal is a study of state-of-the-art anomaly detection techniques as applied to business ownership graphs.

1.2 Aims, Objectives, and Research Questions

1.2.1 Aims

The primary aim of this project is to develop an effective approach for detecting anomalous entities in a business ownership network. Second, the project will offer a comparison of Graph Neural Network (GNN) models to traditional anomaly detection approaches on a business ownership graph. We contribute a dataset describing a real business ownership network that is suitable for graph learning.

1.2.2 Objectives

- Compose a business ownership graph from open data sources.
- Perform the anomaly detection task with traditional machine learning methods.
- Train and evaluate GNN models for anomaly detection.
- Compare approaches in terms of effectiveness.

1.2.3 Research Questions

The questions driving the research are:

- Q1: What is the most effective strategy for detecting anomalous entities in business ownership networks?
- Q2: How do GNN models compare to traditional approaches in terms of classification performance?
- Q3: What are the challenges that arise in building and training a GNN model and what recommendations can be made for practitioners?

1.2.4 Ethical Considerations

All data used in this project has been made available for public scrutiny and academic research.

In the interests of personal privacy, any sensitive personal information is transformed and/or removed from the dataset before analysis and modelling. We remove address information and the names of individual persons and companies.

Further, the study refrains from speculating on any business structure or entity, and any speculation on the legitimacy or legality of a particular arrangement is deemed out of the project scope.

To facilitate academic openness and collaboration, the instructions and code required to produce the datasets and experimental results will be made available following the review process.

1.3 Prior and Related Work

Few published studies focus on the detection of anomalous business ownership structures. We review the most relevant literature below.

1.3.1 Detecting Financial Fraud

Luna et al. (2018) describes a procedure for identifying suspected shell company accounts using distance and density-based anomaly detection techniques. The authors were successful in detecting shell companies by observing differences in transactional behaviour. A notable caveat is that the data was simulated for the study, which leaves questions about its applicability to real-world scenarios.

Recent work by Dumitrescu et al. (2022) demonstrates how local neighbourhood features and statistical scores can be used in Anti-Money Laundering (AML) models. Relevant features included unsupervised anomalous node detection techniques (Akoglu et al., 2010) and local neighbourhood connectivity features (Molloy et al., 2016) calculated on *reduced egonets*. A strength of the study is that it was conducted on genuine labelled transactional data with positive results. The authors did not implement any GNN or other deep learning approaches for comparison.

Fronzetti Colladon & Remondi (2017) explore a range of social network analysis techniques for identifying money laundering using data kept by an Italian factoring company. The authors found that constructing numerous networks from different projections of the graph entities improved the power of individual risk metrics. Degree centrality was determined to be a significant predictor of risk in all cases, while in certain scenarios network constraint proved to be informative. We note these results are for clients of a single business and that additional work is required to demonstrate wider validity.

1.3.2 Anomaly Detection for Graphs

Akoglu et al. (2014) highlights four main reasons for the suitability of graph structures in anomaly detection:

Inter-dependent nature of the data – "Data objects are often related to each other and share dependencies." This can be observed in business ownership data through the relationships that connect individuals and companies in legal hierarchies and communities.

Powerful representation – Graphs offer a powerful way of representing inter-dependencies and long-range correlations between related entities. By using different node and edge types, as well as additional attributes, it is possible to represent rich datasets. These properties are valuable in capturing the different entity types present in a business ownership graph. A business, for example, will have attributes not shared by individuals, such as an industry classification code.

Relational nature of problem domains – "The nature of anomalies could exhibit themselves as relational". In the context of detecting anomalous business ownership, individuals and businesses may be anomalous through their unusual relationships with other entities.

Robust machinery – "Graphs serve as more adversarially robust tools." It is suggested that graph-based systems are well suited for fraud detection, as bad actors will find it difficult to alter or fake their position in the global structure.

A thorough description of graph anomaly detection tasks and approaches is offered by X. Ma et al. (2021). Their taxonomy categorises tasks based on the graph component being targeted: nodes, edges, sub-graphs, or full graphs. The authors state their belief that "because the copious types of graph anomalies cannot be directly represented in Euclidean feature space, it is not feasible to directly apply traditional anomaly detection techniques to graph anomaly detection".

1.3.3 Graph Neural Networks

Kipf & Welling (2017) proposes a scalable GNN architecture for classifying nodes in a partially labelled dataset. Early attempts to apply deep learning to graph structures utilised RNN architectures, which prove difficult to scale (Gori et al., 2005; Li et al., 2017; Scarselli et al., 2009). Kipf et al. extend prior work on spectral GNNs (Bruna et al., 2014; Defferrard et al., 2017) to produce a flexible model that scales in linear time with respect to the number of graph edges.

Ding et al. (2019) combines a GNN architecture with an auto-encoder in a method that identifies anomalous nodes by reconstruction error. The proposed method, DOMINANT, uses a GNN to generate node embeddings and separately reconstructs both the graph topology and the node attributes. This strategy is further developed and applied to multi-view data by combining multiple graph encoders (Peng et al., 2022).

An alternative method is offered by Li et al. (2019), in which a spectral convolution and deconvolution framework are used to identify anomalous nodes in conjunction with a density estimation model. The approach continues to indicate the importance of combining multiple perspectives of the network data, with the innovation being the use of a Gaussian Mixture Model to combine representations in a single view.

1.3.4 Graph Attention Networks

Veličković et al. (2018) demonstrate the use of self-attention layers to address shortcomings in the representations captured by GNN architectures. However, a comparison of Relational Graph Attention (GAT) models to GNNs showed that relative performance was task-dependent and that current GAT

models could not be shown to consistently outperform GNNs on benchmark exercises (Busbridge et al., 2019).

In an application of graph attention-based models to financial fraud detection, D. Wang et al. (2019) shows that their SemiGNN model outperforms established approaches when predicting the risk of default and in attribute prediction. Baseline methods used for comparison included XGBoost (Chen & Guestrin, 2016), GNN, GAT, and LINE (Tang et al., 2015).

1.3.5 Heterogeneous Graph Neural Networks

The studies reviewed above have focused on the detection of anomalous nodes in homogeneous graphs. However, the structure of a business ownership graph is necessarily heterogeneous if we wish to capture the specific attributes of people and companies. The field of heterogeneous graph learning is less developed than its homogeneous counterpart, and so we determine that the most promising approach is to adapt existing methods to the heterogeneous setting.

Fortunately, implementations of many state-of-the-art GNN architectures are available in the PyTorch Geometric library (Fey & Lenssen, 2019), as well as methods to adapt homogeneous models to heterogeneous graph learning tasks.

2 Dataset

Training a supervised classification model requires access to a dataset of labelled examples. Given the typical infrequency of fraudulent events compared to legitimate cases, it is common for fraud classification projects to require large amounts of data to provide a modest number of fraudulent examples from which the model can learn. Considering the sensitive nature of fraud investigations, it is not surprising to find that no such data is available in the public domain.

In the following section, we detail the public data sources used in this study and the steps necessary for processing them. We further propose a method for simulating anomalous business ownership structures using this publicly available data.

2.1 External Sources

2.1.1 People with Significant Control Register

Since 2016, it has been a requirement for all businesses in the United Kingdom to declare People of Significant Control (PSC) (*Keeping Your People with Significant Control (PSC) Register*, 2016). This

includes all shareholders with ownership greater than 25%, any persons with over 25% of voting rights, and any person with the right to appoint or remove the majority of the board of directors (*People with Significant Control (PSCs)*, 2022). The register is available as a daily snapshot, available to download from the Companies House webpage (*Companies House*, 2022).

Included in the register are the name, address, and identification number of each company for which a declaration has been received. Also listed are the name, address, country of origin, date of birth, and nature of control for each person listed as a PSC.

Rather than consume this data directly, we obtain this information from a data feed curated by the Open Ownership organisation.

2.1.2 Open Ownership Register

The Open Ownership organisation maintains a database of over 16 million beneficial ownership records, including those collated in the UK PSC register and from sources made available by other countries (*Open Ownership*, 2022). This data is provided in a standardised format that is conducive to machine processing and graph generation. Additional data quality processing is undertaken by Open Ownership, such as the merging of duplicated records for persons that have more than one PSC declaration (*Beneficial Ownership Data Standard*, 2022).

The Open Ownership Register is used as a canonical source of company, person, and ownership relationships to generate our UK business ownership graph.

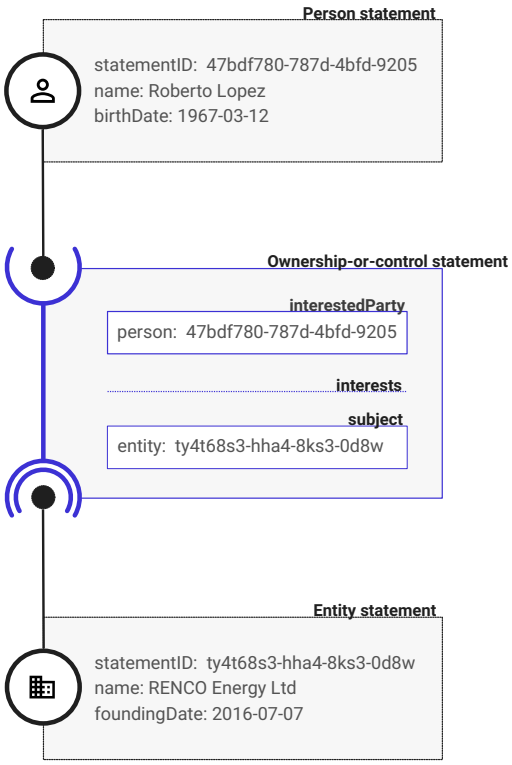

Figure 1: Open Ownership data schema (*Beneficial Ownership Data Standard*, 2022)

2.1.3 The Free Company Data Product

The Free Company Data Product is a monthly snapshot of data for all live companies on the UK public register. Taken from the Companies House data products website, this data includes:

- basic information including company type and registered office address
- the nature of business or standard industrial classification (SIC)
- company status, such as “live” or “dissolved”
- date of last accounts or confirmation statement filed
- date of next accounts or confirmation statement due
- previous company names

(*Companies House Data Products*, n.d.)

This data serves as an additional source of node features for company entities in the generated dataset.

2.2 Initial Data Preparation

The Open Ownership data file comprises over 20 million records stored as nested JSON objects. This includes data for businesses and relevant persons registered outside the UK. The Apache Spark framework (*Apache Spark*, 2022) is used for bulk data preparation because of its parallel and out-of-core processing capabilities, as well as its support for nested data structures.

As this study covers only UK registered companies and their shareholders, records for entities that do not have a shareholding interest in a UK company are discarded. Non-UK companies that are registered as a shareholder of a UK company are also discarded, as we cannot obtain information for these entities via Companies House. Computational resource constraints also prevent the handling of a larger dataset. To further limit dataset size, and in the interests of only considering accurate and up-to-date information, we also filter out companies that are listed as dissolved.

While the initial nested data schema is desirable for clean representation and compact storage, we require a flat relational table structure for analytical and model training purposes. Relevant data items are extracted into a flat table for each entity type. This results in three tables of interim output: company information, person information, and statements of control that link entities to their ownership interests.

The Companies House data is joined to the company entity table via the UK company registration number.

2.3 Graph Generation

2.3.1 Nodes and Edges

The company, person, and ownership relationship tables prepared in the previous steps are used to create a graph data structure. Companies and persons are represented as nodes in the graph, and the ownership relationships are represented as directed edges (from owner to owned entity) between them. The resulting graph is attributed with node features and edge weights. Since the graph comprises two types of nodes with distinct feature sets, it can be described as an attributed heterogeneous graph (X. Ma et al., 2021).

2.3.2 Connected Components

Connected components are labelled in the graph to facilitate additional filtering and to allow for parallel computation of topological features.

Two nodes, u and v, are considered connected if a path exists between them in the graph G. If no path exists between u and v, then they are said to be disconnected. Not all entities in the ownership graph are

connected by any path, and so the ownership graph G can be viewed as a set of disjoint sets of nodes, or components. The graph $G = (V, E)$ is described by its components thus: $G[V_1], G[V_2], ...G[V_\omega]$. If the number of nodes in a graph is denoted by $|V|$, the number of nodes in a component ω can be described by $|V_\omega|$(Bondy & Murty, 1982, p. 13).

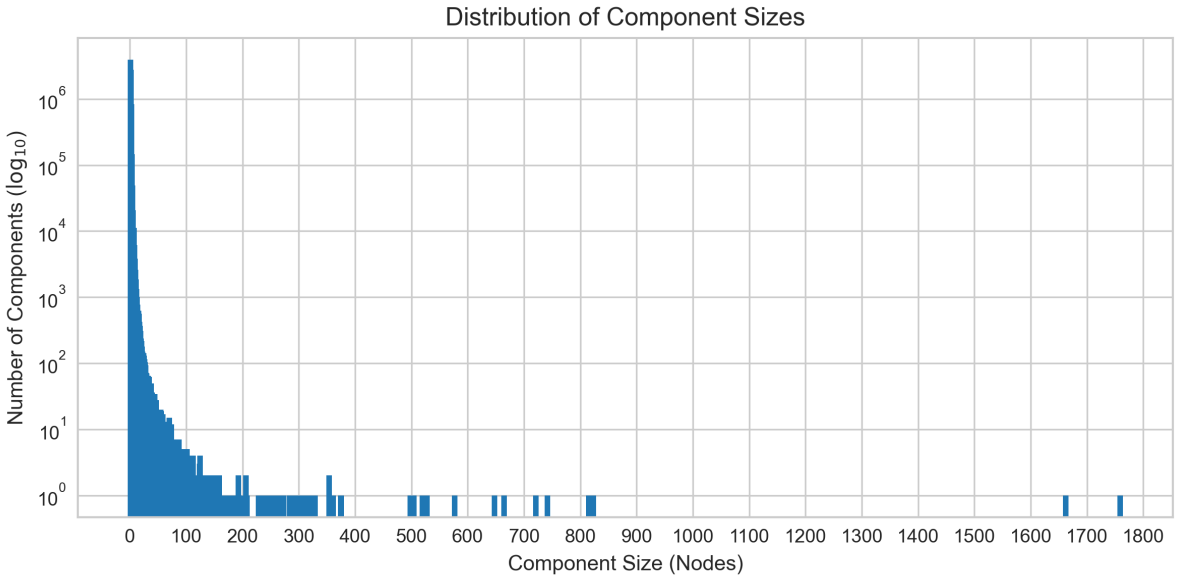

Figure 2: Initial distribution of component sizes. A single super-component comprising 27,008 nodes is excluded from the plot.

The vast majority of businesses in the dataset have only one or two declared PSC shareholders. The initial distribution of component sizes is shown in figure 2. These small sub-graphs are not of interest in this study and are excluded, along with any component that contains fewer than ten nodes or with a ratio of less than 1 in 10 natural persons to companies.

The rationale for excluding these components is threefold: first, the small number of nodes and relationships presents little information for a model to learn from; second, these small networks are less likely to be of interest in real-world fraud detection, as illegitimate ownership is often concealed behind multiple layers of ownership; and from a practical standpoint, the volume of data is brought down to a manageable size. Networks with fewer than 1 in 10 natural persons to companies are excluded, as a focus of this study is on the performance of anomaly detection methods on heterogeneous graph data, as opposed to homogeneous networks.

We address an observed data quality issue in which multiple nodes in the same component may share the same name. These nodes are merged into a single node, with the resulting node taking on all the ownership relationships of the merged nodes.

2.4 Structural Anomaly Simulation

To simulate structural anomalies, 10% of the nodes are selected at random and marked as anomalous ($y = 1$) and the remaining 90% are marked as normal ($y = 0$). A single outgoing edge is chosen from each anomalous node, $\epsilon_{norm} = \epsilon(u_i, v_i)$, and its source ID is replaced with that of another node marked as anomalous, u_j. This produces the new anomalous edge $\epsilon_{anom} = \epsilon(u_j, v_i)$ and eliminates the original edge ϵ_{norm}. The result is an exchange of ownership interests between anomalous nodes while preserving the overall structure of the graph. The procedure is illustrated in figure 3. This process is repeated until all anomalous nodes have one edge replaced with a different edge so that for all anomalous nodes, $\epsilon_{anom} \neq \epsilon_{norm}$. The outgoing edges of normal nodes are not altered, though their incoming edges may be affected by the anomaly simulation process.

The final step of the simulation is to discard any small components ($n < 9$) that have been created as a result of the anomaly simulation process. This is done to ensure that anomalous nodes belong to components with a similar size distribution to the original graph, as demonstrated in figures 8 and 9. The final proportion of anomalous nodes in the simulated graph is 7.3%.

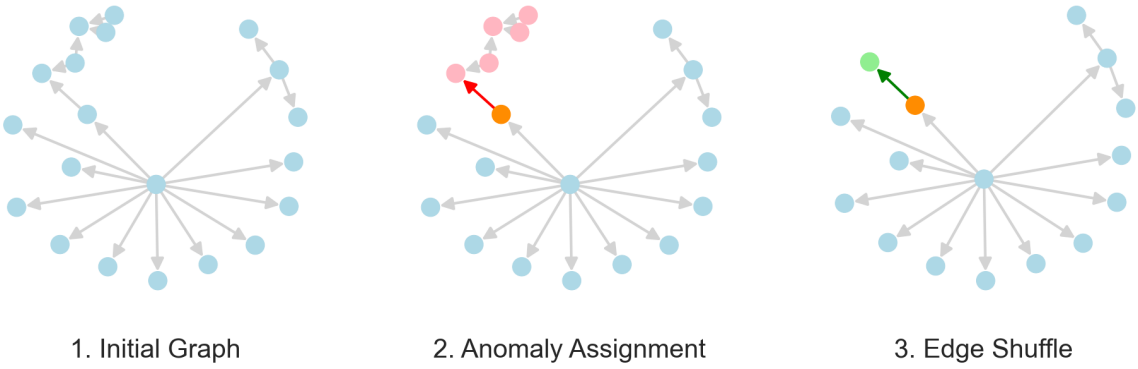

Figure 3: Process for simulating structural anomalies. The initial graph is shown in (1). A node is selected for anomalisation (orange circle) in (2). An outgoing edge (red arrow) is swapped for that of another anomalised node (green arrow), resulting in the exchange of pink nodes for green (3).

The anomaly simulation process introduces unusual ownership relationships into the graph. The true frequency of these occurrences is unknown but is expected to be far lower than 7.3% of nodes. This anomaly rate is chosen to ensure that model training is not impossible because of extreme class imbalance, and the same effect can be achieved by over-sampling the minority class (Chawla et al., 2002) or under-sampling the majority class (Fernández, 2018, p. 82).

When labelling the unaltered nodes as normal, it is assumed that the data provided by Open Ownership is accurate and that if any anomalies are present, they will have a negligible impact on experimental

results. Anomalies in the initial dataset will be seen by all candidate models, and so should not bias the results.

2.5 Node Features

To train a baseline model for comparison, a set of node-level features is generated to represent each node in a tabular format (Leskovec, 2020). The following topological features are extracted for each node:

- In-degree: The number of incoming edges to the node.
- Out-degree: The number of outgoing edges from the node.
- Closeness Centrality: Inverse of the sum of shortest path distances to all other nodes.
- Clustering Coefficient: Connectedness of neighbouring nodes.
- PageRank: A measure of node importance, based on the importance of neighbouring nodes (Page et al., 1998).

To capture information about the node's position in the graph, aggregate statistics are calculated for the aforementioned topological features for the node's neighbours and added as features. We consider the minimum, maximum, sum, mean, and standard deviation of all the aforementioned features for the node's immediate neighbours. We also include the count of immediate neighbours as a feature. These aggregate features are generated only for the training of the benchmark model.

All numerical features are normalised to the range $[0, 1]$ using the StandardScaler from the Scikit-Learn library (Pedregosa et al., 2011). Categorical features are one-hot encoded using the OneHotEncoder transformer.

2.6 Dataset Properties

We convert the dataset to an undirected graph by adding a reverse edge for each edge in the original graph. This is done to allow GNN message-passing algorithms to work across the graph rather than being able to communicate attributes in only one direction.

The final dataset comprises 124,934 nodes, made up of 94,054 company entities and 30,880 natural persons. There are 131,892 edges, of which 77,971 indicate person-to-company ownership, while the number of company-to-company ownership edges is 53,921.

3 Computational Resources

The computational resources required to run the main experiments in this study are substantial and beyond the capabilities of a typical consumer-grade machine. We use the Google Cloud Platform service to provision a virtual machine with the following specifications:

Table 1: Computational resources used for the experiments.

Resource	Specification
CPU	4 vCPUs
GPU	1 x T4 (16GB)
RAM	15GB
Disk	100GB

This machine is used for training the GNN models and is not required for processing the dataset or training the baseline model. The baseline model and dataset can be produced in a reasonable amount of time on a consumer-grade machine with 16GB of RAM and no GPU.

We recommend making use of spot instances to benefit from the reduced cost of preemptible hardware. The framework presented by this study is designed to be fault tolerant to help address some challenges encountered when conducting our experiments.

Scripts and commands for repeating the experiments in this study are provided in the accompanying code repository, as well as a Dockerfile, for recreating the container image.

4 Methods

4.1 Traditional Machine Learning Approaches to Binary Classification

To assess the relative improvement that can be achieved by using Graph Neural Networks, we compare their performance to a baseline model trained on a set of node-level features.

4.1.1 Gradient Boosted Trees

Gradient boosted tree ensembles achieve robust performance on a wide range of machine learning tasks

(Friedman, 2001). Modern applications are favoured for their strong performance in capturing complex dependencies, native handling of heterogeneous and missing data, and numerical scale invariance. We use the CatBoost library to train our baseline model, as the current state-of-the-art implementation (Prokhorenkova et al., 2019).

4.2 Graph Neural Networks

The PyTorch Geometric library offers a comprehensive set of methods for deep learning on graph data structures (Fey & Lenssen, 2019). We use the implementations provided in this library to train our Graph Neural Network models.

We select two architectures, GraphSAGE (Hamilton et al., 2018) and kGNN (Morris et al., 2021), for our experiment. These models are chosen for their demonstrated performance at node classification tasks and ability to handle heterogeneous graph data with slight modification (detailed below) Morris et al. (2021). A further consideration in this choice of models is the size of our dataset and the available computational resources. Both architectures are lightweight compared to other models, such as Graph Attention Networks (Veličković et al., 2018), and can be trained on a single GPU in a reasonable amount of time.

Since both GNN architectures selected for this study were designed for learning on homogeneous graphs, we use a method provided by PyTorch Geometric for adapting these homogeneous architectures for learning on our heterogeneous dataset. A homogeneous model is adapted for heterogeneous learning by duplicating message-passing functions to operate on each edge type. Node representations are learned for each node type and aggregated using a user-provided function that we choose via neural architecture search. This technique is described by Schlichtkrull et al. (2017) and illustrated in figure 4.

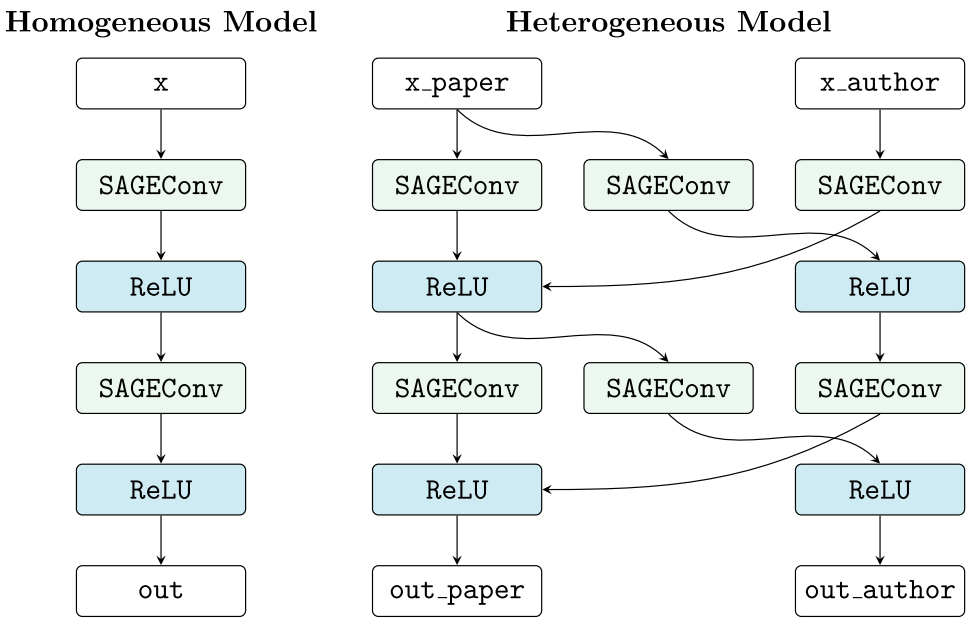

Figure 4: Converting homogeneous GNN architectures for heterogeneous learning (*Heterogeneous Graph Learning — PyTorch Geometric Documentation*, n.d.)

4.2.1 GraphSAGE

The GraphSAGE framework proposed by Hamilton et al. (2018) is a node embedding technique that aggregates node attributes and the attributes of neighbouring nodes to generate node representations. The embedding function is learned through the efficient sampling of each node's local neighbourhood, resulting in a low-dimensional vector embedding that can generalise to unseen data.

The aggregation function can be any symmetric function, such as the mean or sum, or an arbitrarily complex function, such as a neural network, that can operate on an ordered set of node features. For supervised learning tasks, the parameters of the aggregation function are learned via backpropagation.

Edge attributes are not used in the GraphSAGE model and are therefore ignored during training.

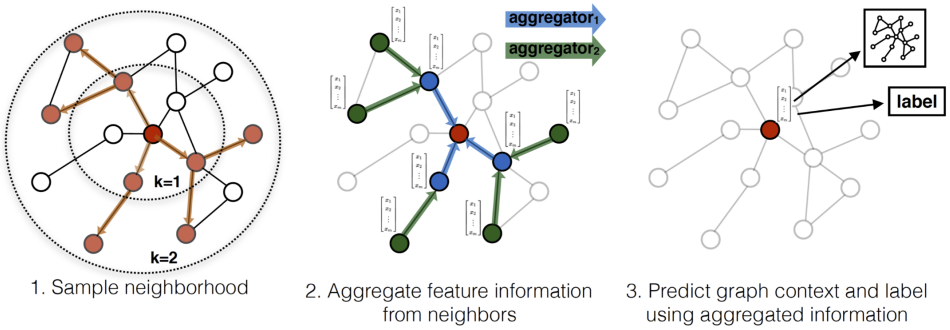

Figure 5: GraphSAGE neighbourhood sampling and aggregation (Hamilton et al., 2018)

4.2.2 Higher-Order GNN (kGNN)

The Higher-Order GNN (or kGNN) proposed by Morris et al. (2021) extends the prototypical GNN described by Kipf & Welling (2017) to higher-order graph representations. These higher-order representations are learned by associating all unordered k-tuples of nodes and learning a representation for each tuple, where k is a hyperparameter of the model. These tuple representations are hierarchically combined in a final dense layer to produce a representation for each node that considers its local neighbourhood and higher-order topological features. This is illustrated in figure 6.

As with the GraphSAGE model, an aggregation architecture is used to learn the node tuple representations. The weights for each layer are also trained via backpropagation for supervised learning tasks. In contrast to GraphSAGE, the kGNN model incorporates edge weights and types into the learning process.

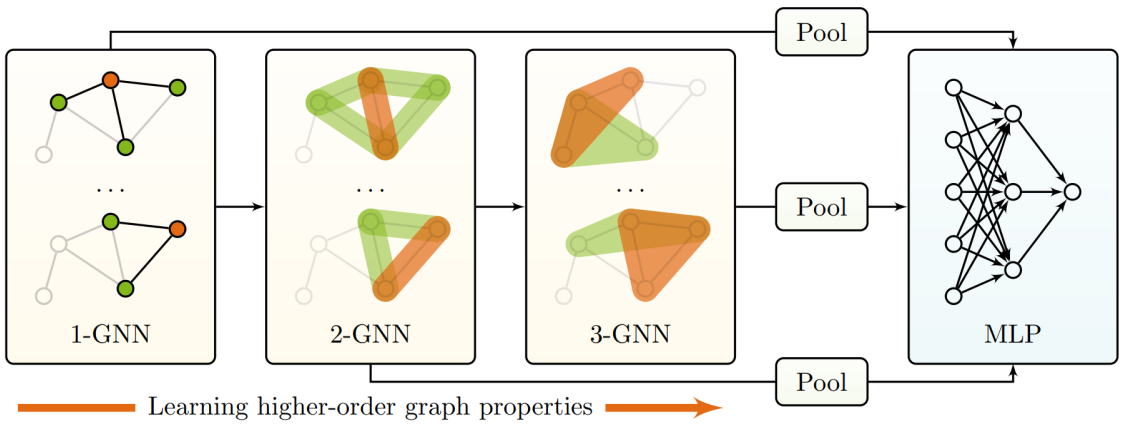

Figure 6: kGNN network architecture (Morris et al., 2021)

4.3 Experimental Setup

4.3.1 Data Splitting

The dataset is split into training, validation, and test sets. The training set is used to train the model, the validation set is used to tune hyperparameters, and the test set is used to evaluate the model's performance. A random 80/10/10 split is used to produce the three data sets, with random selection applied to each component in the graph to ensure that nodes in the same component are not split across multiple sets.

4.3.2 Class Weighting

Class imbalance in the dataset is addressed by assigning a weight to each class during model training. A weight of 10 is applied to anomalous nodes and a weight of 1 for normal nodes. These weights are used as multipliers for the errors of their respective classes, leading to an increased penalty and greater emphasis on anomalous nodes. This is a cost-sensitive approach to class imbalance that does not require the use of over-sampling or under-sampling (He & Garcia, 2009).

4.3.3 Evaluation Metrics

We use two metrics to evaluate model performance: the area under the precision-recall curve (AUC-PR) and the area under the receiver operating characteristic curve (AUC-ROC). These metrics are chosen as they are insensitive to the choice of threshold, and are more robust to class imbalance than threshold-dependent metrics, such as accuracy. (Y. Ma & He, 2013, p. 72)

4.3.3.1 Area under the Receiver Operating Characteristic Curve (AUC-ROC) The Receiver Operating Characteristic Curve (ROC) is a plot of the true positive rate against the false positive rate along a range of threshold settings. A perfect model will achieve a 100% true positive rate with a 0% false positive rate, while a zero-skill classifier will achieve a 50% true positive rate with a 50% false positive rate. The area under the ROC curve (AUC-ROC) provides a way to summarise model performance over the range of threshold values. A perfect model will have an AUC-ROC of 1, while the zero-skill model will have an AUC-ROC of 0.5. (Fawcett, 2006)

4.3.3.2 Area under the Precision-Recall Curve (AUC-PR) The Precision-Recall Curve (PR) is a plot of precision against recall along the range of recall values. A perfect model will achieve 100% precision and 100% recall, while a zero-skill classifier will achieve a rate of precision equal to the proportion of positive cases in the dataset for all recall values. The area under the PR curve (AUC-PR) provides a

way to summarise model performance over the range of recall values. A perfect model will have an AUC-PR of 1, while the zero-skill model will have an AUC-PR equal to the proportion of positive cases in the dataset. The PR curve can provide a more accurate view of model performance on imbalanced datasets. (Saito & Rehmsmeier, 2015)

4.3.4 Graph Neural Network Training

4.3.4.1 Optimiser The Adam optimiser (Kingma & Ba, 2017) is used to train the GNNs with an initial learning rate of 0.01.

4.3.4.2 Neural Architecture Search For each of the GNN models (GraphSAGE and kGNN), we learn an optimal neural architecture for the anomalous node classification task. We use the Optuna library (Akiba et al., 2019) to explore the search space, train candidate models on the training dataset and select the architecture that achieves the highest AUC-PR on the validation data.

Each model is trained for a maximum of 2000 epochs, with an early stopping callback used to terminate trials that do not improve for 200 consecutive epochs. Fifty trials are performed for each model. Both GNN models share the same search space, the parameters of the space are provided in the appendix tables 3 and 4.

4.3.4.3 Hyperparameter Tuning Parameters for dropout and weight decay are also tuned using Optuna. These trials occur after the architecture search, to limit the dimensionality of the search space in each experiment. 20 trials are performed for each pair of candidate values, with the best hyperparameters selected based on the AUC-PR score on the validation set.

4.3.4.4 Weight Initialisation The GNN models are initialised with random weights. To ensure that the final model is not trained with a sub-optimal set of initial weights, multiple candidate models are trained with the chosen architecture and hyperparameters. The model with the highest AUC-PR on the validation set is selected as the final model.

4.3.5 CatBoost Training

The CatBoost classifier is trained similarly to the GNNs. Each candidate model is trained and evaluated on the same training and validation sets as the GNN models and evaluated using the AUC-PR metric. The hyperparameter search space is provided in the appendix tables 5 and 6.

5 Results and Discussion

5.1 Model Performance

Both of the GNN models achieved higher AUC-ROC and AUC-PR scores than the CatBoost model. The kGNN model achieved the highest AUC-ROC and AUC-PR scores, with an AUC-ROC of 0.982 and an AUC-PR of 0.904. The GraphSAGE model achieved an AUC-ROC of 0.943 and an AUC-PR of 0.735. The CatBoost model achieved an AUC-ROC of 0.639 and an AUC-PR of 0.104.

95% confidence intervals are provided for the AUC-ROC and AUC-PR scores. These are calculated using the bootstrap method (Davison & Hinkley, 1997) for 10,000 iterations.

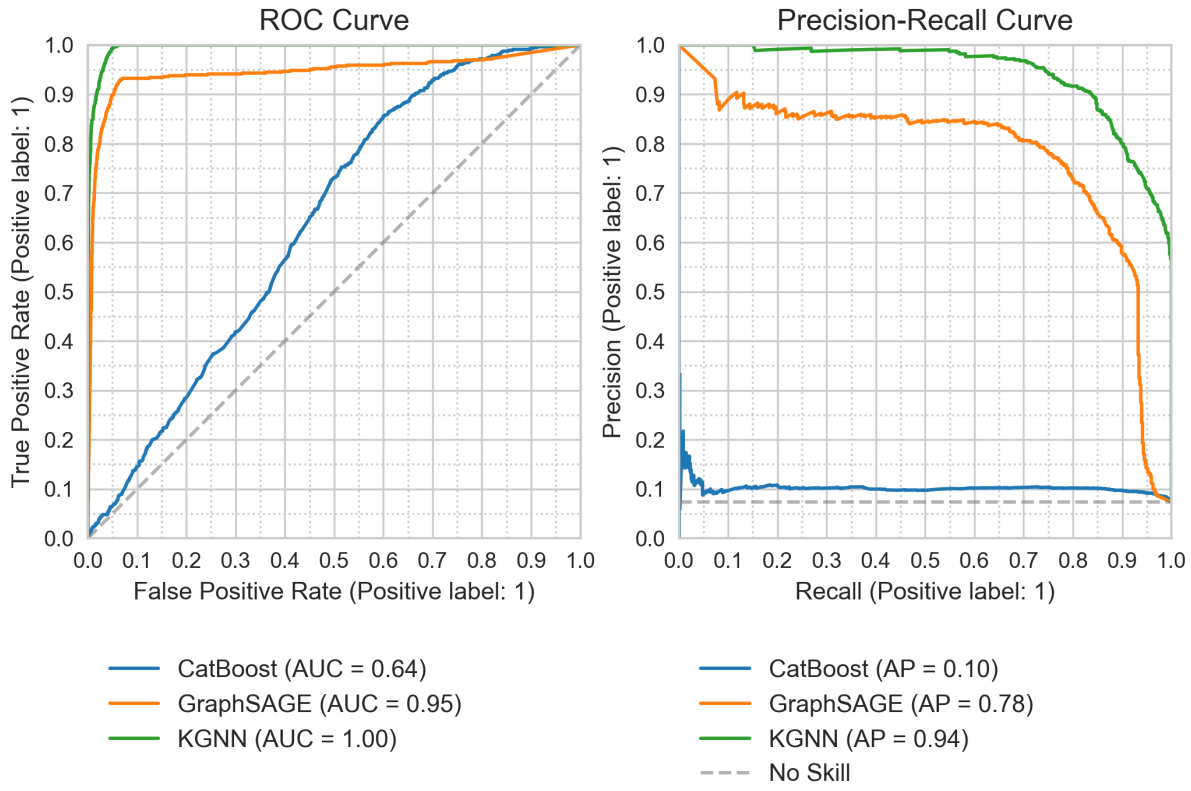

Figure 7: ROC and PR curves on the test set.

Table 2: Model performance on the test set.

Model	AUC-ROC	95% CI	AUC-PR	95% CI
CatBoost	0.639	[0.619, 0.659]	0.104	[0.092, 0.116]
GraphSAGE	0.945	[0.931, 0.959]	0.776	[0.736, 0.816]
kGNN	**0.995**	[0.993, 0.997]	**0.945**	[0.931, 0.959]

We note the CatBoost model's performance is little better than an unskilled classifier in terms of its AUC-PR score, suggesting that it is of little value for fraud detection in this scenario. In answer to our first research question (Q1), we conclude the GNN models are superior to the CatBoost model for the anomalous node classification task. Further, the increase in performance on this task more than justifies the additional cost in terms of complexity and computational resource required to train the GNN models. To answer our second research question (Q2), our best performing kGNN model achieves an uplift of 55.7% in the AUC-ROC score over the CatBoost model, and an uplift of 808.7% in AUC-PR score over the CatBoost model.

The outstanding performance of the kGNN model may be explained by its capacity to learn and combine higher-order features of the graph. A potential avenue for further study would be to investigate what features the model is learning and how they influence the model's performance. Work by Ying et al. (2019) on the GNNExplainer tool is likely worth exploring.

5.2 Neural Architecture Search and Hyperparameter Tuning

5.2.1 GraphSAGE

The best performing architecture for the GraphSAGE model was a 7-layer model with 128 hidden channels, no bias term, and an additional linear layer after the final message-passing layer. A min aggregation function was selected for both the message-passing layers and the aggregation of heterogeneous node representations. A leaky ReLU activation function was selected for all layers. Neither dropout nor weight decay were beneficial to model performance.

5.2.2 kGNN

The best performing architecture for the kGNN model was a 5-layer model with 128 hidden channels, a bias term, and no additional linear layer after the final message-passing layer. A max pooling aggregation function was selected for the message-passing layers, while min was used for the aggregation of

heterogeneous node representations. A ReLU activation function was selected for all layers. Neither dropout nor weight decay were beneficial to model performance.

5.2.3 CatBoost

The best performing CatBoost was trained with hyperparameters:

- learning rate: 0.09
- depth: 9
- boosting type: Plain
- bootstrap type: MVS
- colsample by level: 0.08

5.2.4 Notes and Recommendations

It was observed during the model tuning process that both GNN models displayed a great deal of variance in performance, even between trials when all hyperparameters were held constant. This suggests a high deal of sensitivity to the initialisation of the model weights.

To ensure that the final model is among the best possible candidates, we perform several rounds of training with the best performing architecture, checking validation performance at each epoch and saving the weights whenever the validation AUC-PR improves beyond the previous best value. This process is repeated for each of the GNN models, and the final models are selected based on the best validation AUC-PR score. This is in contrast to the common strategy of training the final model for a predetermined number of epochs on both the training and validation datasets.

Further, both models exhibited a non-linear progression in performance during training and demonstrated no signs of over-fitting. While this could suggest that the models have not converged, allowing the models to train over thousands of epochs showed that the validation AUC-PR continues to increase and decrease in a non-linear fashion. It may be worth studying this behaviour in future work to determine whether the training instability results from the model architecture, the dataset, or some other factor in the experimental design. Validation APRC histories for the 3 best and 3 worst GraphSAGE and kGNN random weight initialisations are shown in figures 11 and 12.

These observations should be noted as potential pitfalls in the model tuning process, and in answer to our third research question (Q3), we would advise caution when training GNN models for industrial applications, especially in critical areas such as fraud detection. Our recommendation is to follow a similar process to that detailed in this study, of staged model training and hyperparameter tuning, with regular check-pointing to ensure that the final model is optimal for the task.

5.3 Further Work

5.3.1 Graph Attention Networks

During the initial planning of this study, we intended to include a Graph Attention Network (GAT) model (Veličković et al., 2018) in the model comparison. However, it was found during the model-tuning process that models using attention mechanisms used more memory than the other models, and could not be trained on the full dataset. This was likely because of the large number of nodes in the dataset, and that the GAT model is a more computationally expensive model than the other models considered. Further studies may wish to investigate the application of attention mechanisms to the fraud detection task, with a proposed solution being to split the dataset into smaller sub-graphs and train the model over batches of nodes and edges. Further developments iterating on the transformer architecture include models that are designed to learn on heterogeneous graphs, such as the Heterogeneous Graph Transformer (Hu et al., 2020) and Heterogeneous Graph Attention Network (X. Wang et al., 2021).

5.3.2 Model Architectures

The GNN architecture search spaces for this study are simple compared to the realm of possible model architectures. Investigating developments such as Jumping Knowledge layers (Xu et al., 2018) and the prospect of using different aggregation functions for each edge type may reveal further improvements in model performance, or the possibility of training equally performant models with fewer parameters.

It is possible to restate the task in this study as one of edge classification. The motivation for framing the problem as one of node classification is primarily to facilitate comparison with traditional ML benchmark models, which require a tabular data structure. Duan et al. (2020) proposes Anomaly Aware Network Embedding (AANE) and demonstrates performance on real-world datasets. If successful on the anomalous business ownership detection task, the results would be useful for fraud detection, as the model should be able to identify anomalous edges, rather than identifying entire nodes as anomalous.

5.3.3 Datasets and Simulation Strategies

The random assignment of anomalous edges makes it difficult to reason about the models' performance, and it would be interesting to investigate the performance of the models on a dataset where the anomalies are generated in a more controlled manner. Investigating alternative simulation strategies could help to understand under what conditions the GNN models can detect anomalies and where they struggle. It could also be worth performing the same anomaly simulation and identification exercise on established benchmark datasets, such as the Cora citation network (McCallum et al., 2000), to see if the results are consistent.

The ICIJ dataset is another potential source of data for training a business ownership anomaly classifier. Several difficulties would need to be overcome, not least of which is linking this data to another source of legitimate company relationships to serve as negative fraud examples. This is a significant challenge because the companies listed by ICIJ are registered in various legal jurisdictions. However, having a known set of positive fraud examples would help build the external validity of any subsequent study and offer valuable insight into these models' potential for detecting fraud.

6 References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework* (No. arXiv:1907.10902). arXiv. https://doi.org/10.48550/arXiv.1907.10902

Akoglu, L., McGlohon, M., & Faloutsos, C. (2010). Oddball: Spotting Anomalies in Weighted Graphs. In M. J. Zaki, J. X. Yu, B. Ravindran, & V. Pudi (Eds.), *Advances in Knowledge Discovery and Data Mining* (Vol. 6119, pp. 410–421). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-13672-6_40

Akoglu, L., Tong, H., & Koutra, D. (2014). *Graph-based Anomaly Detection and Description: A Survey* (No. arXiv:1404.4679). arXiv. https://doi.org/10.48550/arXiv.1404.4679

Apache Spark: A unified engine for big data processing: Communications of the ACM: Vol 59, No 11. (2022, June 13). https://dl.acm.org/doi/10.1145/2934664

Beneficial Ownership Data Standard. (2022, September 18). openownership.org. https://www.openownership.org/en/topics/beneficial-ownership-data-standard/

Bondy, A., & Murty, U. S. R. (1982). *Graph Theory*. Springer London. https://books.google.com?id=5Z71jwEACAAJ

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2014). *Spectral Networks and Locally Connected Networks on Graphs* (No. arXiv:1312.6203). arXiv. https://doi.org/10.48550/arXiv.1312.6203

Busbridge, D., Sherburn, D., Cavallo, P., & Hammerla, N. Y. (2019). *Relational Graph Attention Networks* (No. arXiv:1904.05811). arXiv. https://doi.org/10.48550/arXiv.1904.05811

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Companies House. (2022, June 12). http://download.companieshouse.gov.uk/en_pscdata.html

Companies House data products. (n.d.). GOV.UK. Retrieved September 18, 2022, from https://www.gov.uk/guidance/companies-house-data-products

Davison, A. C., & Hinkley, D. V. (1997). *Bootstrap Methods and their Application*. Cambridge University Press. https://doi.org/10.1017/CBO9780511802843

Defferrard, M., Bresson, X., & Vandergheynst, P. (2017). *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (No. arXiv:1606.09375). arXiv. https://doi.org/10.48550/arXiv.1606.09375

Ding, K., Li, J., Bhanushali, R., & Liu, H. (2019). *Deep Anomaly Detection on Attributed Networks* (pp. 594–602). https://doi.org/10.1137/1.9781611975673.67

Duan, D., Tong, L., Li, Y., Lu, J., Shi, L., & Zhang, C. (2020). AANE: Anomaly Aware Network Embedding for Anomalous Link Detection. *2020 IEEE International Conference on Data Mining (ICDM)*, 1002–1007. https://doi.org/10.1109/ICDM50108.2020.00116

Dumitrescu, B., Bălteanu, A., & Budulan, Ş. (2022). Anomaly Detection in Graphs of Bank Transactions for Anti Money Laundering Applications. *IEEE Access*, *10*, 47699–47714. https://doi.org/10.1109/ACCESS.2022.3170467

European Commission. Directorate General for Taxation and Customs Union. (2019). *Estimating international tax evasion by individuals.* Publications Office. https://data.europa.eu/doi/10.2778/300732

European Union Agency for Law Enforcement Cooperation. (2021). *Shadow money: The international networks of illicit finance.* https://op.europa.eu/publication/manifestation_identifier/PUB_QLAN21003ENN

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Fernández. (2018). *Learning from Imbalanced Data Sets* (1st ed. 2018 edition). Springer.

Fey, M., & Lenssen, J. E. (2019). *Fast Graph Representation Learning with PyTorch Geometric* (No. arXiv:1903.02428). arXiv. https://doi.org/10.48550/arXiv.1903.02428

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 1189–1232.

Fronzetti Colladon, A., & Remondi, E. (2017). Using social network analysis to prevent money laundering. *Expert Systems with Applications*, *67*, 49–58. https://doi.org/10.1016/j.eswa.2016.09.029

Gori, M., Monfardini, G., & Scarselli, F. (2005). A new model for earning in raph domains. In *Proceedings of the International Joint Conference on Neural Networks* (Vol. 2, pp. 734 vol. 2). https://doi.org/10.1109/IJCNN.2005.1555942

Hamilton, W. L., Ying, R., & Leskovec, J. (2018). *Inductive Representation Learning on Large Graphs* (No. arXiv:1706.02216). arXiv. https://doi.org/10.48550/arXiv.1706.02216

He, H., & Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

Heterogeneous Graph Learning — PyTorch Geometric documentation. (n.d.). Retrieved September 19, 2022, from https://pytorch-geometric.readthedocs.io/en/latest/notes/heterogeneous.html

Hu, Z., Dong, Y., Wang, K., & Sun, Y. (2020). *Heterogeneous Graph Transformer* (No. arXiv:2003.01332). arXiv. https://doi.org/10.48550/arXiv.2003.01332

ICIJ. (2021, October 3). *Offshore havens and hidden riches of world leaders and billionaires exposed in unprecedented leak - ICIJ*. https://www.icij.org/investigations/pandora-papers/global-investigation-tax-havens-offshore/

Keeping your people with significant control (PSC) register. (2016, April 6). GOV.UK. https://www.gov.uk/government/news/keeping-your-people-with-significant-control-psc-register

Kingma, D. P., & Ba, J. (2017). *Adam: A Method for Stochastic Optimization* (No. arXiv:1412.6980). arXiv. https://doi.org/10.48550/arXiv.1412.6980

Kipf, T. N., & Welling, M. (2017). *Semi-Supervised Classification with Graph Convolutional Networks* (No. arXiv:1609.02907). arXiv. https://doi.org/10.48550/arXiv.1609.02907

Leskovec, J. (2020). *Traditional Methods for Machine Learning in Graphs*. Lecture. http://snap.stanford.edu/class/cs224w-2020/slides/02-tradition-ml.pdf

Li, Y., Huang, X., Li, J., Du, M., & Zou, N. (2019). *SpecAE: Spectral AutoEncoder for Anomaly Detection in Attributed Networks* (No. arXiv:1908.03849). arXiv. https://doi.org/10.48550/arXiv.1908.03849

Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2017). *Gated Graph Sequence Neural Networks* (No. arXiv:1511.05493). arXiv. https://doi.org/10.48550/arXiv.1511.05493

Luna, D. K., Palshikar, G. K., Apte, M., & Bhattacharya, A. (2018). Finding shell company accounts using anomaly detection. *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, 167–174. https://doi.org/10.1145/3152494.3152519

Ma, X., Wu, J., Xue, S., Yang, J., Zhou, C., Sheng, Q. Z., Xiong, H., & Akoglu, L. (2021). *A Comprehensive Survey on Graph Anomaly Detection with Deep Learning*. http://arxiv.org/abs/2106.07178

Ma, Y., & He, H. (Eds.). (2013). *Imbalanced Learning: Foundations, Algorithms, and Applications* (1st edition). Wiley-IEEE Press.

McCallum, A. K., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, *3*(2), 127–163. https://doi.org/10.1023/A:1009953814988

Molloy, I., Chari, S., Finkler, U., Wiggerman, M., Jonker, C., Habeck, T., Park, Y., Jordens, F., & Schaik, R. (2016, February 22). *Graph Analytics for Real-time Scoring of Cross-channel Transactional Fraud*.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., & Grohe, M. (2021). *Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks* (No. arXiv:1810.02244). arXiv. https://doi.org/10.48550/arXiv.1810.02244

Open Ownership. (2022, May 24). openownership.org. https://www.openownership.org/en/

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). *The PageRank Citation Ranking: Bringing Order to the Web*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Peng, Z., Luo, M., Li, J., Xue, L., & Zheng, Q. (2022). A Deep Multi-View Framework for Anomaly Detection on Attributed Networks. *IEEE Transactions on Knowledge and Data Engineering*, *34*(6, 6), 2539–2552. https://doi.org/10.1109/TKDE.2020.3015098

People with significant control (PSCs). (2022, September 18). GOV.UK. https://www.gov.uk/guidance/people-with-significant-control-pscs

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2019). *CatBoost: Unbiased boosting with categorical features* (No. arXiv:1706.09516). arXiv. https://doi.org/10.48550/arXiv.1706.09516

Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, *10*(3), e0118432. https://doi.org/10.1371/journal.pone.0118432

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, *20*(1, 1), 61–80. https://doi.org/10.1109/TNN.2008.2005605

Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. van den, Titov, I., & Welling, M. (2017). *Modeling Relational Data with Graph Convolutional Networks* (No. arXiv:1703.06103). arXiv. https://doi.org/10.48550/arXiv.1703.06103

Steven M., D. (2019, May 21). *Combating Illicit Financing by Anonymous Shell Companies — FBI* [Testimony]. https://www.fbi.gov/news/testimony/combating-illicit-financing-by-anonymous-shell-companies

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). LINE: Large-scale Information Network Embedding. *Proceedings of the 24th International Conference on World Wide Web*, 1067–1077. https://doi.org/10.1145/2736277.2741093

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). *Graph Attention Networks* (No. arXiv:1710.10903). arXiv. http://arxiv.org/abs/1710.10903

Wang, D., Lin, J., Cui, P., Jia, Q., Wang, Z., Fang, Y., Yu, Q., Zhou, J., Yang, S., & Qi, Y. (2019). A Semi-supervised Graph Attentive Network for Financial Fraud Detection. *2019 IEEE International Conference on Data Mining (ICDM)*, 598–607. https://doi.org/10.1109/ICDM.2019.00070

Wang, X., Ji, H., Shi, C., Wang, B., Cui, P., Yu, P., & Ye, Y. (2021). *Heterogeneous Graph Attention Network* (No. arXiv:1903.07293). arXiv. https://doi.org/10.48550/arXiv.1903.07293

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., & Jegelka, S. (2018). Representation Learning on Graphs with Jumping Knowledge Networks. *Proceedings of the 35th International Conference on Machine Learning*, 5453–5462. https://proceedings.mlr.press/v80/xu18c.html

Ying, R., Bourgeois, D., You, J., Zitnik, M., & Leskovec, J. (2019). *GNNExplainer: Generating Explanations for Graph Neural Networks* (No. arXiv:1903.03894). arXiv. https://doi.org/10.48550/arXiv.1903.03894

7 Appendix

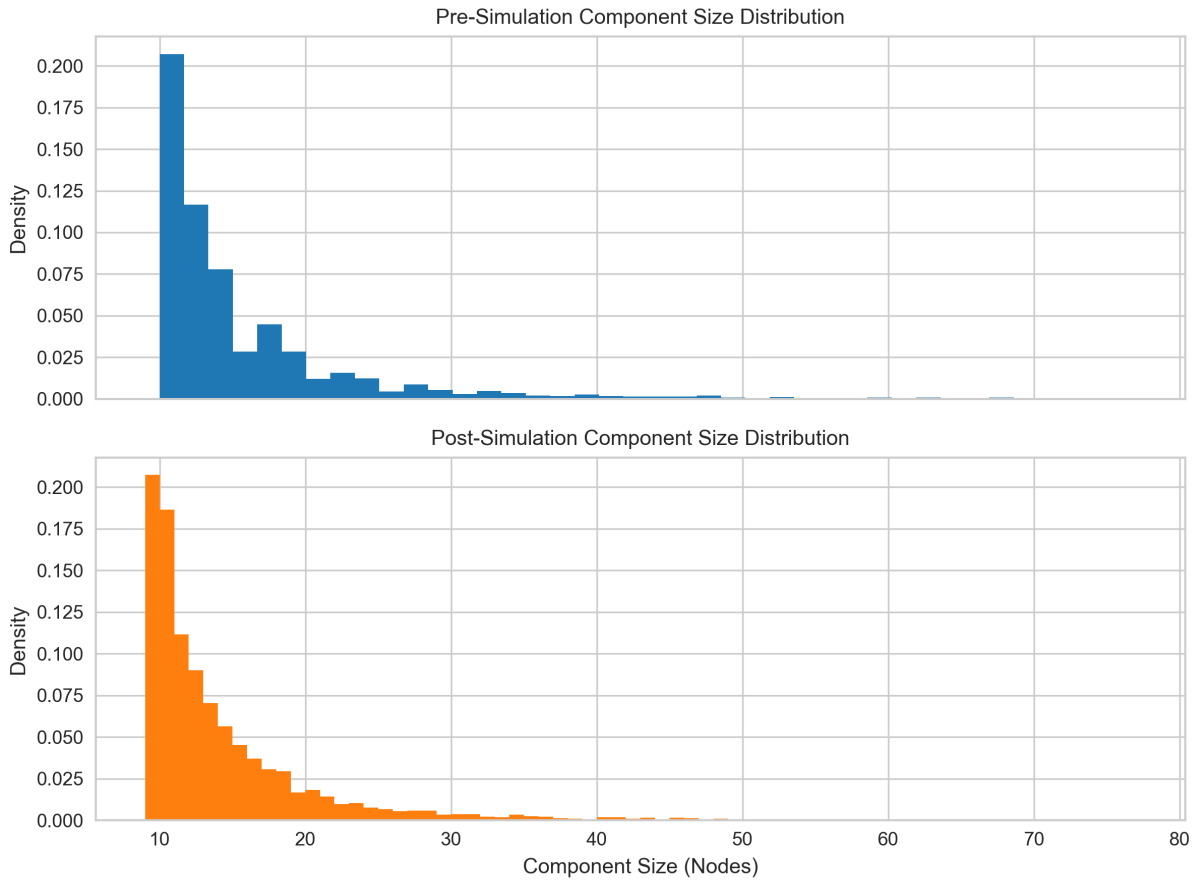

Figure 8: Distribution of component sizes before and after anomaly simulation.

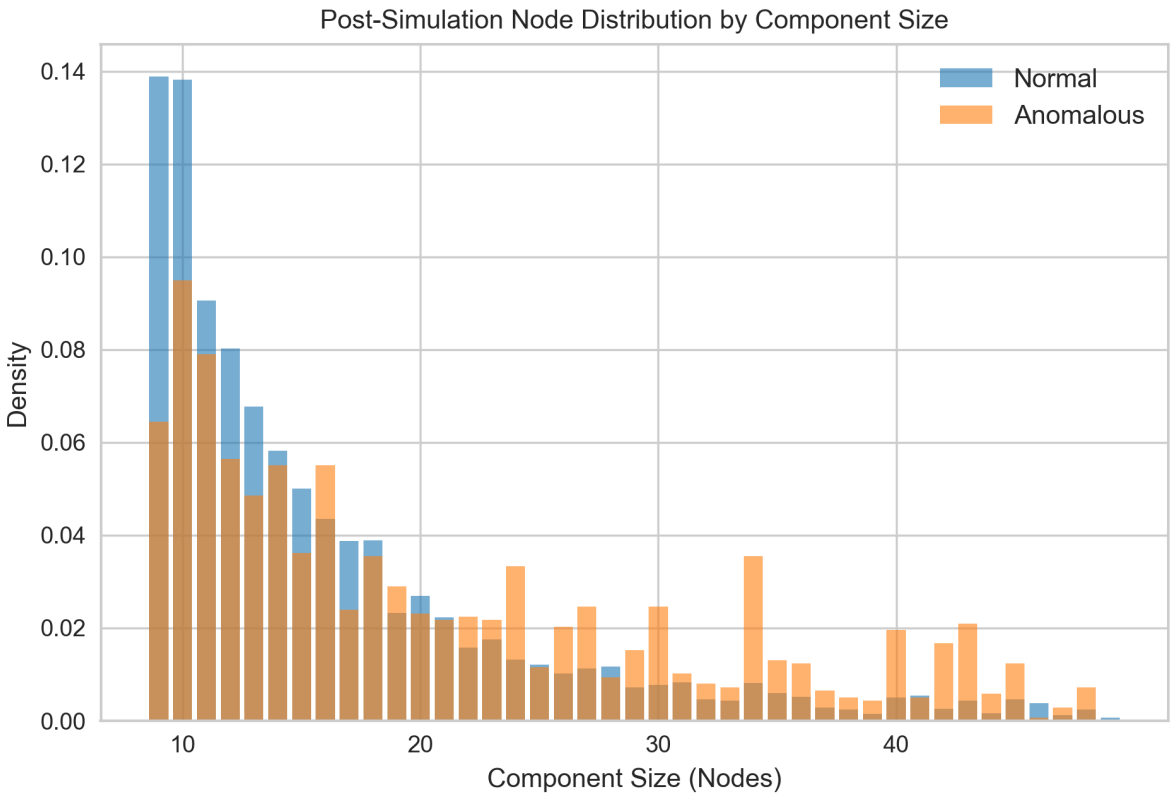

Figure 9: Distribution of anomalous nodes by component size before and after anomaly simulation.

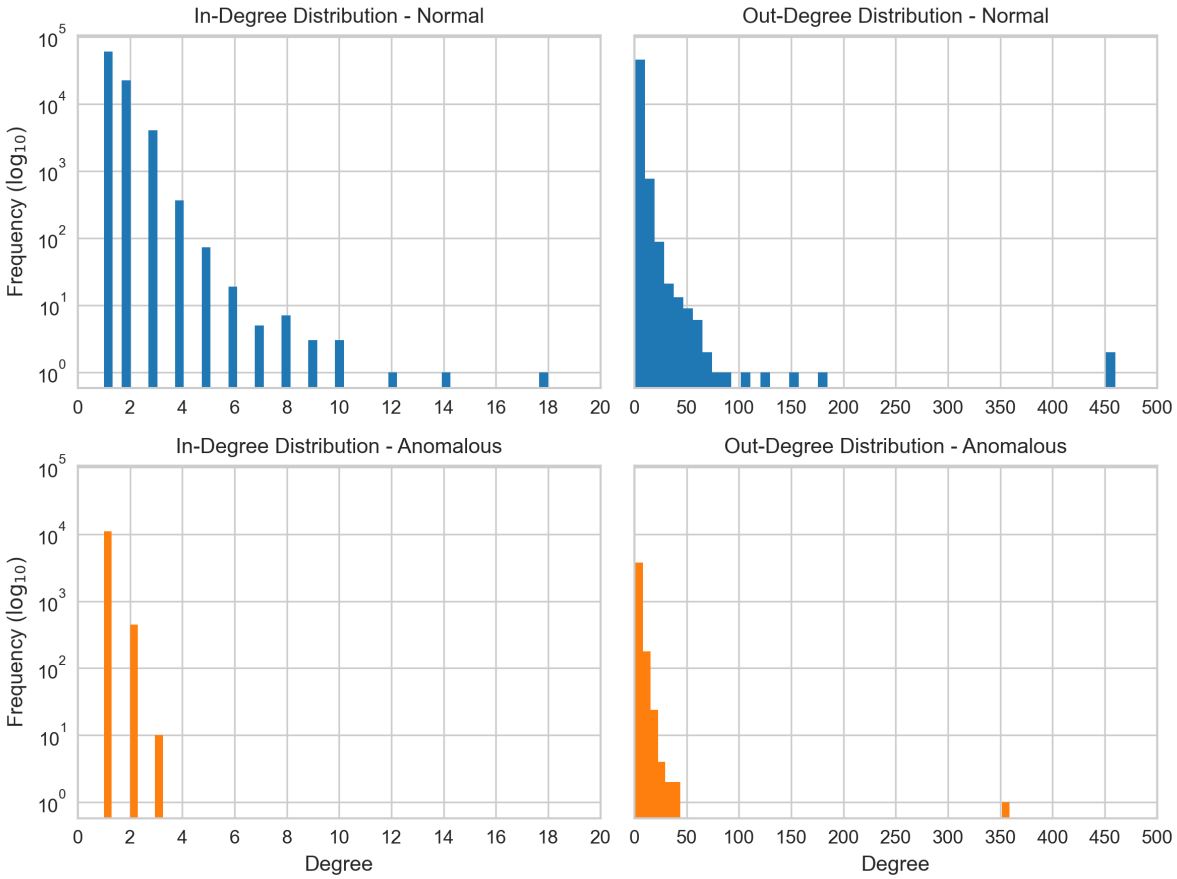

Figure 10: Distribution of node degrees before and after anomaly simulation.

Table 3: Neural architecture search space for GNN models - continuous.

Property	Min	Max
hidden layers	1	8
hidden channels	2	256
weight decay	0	0.01
dropout	0	0.5

Table 4: Neural architecture search space for GNN models - categorical.

Property	Choices
activation function	ReLU, GELU, leaky ReLU
linear layer post-message-passing	true, false
message-passing aggregation	min, max, sum, mean
heterogeneous embedding aggregation	min, max, sum, mean

Table 5: CatBoost hyperparameter search space - continuous.

Property	Min	Max
learning rate	0.001	0.1
colsample by level	0.01	0.1
depth	1	12
bagging temperature (If Bayesian bootstrap)	0	10
subsample (if Bernoulli bootstrap)	0.1	1

Table 6: CatBoost hyperparameter search space - categorical.

Property	Choices
boosting type	ordered, plain
bootstrap type	Bayesian, Bernoulli, MVS

Table 7: Hyperparameters for the best performing GNN models.

Hyperparameter	GraphSAGE	kGNN
Learning Rate	0.01	0.01
Optimiser	Adam	Adam
Activation	Leaky ReLU	ReLU
Depth	7	5
Hidden Channels	128	128
Bias	No	Yes
Linear Layer	Yes	No
Message Aggregation	Min	Max
Heterogeneous Aggregation	Min	Min
Dropout	No	No
Weight Decay	No	No

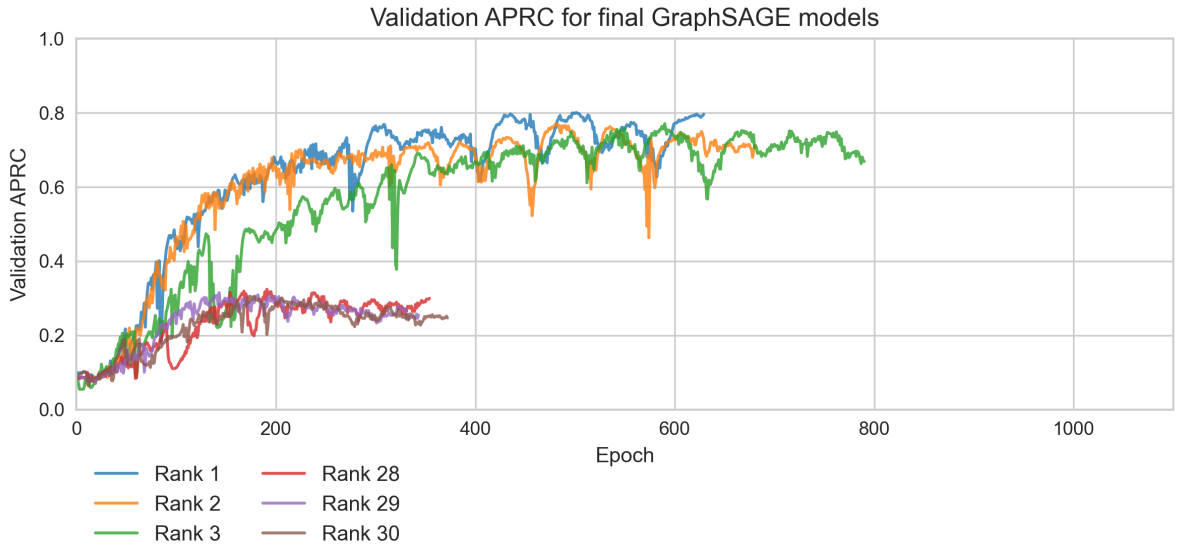

Figure 11: GraphSAGE: validation APRC history for the 3 best and 3 worst random weight initialisations. All models are built with the same best performing architecture and hyperparameters.

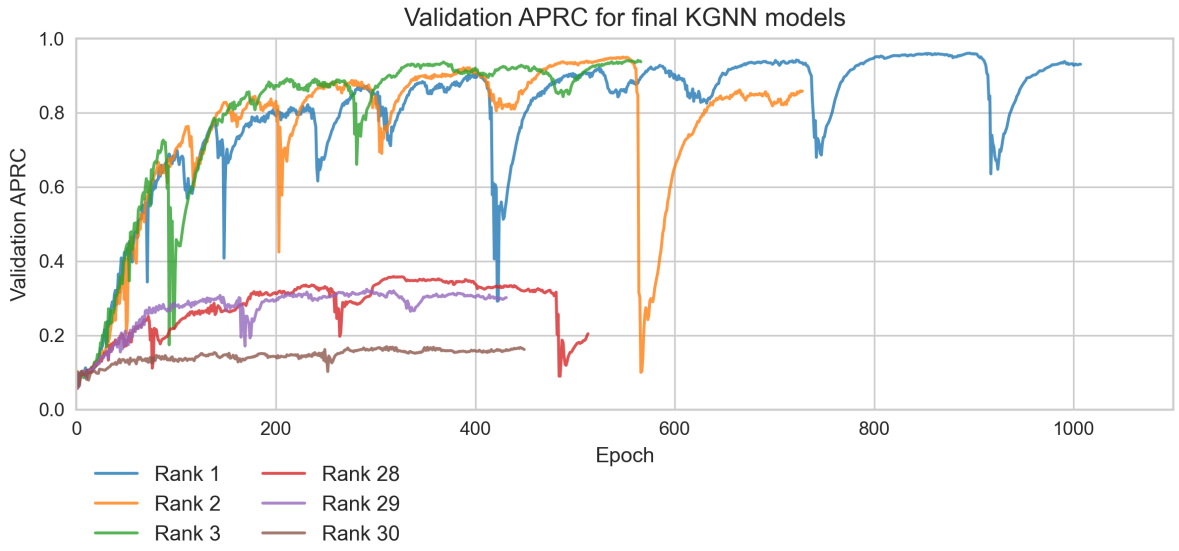

Figure 12: kGNN: validation APRC history for the 3 best and 3 worst random weight initialisations. All models are built with the same best performing architecture and hyperparameters.