Detecting Anomalous Business Ownership Structures with Graph Neural Networks

Final Project

Dominic Thorn

Contents

List of Tables

List of Figures

1 Introduction

1.1 Contributions

- Insights into training on a relatively large, novel dataset.

2 Prior and Related Work

3 Dataset

Training a supervised classification model requires access to a dataset of labelled examples. Given the typical infrequency of fraudulent events to legitimate cases, it is common for fraud classification projects require large amounts of data in order 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 publically available data.

3.1 External Sources

3.1.1 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 more than 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.

3.1.2 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 additional 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 for the generation of our UK business ownership graph.

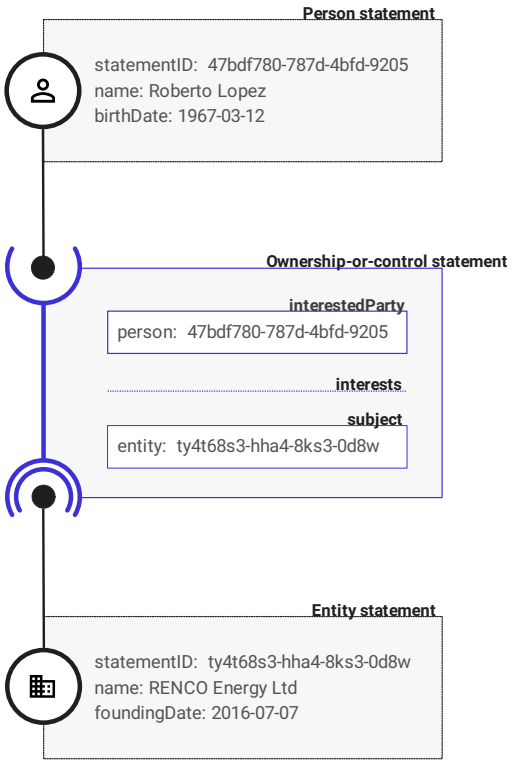

Figure 1: Open Ownership data schema (*Beneficial Ownership Data Standard*, 2022)

3.1.3 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.

3.2 Initial Data Preparation

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

As the scope of 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 are unable to obtain information for these entities via Companies House. Computational resource constraints also prevent 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.

3.3 Graph Generation

3.3.1 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 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 consists of two types of nodes with disitinct feature sets, it can be described as an attributed heterogeneous graph (X. Ma et al., 2021).

3.3.2 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 to be 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).

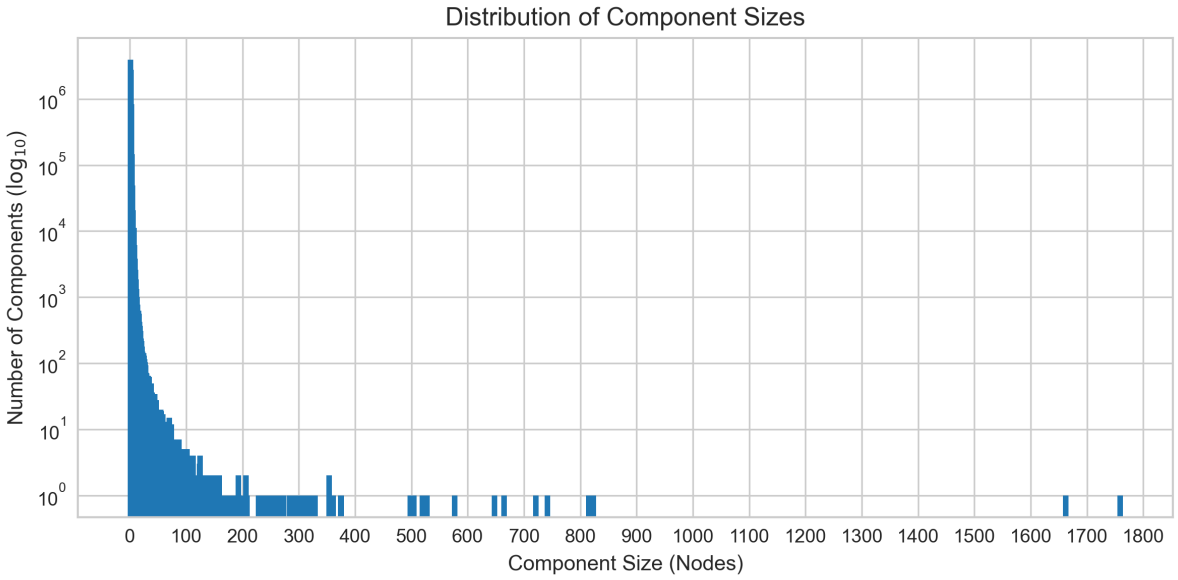

Figure 2: Initial distribution of component sizes. A single super-component consisting of 27,008 nodes is excluded from the plot.

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 subgraphs 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 usually 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.

Finally, 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.

3.4 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% 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 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 exchanging 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%.

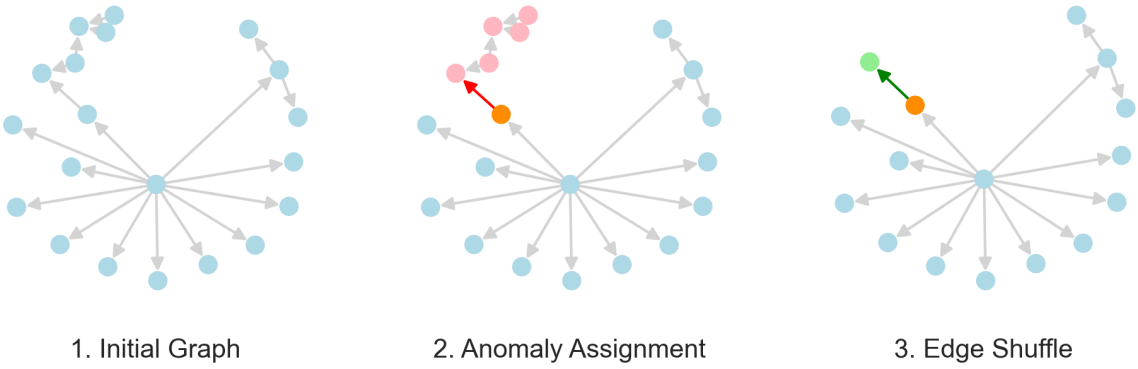

Figure 3: Process for simulating structural anomalies. The initial graph is shown in (1). A node is selected for anomalisation (orange circle) in (2). The outgoing edge (red arrow) is swapped for that of another anomalised node (green arrow), resulting in the exchange of pink nodes for green (3).

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

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

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

3.5 Node Features

In order to train a baseline model for comparison, a set of node level features is generated to represent each node in a tabular format (Leskovec, 2021). The following topological features are extracted for each node:

- Indegree: The number of incoming edges to the node.
- Outdegree: 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:

- Minimum
- Maximum
- Sum
- Mean
- Standard Deviation

The count of immediate neighbours is also included as a feature.

4 Methods

4.1 Traditional Machine Learning Approaches to Binary Classification

In order 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.

Since the GNN architectures selected for this study were originally implemented 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 individually. 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.

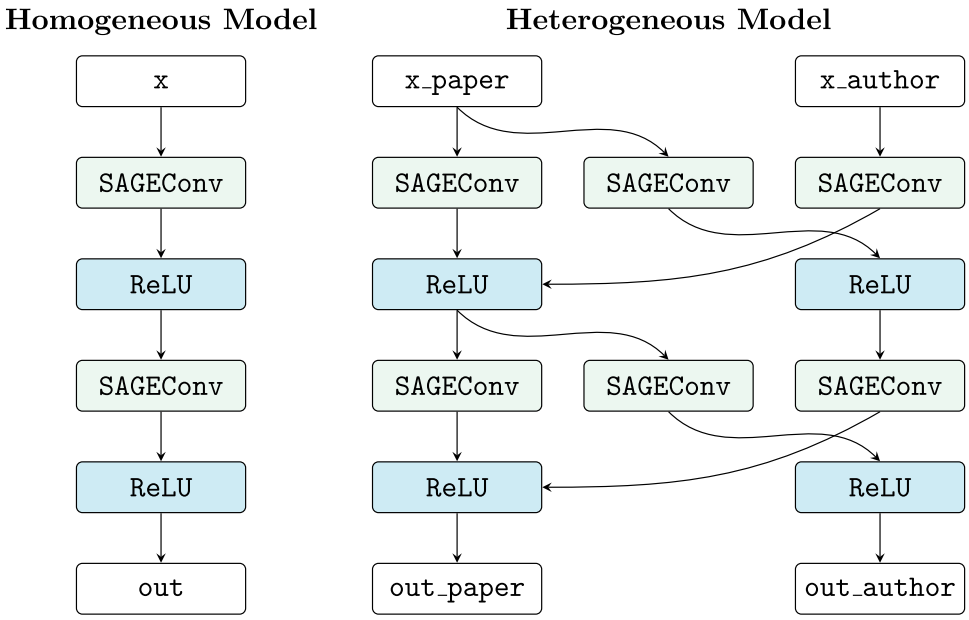

Figure 4: Converting homogeneous GNN architectures for heterogeneous learning (*Heterogeneous Graph Learning — Pytorch_geometric Documentation*, n.d.)

4.2.1 GraphSAGE

The GraphSAGE model proposed by Hamilton et al. (2018) is a node embedding framework that uses both node attributes and the attributes of neighbouring nodes to generate node representations. During training, the model learns how to effectively aggregate information from the neighbourhood of each node and combine this with the node's own features to produce a learned representation.

The aggregation architecture can be any symmetric function, such as the mean or sum, or a more 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.

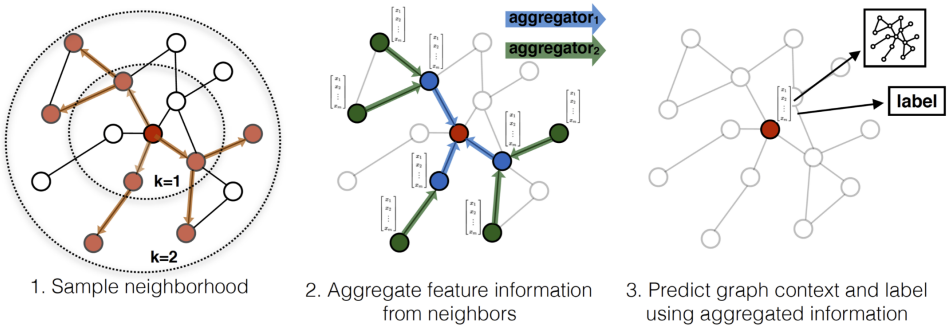

Figure 5: GraphSAGE neighbourhood sampling and aggregation (Hamilton et al., 2018)

4.2.2 Higher Order GNN (kGNN)

The Higher-Order GNN (or kGNN) proposed by Morris et al. (2021) is an extension of 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 takes into account 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.

Unlike the GraphSAGE model, the kGNN model does incorporate edge weights in its learning process.

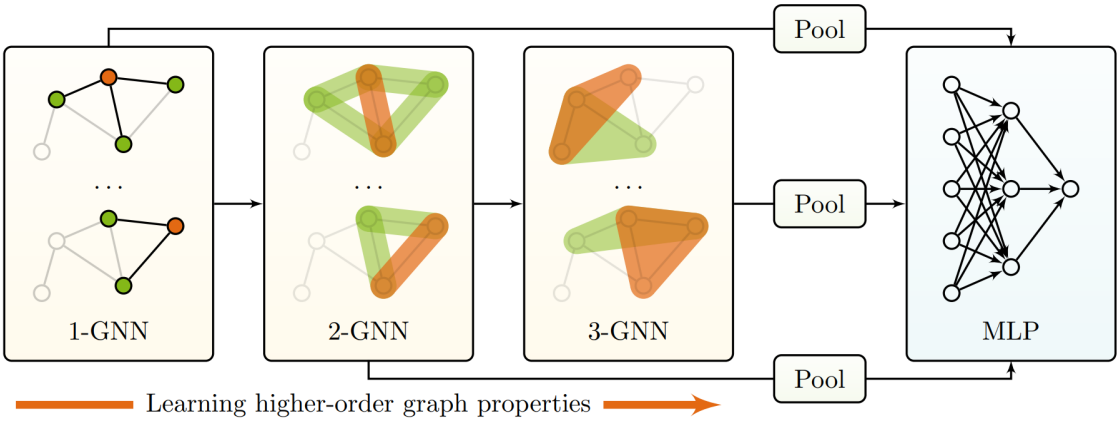

Figure 6: Hierarchical 1-2-3 GNN network architecture (Morris et al., 2021)

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. The split is performed using a random 80/10/10 split, with random selection applied to each component in the graph to ensures 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 increased penalty and greater emphasis for anomalous nodes. This is a cost sensitive approach to class imbalance that conveniently does not require the use of oversampling or undersampling (He & Garcia, 2009).

4.3.3 Evaluation Metrics

We use two threshold-free 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. Precision is defined as the proportion of positive predictions that are correct, while recall is the proportion of all positive cases that have been correctly identified. A perfect model will achieve a 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 along 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 peformance on imbalanced datasets. (Saito & Rehmsmeier, 2015)

4.3.4 Graph Neural Network Training

4.3.4.1 Optimiser The Adam optimiser (**kingmaAdamMethodStochastic2017?**) is used to train the GNNs with an initial learning rate of 0.01.

4.3.5 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 (**akibaOptunaNextgenerationHyperparameter2019?**) to explore the search space, training candidate models on the training dataset and selecting 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. Thirty trials are performed for each model. The search spaces used for each model are provided in the appendix tables {(**tbl:graphsage-search-space?**)} and {(**tbl:graph-conv-search-space?**)}.

4.3.6 Hyperparameter Tuning

Parameters for dropout and weight decay are also tuned using Optuna. These trials take place after the architecture search, in order to limit the dimensionality of the search space in each experiment. A total of 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.

4.3.7 CatBoost Training

The CatBoost classifier is trained in a similar manner 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 table {(**tbl:catboost-search-space?**)}.

5 Results

5.1 Neural Architecture Search and Hyperparameter Tuning

5.1.1 GraphSAGE

5.1.2 kGNN

The optimal architecture for the kGNN model was found to be a 4-layer model with 128 hidden channels and an additional linear layer after the final message passing layer. Neither dropout nor weight decay were found to be beneficial to model performance. A minimum pooling aggregation function was selected for both the message passing layers and for the aggregation of heterogeneous node representations. A gelu activation function was selcted for all layers.

5.1.3 CatBoost

The most successful CatBoost model was found to have the following parameters:

- learning rate: 0.09
- depth: 9
- boosting_type: Plain
- bootstrap_type: MVS
- colsample_bylevel: 0.08

5.2 Model Performance

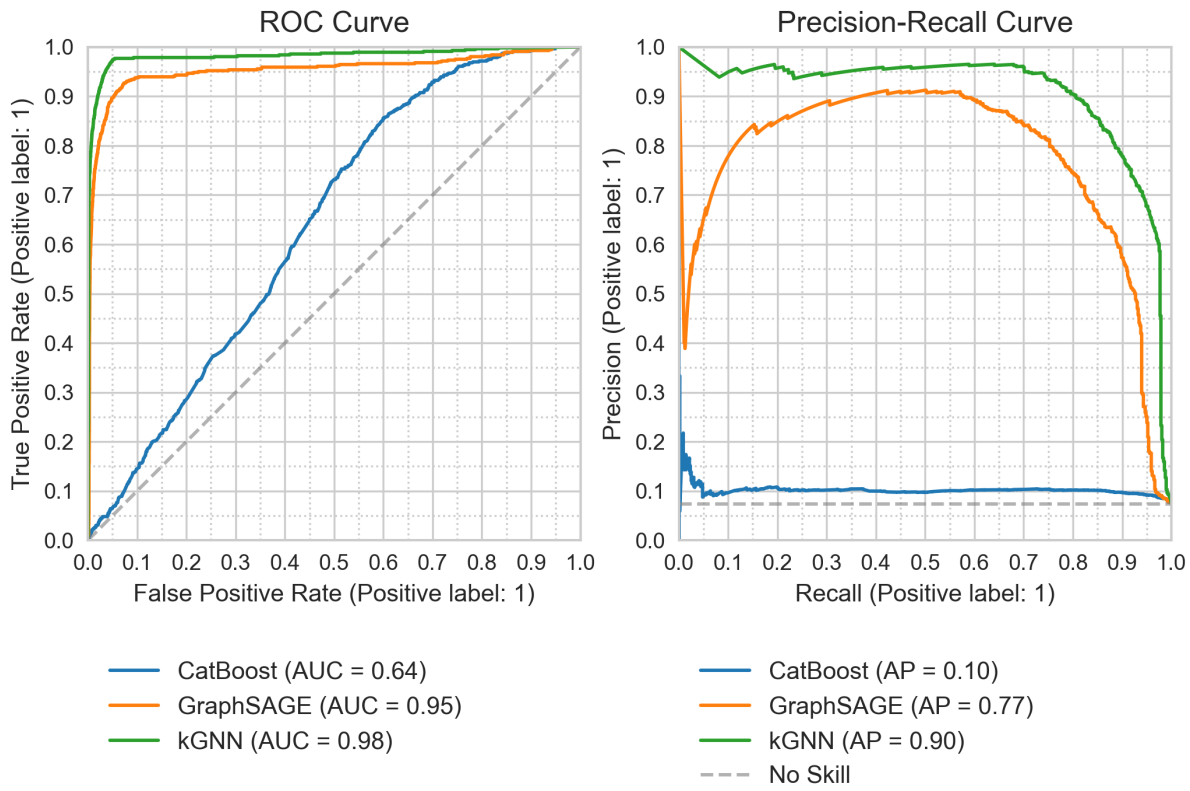

Figure 7: ROC and PR curves on the test set.

Table 1: Model performance on the test set.

Model	ROC AUC	95% CI	PR AUC	95% CI
CatBoost	0.639	[0.619, 0.659]	0.104	[0.092, 0.116]
GraphSAGE	0.953	[0.939, 0.967]	0.767	[0.723, 0.811]
kGNN	0.982	[0.974, 0.990]	0.904	[0.876, 0.932]

6 Conclusion

We have presented a novel dataset of ownership relationships between companies and persons in the UK. The dataset is made available for use in future research, and is accompanied by a set of baseline

models for comparison.

7 References

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

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

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

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.

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

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

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. (2021). *Traditional Methods for Machine Learning in Graphs*. Lecture. http://snap.stanford.edu/class/cs224w-2020/slides/02-tradition-ml.pdf

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.

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*.

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

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

8 Appendix

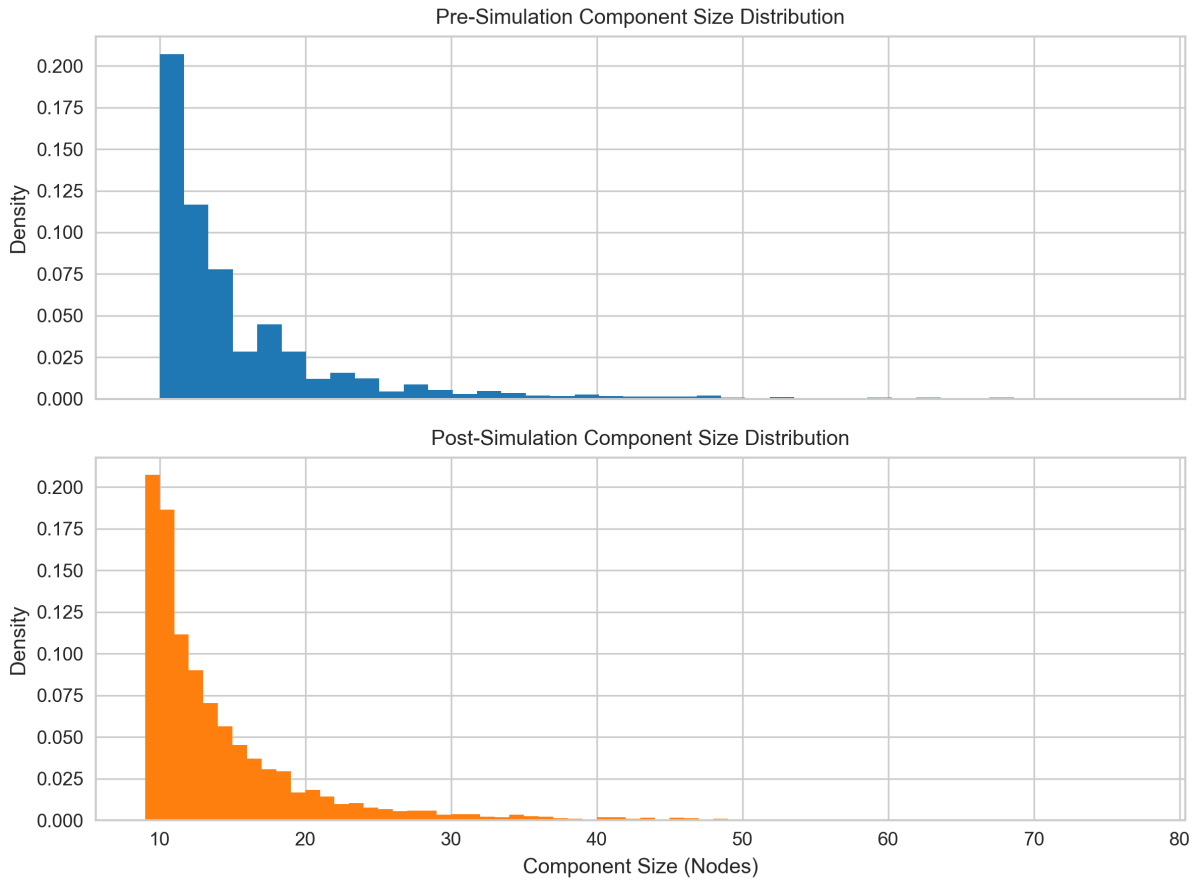

Figure 8: Distribution of component sizes before and after anomaly simulation.

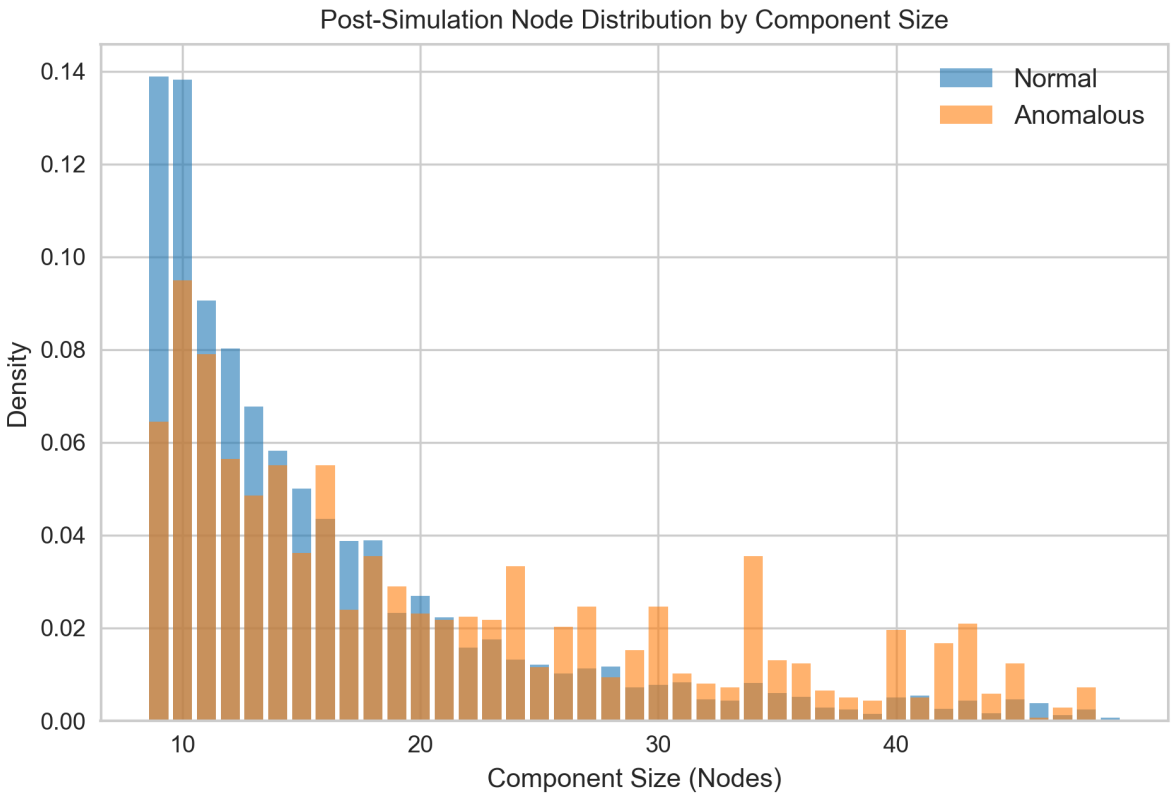

Figure 9: Distribution of anomalous nodes by component size before and after anomaly simulation.

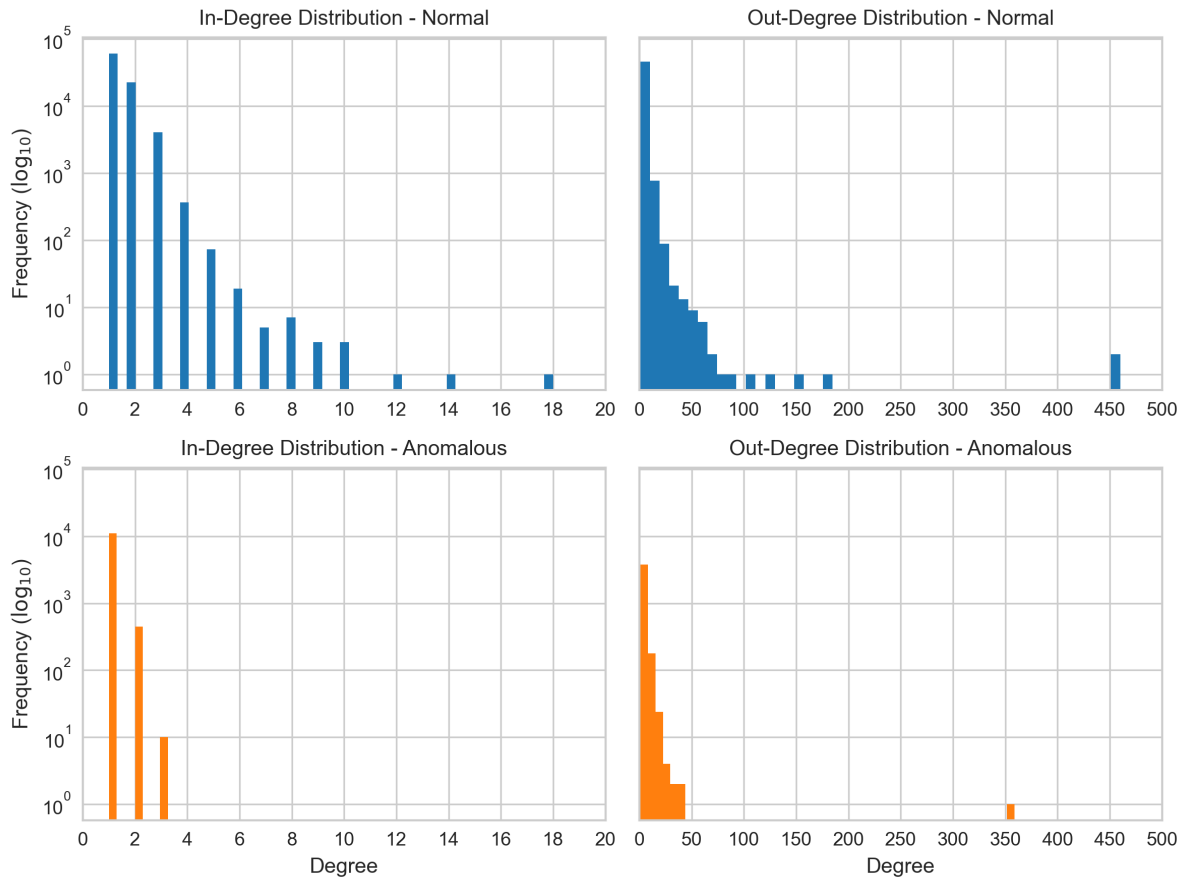

Figure 10: Distribution of node degrees before and after anomaly simulation.