TRANSPARENCY DOCUMENT

Manuscript Title: How a Rotten Apple May Spoil the Barrel: Corporate Fraud Detection via Dynamic Business Networks

1. Overview

The **Data Availability and Provenance** section provides a detailed description of the data collection process and intermediate data processing steps, enabling full reproducibility of the dataset. Due to licensing restrictions, the original dataset cannot be shared publicly. Researchers with access to the CSMAR database can reconstruct the dataset using the provided instructions.

A **Variable Dictionary** lists all meta-data used in the research. For demonstration purposes, only a small sample of the data is provided, as the full dataset remains restricted by license.

The required **Computational Environment** is specified, including all necessary Python packages and their corresponding versions.

All source code files are clearly organized and documented in **Programs/Code** section. To replicate the experiments, execute **python main.py** with the specified parameter settings.

By following these instructions, researchers can reproduce the results reported in the paper.

2. Data Availability and Provenance

Primary data collection. The primary data used in this study was collected from the China Stock Market & Accounting Research Database (CSMAR). It includes:

- **Fraud information** of listed companies, retrieved from the *Financial Irregularity Table of Listed Companies*.
- **Financial ratios**, obtained from the *Accounting Information Quality - Financial Indicators*.
- **Non-financial features**, retrieved from the *Database of Governance Structure of Listed Companies*.
- **Management discussion and analysis (MD&A) text** from annual reports, collected from *Database of Operational Difficulties - Management Governance*.
- **Four types of business relationships:**
  - **Shareholder** information of listed companies was collected from Shareholder database.
  - The **employment** information of managers from listed companies was collected from the Corporate Governance Structure Database.
  - The **kinship** data was sourced from the Equity Relationship Database.
  - The **RPTs** data was collected from the Related Party Transaction Database.

Intermediate data processing. The intermediate data processing consists of three parts: (1) conventional feature processing, (2) business network construction, and (3) fraud sample collection.

(1) Conventional Feature Processing:

- Calculate financial ratios from raw financial data and fill missing values.
- Process non-financial ratios and remove company-year records with missing features.

➓ Process MD&A text data by removing stopwords, tokenizing, and fine-tuning a pre-trained model to obtain linguistic feature embeddings.

(2) Business Network Construction:

➓ Standardize entity names, perform named entity recognition (NER), deduplicate records, merge entities, and match person names with the individual IDs provided by CSMAR.

➓ Integrate the four types of relationship data, count the number of each meta-path in the business networks, and save the graph for each meta-path into a sparse matrix file.

(3) Fraud Sample Collection:

➓ Exclude invalid enforcement actions (e.g., missing firm names or dates, financial industry cases, unrelated to financial statement fraud, or restatements without delay).

➓ Generate successive time-series for each fraud sample, covering the fraud year and up to 5 years prior to the fraud.

➓ Remove company-year records that lack the conventional features to create the positive sample set.

➓ Treat all remaining non-fraud company-year records as the negative sample set.

➓ Integrate positive and negative samples and divide them into training, validation and testing set according to calendar years.

3. Variable Dictionary

The data dictionary that lists key variables in our dataset is illustrated as follows:

- **Stkcd_year**: a 2-D tensor storing the stock ID and calendar year for each sample.

- **label**: a 1-D tensor indicating the fraud type (1) or non-fraud type (0) of a sample.
- **fin_ratio**: a 2-D tensor storing all financial ratios for fraud and non-fraud samples.
- **nonfin_ratio**: a 2-D tensor storing all non-financial features.
- **mda_bert**: a 2-D tensor storing all linguistic features of MD&As.
- **fraud_series:** a tensor of sequences reporting the historical fraud records of samples.
- **fin_seq**: index sequences of the financial ratios for each sample.
- **nonfin_seq**: index sequences of the non-financial features for each sample.
- **mda_seq**: index sequences of the linguistic features for each sample.
- **node_seq**: index sequences of for each sample.
- **seq_len**: a 1-D tensor reporting the sequence length of the time series for each sample.
- **seq_mask**: a 2-D tensor that records the padding positions in time series of samples.
- **graphs**: yearly graphs compressed by 26 meta-paths.
- **hetegraphs**: heterogeneous graphs that store the original triples of business networks.
- **sample_idx_by_year**: node subscripts in the heterogeneous graph for the listed companies within the sample set.

4. Computational Environment

The packages and corresponding versions used to run the code are listed as follows:

- Python >= 3.8.0
- numpy >= 1.24.2
- torch >= 1.12.0
- dgl >= 1.1.2

- sklearn >= 1.3.2
- scipy >= 1.10.1
- matplotlib>=3.7.5
- re >= 2.2.1
- json >= 2.0.9
- argparse >= 1.1
- shap>=0.44.1
- xgboost>=1.6.2
- yaml>=5.1.2

5. Programs/Code

All code is provided in the source code package. The functionality of each file or directory is summarized below:

- **dataloader.py**: handles intermediate stages of data preparation for experiments.
- **attention.py**: implements the basic attention mechanism and mean pooling module.
- **loss.py**: implements multiple loss functions used in training NetDetect, including *Focal Loss*, *Reweight Loss*, and *Reweight Revision Loss*.
- **model.py**: implements the NetDetect framework.
- **main.py**: implements the training, validating, testing, and evaluation processes of experiments.
- **utils.py**: provides basic functions required for model training, such as generating masks, setting random seeds, and early stopping.

- **attn_visual.py:** outputs hierarchical attention weights in case study.
- **t_test.py:** used to conduct paired t-test
- **shap_test.py:** output Shapley values and the corresponding figure
- **the directory of "config" in src_clean:** a set of .yaml files that define the parameter settings of all models
- **output/CN/significance_test:** stores the original experimental results used for paired t-test on the CSMAR dataset.
- **output/CN/results_replicate:** contains saved model checkpoints and logs from the main experiments.
- **output/CN/results:** a directory used to store the outputs from user replications.
- **output/CN/ significance_test:** a directory used to store the output of paired t-test replications.