INFORMS Journal on Data Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Credit Risk Modeling with Graph Machine Learning

Sanjiv Das, Xin Huang, Soji Adeshina, Patrick Yang, Leonardo Bachega

To cite this article:
Sanjiv Das, Xin Huang, Soji Adeshina, Patrick Yang, Leonardo Bachega (2023) Credit Risk Modeling with Graph Machine Learning. INFORMS Journal on Data Science 2(2):197-217. https://doi.org/10.1287/ijds.2022.00018

https://pubsonline.informs.org/journal/ijds

INFORMS JOURNAL ON DATA SCIENCE
Vol. 2, No. 2, October–December 2023, pp. 197–217
ISSN 2694-4022 (print), ISSN 2694-4030 (online)

Credit Risk Modeling with Graph Machine Learning

Sanjiv Das,[a,b,*] Xin Huang,[c] Soji Adeshina,[a] Patrick Yang,[d] Leonardo Bachega[d]

[a] Amazon Web Services, Santa Clara, California 95053; [b] Santa Clara University, Santa Clara, California 95053; [c] Amazon Web Services, New York, New York 10001; [d] Amazon Web Services, Seattle, Washington 98109

*Corresponding author

Contact: srdas@scu.edu, https://orcid.org/0000-0003-2168-0909 (SD); xinxh@amazon.com (XH); adesojia@amazon.com (SA); yhuichun47@gmail.com (PY); bachega@amazon.com (LB)

Received: June 14, 2022
Revised: February 27, 2023; September 25, 2023; October 24, 2023
Accepted: October 25, 2023
Published Online in Articles in Advance: November 20, 2023

https://doi.org/10.1287/ijds.2022.00018

Abstract. Accurate credit ratings are an essential ingredient in the decision-making process for investors, rating agencies, bond portfolio managers, bankers, and policy makers, as well as an important input for risk management and regulation. Credit ratings are traditionally generated from models that use financial statement data and market data, which are tabular (numeric and categorical). Using machine learning methods, we construct a network of firms using U.S. Securities and Exchange Commission (SEC) filings (denoted CorpNet) to enhance the traditional tabular data set with a corporate graph. We show that this generates accurate rating predictions with comparable and better performance to tabular models. We ensemble graph convolutional networks with highly-performant ensembled machine learning models using AutoGluon. This paper demonstrates both transductive and inductive methodologies to extend credit scoring models based on tabular data, which have been used by the ratings industry for decades, to the class of machine learning models on networks. The methodology is extensible to other financial machine learning models that may be enhanced using a corporate graph.

History: David Martens served as the senior editor for this article.

Data Ethics & Reproducibility Note: No data ethics considerations are foreseen related to this article. The paper deals with corporate credit risk and not consumer credit, which usually entails issues around privacy and bias. The code capsule is available on Code Ocean at https://codeocean.com/capsule/5230264/tree/v2 and in the e-Companion to this article (available at https://doi.org/10.1287/ijds.2022.00018).

Keywords: credit ratings • machine learning • corporate graph • graph neural networks

1. Introduction

The credit risk of companies has been modeled for many decades. In fact, possibly the first use of machine learning in finance began with the work of Altman (1968), who used a linear discriminant analysis (LDA) classifier to predict corporate defaults over horizons of one year or more. Altman's Z-score model uses just five features (publicly available variables) yet is highly accurate and effective, the model is parsimonious and robust, and the coefficients are stable over time. These features are based on income statement and balance sheet ratios, and in the case of public firms, also include the market value of the firm's equity.[1]

Since Altman's seminal work, modeling credit risk has specialized with various follow-on models. (1) *Accounting-based* models such as Altman (1968) and Ohlson (1980) used financial ratios to predict bankruptcy. These models use mainly balance sheet and income statement data as features to fit the models. (2) *Structural* models discussed in Black and Scholes (1973) and Merton (1974) use equity returns and volatilities in a stochastic differential equation framework to estimate probabilities of default, which are then mapped to ratings. These models have seen extensive industrial use as they were popularized by Moody's/KMV (Kealhofer, McQuown, and Vasicek) and have been used globally (Falkenstein et al. 2000, Sobehart et al. 2000, Dwyer et al. 2004). The philosophy behind structural models is that the best information about the credit quality of a firm comes from the equity markets. A comparison of the efficacy of accounting-based models versus structural models is presented in Das et al. (2009). Chan-Lau (2006) offers a good review of structural models for credit risk modeling, and Altman (2018) provides a 50-year retrospective on credit risk models. (3) *Reduced-form* models, such as those by Jarrow and Turnbull (1995) and Duffie and Singleton (1999), determine probabilities of default (and consequently, ratings) using bond and credit default swap spreads. Here the philosophy is that credit risk information is easiest to extract accurately from corporate bonds and more recently from credit default swaps. Hilscher and Wilson (2016) argue that more than one measure of credit risk is ideally necessary and that probability of default should be used in addition to ratings.

In this paper, we take a complementary approach and introduce network information into credit modeling. To exploit this information, we build machine learning models using graph neural networks (GNNs). Networks are ubiquitous in physical and social domains. A large

number of these are scale-free networks (Barabasi and Bonabeau 2003), typically visualized as hub-and-spoke graphs. Firms in the economy are interconnected and credit risk that emerges in some firms will spill over onto other firms, depending on the network structure of connections between firms. In the case of small and medium enterprises (SMEs), relational information improves bankruptcy prediction (Tobback et al. 2016), and for financial firms, an analysis of default risk linkages using networks has been undertaken in Billio et al. (2012), Das (2016), Kok and Montagna (2016), and Das et al. (2019). These papers have used graph theory to generate features (such as centrality, node degree, etc.) that are added to tabular data sets for econometric analysis, remaining in the realm of two-dimensional tabular data structures, where models are fitted to features derived from graphs but do not use the graphs in entirety themselves. The same ideas have been modeled for credit scoring individuals in social media networks, as in Wei et al. (2016). An interesting extension to centrality from multilayer (bipartite) graphs is provided in some papers (Poledna et al. 2015, Gupta and Kumar 2021, Óskarsdóttir and Bravo 2021), where credit models are improved using multiple types of relational information. Deep learning on graphs enables the exploitation of high-dimensional, nontabular data structures that may represent the world better and lead to improved models for predicting credit quality. Additionally, networks are effective structures for representing and understanding complex systems (Benson et al. 2016) such as the corporate ecosystem in the global economy. Machine learning on networks piggybacks on graph theory for tasks such as node classification, link prediction, network similarity, community detection (Fortunato 2010), improving question-answering systems, and so on. Using GNNs, we extend these related works to the credit scoring of firms.

Modeling credit scoring with networks has seen recent success in improving models for lending to individuals and in furthering financial inclusion. Muñoz-Cancino et al. (2021) provide assessment of the credit-worthiness of thin-file borrowers; Poenaru-Olaru (2020) discuss scoring SMEs; Ballen (2021) discusses scoring of online credit card borrowers; and Shumovskaia et al. (2021) discuss bank clients creditworthiness for loans. Network information is also used to study multiparty loans (Cheng et al. 2020), and complex relationships are studied in agricultural lending (Bai et al. 2019). Our research in this paper complements the work in individual credit scoring for loans with an application to scoring corporations' credit risk, for which network relationships are harder to come by.

We address the question of corporate network construction as well. Various approaches are possible. (1) *Time series approach*: Construct the corporate graph using stock return spillovers, where nodes represent firms. This approach is used in Billio et al. (2012) and generates a directed graph using Granger causality regressions—a link from corporate firm A to firm B is added to the graph if the lagged stock return of node A explains (at a chosen level of statistical significance) the current return of node B in a regression that also contains the lagged return of B as a regressor. Likewise, the reverse regression would generate a link from node B to node A if the lagged value of B's stock return explained the current return of A. (2) *Physical linkages*. Construct the graph using data on business relationships such as banking relationships (Das et al. 2021) or supply chain relationships (Surana et al. 2005, Rios et al. 2020). Supply chain relationships are likely drivers of credit risk spillovers. (3) *Reports-based relationships*. This graph construction technique is an innovation in this paper and uses textual data in U.S. Securities and Exchange Commission (SEC) filings (10-Q/10-Ks, quarterly/annual reports) to construct links in a network based on similarity in the management discussion and analysis (MD&A) sections of the 10-K filings by firms. The MD&A section discusses current and future business conditions and companies that have similar text are likely to be related as they mention the same risk factors, markets, industry conditions, and other business features. The benefit of this approach is that SEC filings are publicly available to everyone, and this work may be easily replicated. Furthermore, the text in the MD&A section covers a wider range of mechanisms through which companies may be related than only specific linkages such as business relationships. A limitation of this approach is that it can only be applied to firms that file 10-Q/10-K reports with the SEC, although the universe of these firms is large (more than 8,000 companies filed 10-Q/10-Ks with the SEC in 2021–2022).

The contributions of this paper are as follows. First, we present a new approach for construction of corporate networks using SEC filings. Second, we show how traditional credit rating models based on tabular data may be enhanced with the addition of a corporate graph. Third, we also show how a GNN model may be ensembled with traditional tabular machine learning (ML) models; we see that ensembling improves overall classifier performance. Whereas our use case focuses on credit ratings, the approaches here may be applied to several other models in the business world to bring in network information because our network construction approach is fungible across applications that use corporate data.

The rest of this paper proceeds as follows. Section 2 provides a brief review of graph machine learning. Section 3 discusses the two datasets used in this paper, real and synthetic. Methodology is presented in Section 4. The analysis and results are presented in Section 5. Section 6 offers concluding comments.

2. Overview of GNNs

ML using graph data are a broad subject and good expositions are available in books by Hamilton (2020), Blumauer and Nagy (2020), and Ma and Tang (2021).

As a first step, graph ML begins with representing the information on graphs in suitable form for ingestion into ML algorithms. Representation learning, as this is broadly denoted, entails conversion of information for each entity in the graph into a numerical vector of fixed dimension d. Graph entities are the nodes of the graph, the links between nodes, and the graph itself. Any or all of these may be represented as a vector in d-dimensional space. In the nomenclature of machine learning, representing entities as numerical vectors allows each entity to be embedded as a coordinate in this d-dimensional vector space, and therefore, these representations are also known as "embeddings." Given the various graph entities, there are node, link, and graph embeddings. The second step in graph machine learning is fitting a neural network to use the embeddings for classification. Both steps are undertaken simultaneously using an end-to-end neural network architecture.[2]

We set some terminology and notation. A graph (or network) G comprises nodes or vertexes (set V, of size $|V| = n$) and links or edges (set E, of size $|E| = m$) and hence may be written as $G(V, E)$. The edge set may be (i) directed or undirected and (ii) weighted or unweighted. The graph may have single node types and single edge types, in which case it is denoted as a "homogeneous" graph, but if it has multiple types of nodes or links, then it is known as a "heterogeneous" graph. (For example, in a lending network, some nodes may be borrowers, others lenders, and some both.) The number of links each node has is the "degree" of the node, and the distribution of number of links across all nodes is denoted the "degree distribution" of the graph. For a directed graph, we can have out-degree and in-degree, depending on how many links emanate from or into a node, respectively.

The links in a homogeneous graph are often represented by an "adjacency matrix" A of size $n \times n$, which represents all links in the graph with nonzero values, such that a link from node i to node j is designated by $A_{ij} > 0$. For an undirected graph, matrix A is symmetric, that is, $A_{ij} = A_{ji}$. For an unweighted graph, $A_{ij} = \{0, 1\}$. An alternate data structure for a graph is a links table L of size $m \times 3$, with one row for each link from node i to node j with edge weight w, which we can denote as $L(i, j, w)$. The links table is a parsimonious data structure compared with the adjacency matrix, and for large (and sparse) graphs, is memory efficient.

For the application in this paper, that is, credit rating of companies, because each firm is a node on the graph, we will train a GNN to create *node* embeddings and use these for classification. Node representations (embeddings) are vectors in d-dimensional embedding space, that is, mappings $f : V \to \mathcal{R}^d$ from vertices to fixed length vectors. The idea is to create embeddings that contain the features of each node and capture the position on the graph and features of the closely connected nodes. Various algorithms have been developed to create node embeddings:

1. *Graph Factorization*. One class of algorithms to generate these embeddings aims to map similarity of nodes in the graph to closeness in the embedding vector space. Let the d-dimensional embedding vectors for nodes i, j be denoted as $\boldsymbol{h}_i, \boldsymbol{h}_j \in \mathcal{R}^d$. The approach is to find vectors that map into the values in the adjacency matrix A, that is, minimize loss $\mathcal{L} = \frac{1}{2}\sum_i \sum_j (\boldsymbol{h}_i^\top \cdot \boldsymbol{h}_j - A_{ij})^2 + \frac{\lambda}{2}\sum_i \|\boldsymbol{h}_i\|^2$ (Ahmed et al. 2013). This is a simple approach but only uses information about direct (one-hop) links between nodes to construct the embeddings. It assumes that nodes that are a few hops away are dissimilar, which may not be the case. This is a shallow embedding but is fast to compute. The approach may be extended to include multihop links using powers of the adjacency matrix (Cao et al. 2015) or community overlap (Ou et al. 2016).

2. *Random Walks*. A similar approach based on random walks on the graph may be used such that the closeness in embedding space corresponds to the likelihood that nodes i and j will co-occur on the same random walk. We are interested in creating an embedding matrix $\mathbf{H} \in \mathcal{R}^{d \times n}$, where each column is the embedding vector for one node. The approach is as follows. Start short random walks of prespecified length from each node i and keep track of the multiset $N_R(i)$ (subscript R denotes random walk neighborhood) of all nodes visited; then optimize the d-dimensional embeddings to ensure that similarity of the embeddings is proportional to the node co-occurrence probabilities. The co-occurrence probabilities are computed using the SkipGram approach, that is, $P(j|\boldsymbol{h}_i) = \frac{\exp(\boldsymbol{h}_i^\top \cdot \boldsymbol{h}_j)}{\sum_{k \in V} \exp(\boldsymbol{h}_i^\top \cdot \boldsymbol{h}_k)}$. The objective is to maximize the likelihood: $\max_{\mathbf{H}} \sum_{i \in V} \ln P[N_R(i)|\boldsymbol{h}_i]$, $\forall i$. Noting that $P[N_R(i)|\boldsymbol{h}_i] = \prod_{j \in N_R(i)} P[j|\boldsymbol{h}_i]$, and using the previous formula for $P(j|\boldsymbol{h}_i)$, we can simplify the objective function to

$$\max_{\mathbf{H}} \sum_{i \in V} \ln P[N_R(i)|\boldsymbol{h}_i] \equiv \max_{\mathbf{H}} \sum_{i \in V} \left\{ \sum_{j \in N_R(i)} \boldsymbol{h}_i^\top \cdot \boldsymbol{h}_j - \ln \left[\sum_{k \in V} \exp(\boldsymbol{h}_i^\top \cdot \boldsymbol{h}_k) \right] \right\}. \quad (1)$$

See Perozzi et al. (2014) for details of an efficient approach for this algorithm to create node embeddings (known as `DeepWalk`). This method is also implemented with biased random walks in the `node2vec` algorithm (Grover and Leskovec 2016) to better explore diverse neighborhoods of the graph. Assessing which embedding generation approach is best depends on the machine learning task—some approaches may perform better for node prediction versus link prediction (Goyal and Ferrara 2018). For a broad survey and introduction to network embeddings, see Chen et al. (2018).

3. *Deep Embeddings*. The *shallow* node embeddings above ignore data available on node features. They also cannot generate embeddings for new nodes that were not part of the original training of node embeddings. Improved *deep* encodings for nodes using node-level features is possible with deep learning, using GNNs. For a detailed exposition, see Scarselli et al. (2009) and Zhou et al. (2020)—a brief overview follows.

The GNN formalism allows the embedding for a particular node to be expressed as a function of the node's features and the features of nodes (and edges) in the subgraph around the node. While there are many instances of GNN architectures that realize this function, they can be summarized by observing that they use neural message passing (Gilmer et al. 2017) to transform and exchange vector messages between nodes in the graph. For a r level deep GNN, the computation performed to generate the embedding for a node i is as follows. The output representation for node i at the rth layer, embedding $\boldsymbol{h}_i^{(r)}$, is the result of aggregating the vector representation of neighbors of i from the previous layer, that is, $Agg(\{\boldsymbol{h}_j^{(r-1)}\}), j \in N(i)$ into a message $m_i^{(r)}$, and then combining $m_i^{(r)}$ with $\boldsymbol{h}_i^{(r-1)}$ the previous layer representation for i. The embedding at the preceding $(r-1)$th layer is computed recursively using the vector representations and neighbours from it's previous layer until the first layer. At the first layer, the input vector representations are the original features of the node $x^{(i)}$ and the original features of it's neighbors. When no natural features are present, the features can be initialized via one hot encoding or random vectors. The choice and parameterization of the Aggregation function and Combine function is what distinguishes different GNN architectures. For example, Graph Convolutional Networks (GCN) (Kipf and Welling 2017) uses the Aggregation function: $\sum_{j \in N(i)} \frac{1}{c} \mathbf{W} \cdot \boldsymbol{h}_j^{(r-1)}$ and no Combine function. (Here, $\mathbf{W}$ are weights in the GNN, and c is a normalization constant (details provided in Section 4.6).) GraphSage (Hamilton et al. 2017), which we use in this paper, provides a number of options for the Aggregation function (GCN, Mean, Long Short Term Memory (LSTM)) and uses a single hidden layer neural network for the Combine function. For most GNN architectures, each layer represents a single hop of the graph since the computation at each layer depends on neighbors that are one hop away. Thus, an r layer–based embedding recursively uses r hops around each node to compute the final embedding of that node. The learning objective for training the GNNs depends on the downstream task. For a classification task, the embedding from the GNN can be fed into a simple one layer neural network classifier, and the overall network (GNN and classifier) can be trained end-to-end with the classification loss. When there is no downstream task, the model can be supervised by learning to predict existing edges in the graph also known as self-supervised learning. Node embeddings have also been extended to graph embeddings and subgraph embeddings (Duvenaud et al. 2015, Li et al. 2016, Chen et al. 2018). Section 4.6 goes into further details (with equations) of the deep node embeddings used in the paper.

3. Data

We demonstrate credit rating prediction on multimodal (graph and tabular) data using two datasets. First, we use an open data set that provides several accounting ratios that are widely used in the credit scoring of companies. Second, we use a data set of financial ratios used in the Altman (1968) Z-score model for which we generate synthetic ratios calibrated to national averages for the United States. (This synthetic data set is released to enable researchers to use the models in the paper.)

Both related data sets are augmented with networks derived from the text of SEC filings as exemplars for generating networks to augment the tabular data for use with graph machine learning. We first describe the real data set and then the synthetic one.

3.1. Data from Extant Companies

This data set comprises tabular data about the financials of several companies. We obtain the corporate credit ratings data set from Kaggle.[3] The data cover 2,029 ratings issued by the major rating agencies. For each firm, 25 features are used comprising numerical financial ratios from the balance sheet and income statement. These comprise the following:

1. *Liquidity Ratios*: current ratio, quick ratio, cash ratio, and days of sales outstanding;
2. *Profitability Ratios*: gross profit margin, operating profit margin, pretax profit margin, net profit margin, effective tax rate, return on assets, return on equity, return on capital employed, and EBIT to revenue;
3. *Debt Ratios*: debt ratio and debt equity ratio;
4. *Operating Performance Ratios*: asset turnover, fixed asset turnover, company equity multiplier, enterprise value multiple, and payable turnover; and
5. *Cash Flow Ratios*: operating cash flow per share, free cash flow per share, cash per share, operating cash flow to sales ratio, and free cash flow to operating cash flow ratio.

The feature set has the industry sector for each company (a categorical variable), date of the rating, the company name, rating agency that provided the rating, and the company ticker. The rating distribution in the data set is imbalanced, as is the case for ratings of companies in the broad U.S. economy. It comprises the following rating levels: (1) investment grade (AAA, AA, A, and BBB) and (2) below investment grade (junk) (BB, B, CCC, CC, C, and D). After aggregation of smaller classes, we have 96, 398, 671, 490, 302, and 72 samples in the {AAA, AA}, A, BBB, BB, B, and {CCC, CC, C, D} classes, respectively. There are 1,165 observations of investment grade

ratings, and 864 observations of below investment grade ratings; hence, the data set is only mildly imbalanced.

To augment this data with a network, for each firm and date combination, we retrieve SEC 10-K/Q filings for each quarter that the tickers appear in the data set. These are widely used for equity and credit analysis (the number of machine downloads of SEC 10-K and 10-Q filings grew from 360,861 in 2003 to 165,318,719 in 2016; Cao et al. 2020). We use a SEC retrieval engine in AWS SageMaker JumpStart[4] to download the 10-K/Q forms. A parser is used to extract the Management Discussion & Analysis (MD&A) section of the SEC filings and join this text with the tabular data to create an enhanced dataframe, denoted as "TabText" (Tabular+Text). These data are joined such that the financials data and ratings are in the same quarter as the SEC filing. Thus, the data frame now comprises a column of long-form text in addition to the numerical financial data and cardinal data on industry category. The text column is used to construct a graph based on similarity in the MD&A section for which details are provided in Section 4.1. This graph is denoted as `CorpNet` and depicts connected companies that might spill over credit risk to each other.

3.2. Synthetic Data Generation

To demonstrate our graph ML approach on a second credit data set, we synthetically generate financial data for use in a credit scoring model, using features from the Z-score approach of Altman (1968)[5] and combine it with graphs based on real textual data from SEC filings. The main steps in simulating the synthetic data set are as follows:

1. Download SEC quarterly report filings (10-K/Q forms) for a sample of companies and parse out the MD&A section. We did this to collect 3,286 filings.

2. Obtain numerical scores for each filing to assess the following 10 attributes, described in Appendix A.1: positivity, polarity, safety, sentiment, certainty, fraud, litigiousness, negativity, uncertainty, and risk. Each score is the percentage of the text that relates to each attribute, assessed using a lexicon of words for each attribute, generated using the algorithm in Das et al. (2022). We compute a net score by adding the scores on the first 5 attributes and subtract the scores on the last five attributes. This net score is used to rank companies and assign high ratings to companies with high net scores and poor ratings to companies with low net scores.

3. Synthetically generate financial statements using normalized average values for U.S. companies and compute Altman's Z-scores for these companies (high scores imply highly rated firms and low scores imply low credit quality firms). Generate the same number of companies as in step 1. This is described in detail in Appendix A.2.

4. (a) Join the text data in Step 1 above with ratings data from Step 2 above and sort it from high to low quality based on ratings. (b) Sort simulated financials in Step 3 based on Z-scores. Concatenate (a) and (b) by column to get a single synthetic data set.

Step 1 is already described in the last paragraph of Section 3.1. The appendix provides more detail on Steps 2, 3, and 4.

In the next section, we discuss the construction of corporate networks using the text from the MD&A section of the 10-K/Q SEC filings. We also present details of the GNN used for the models in the paper.

4. Methodology

We now present the modeling structure to fit various credit risk rating classifiers. The novelty of the methodology lies in the use of SEC reporting data to construct networks and the combination of the reported data, network construction with similarity, and GNNs to predict the credit rating. The contribution of this work also comes from the combination of these three processes into a prediction pipeline. We discuss network construction first. Then we describe how graph machine learning is applied to create better classifiers than those obtained from tabular ML. We also delineate the GCN specification and use two training approaches, inductive and transductive.

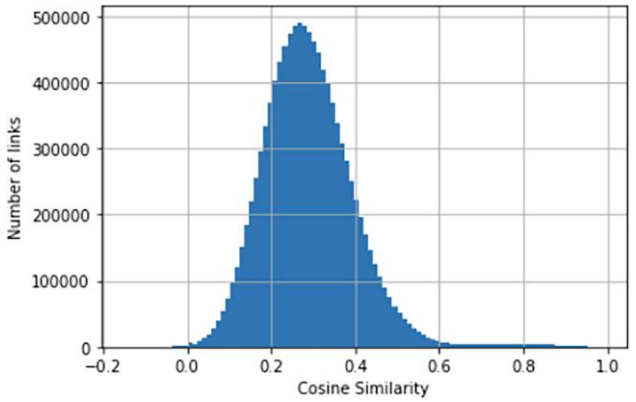
4.1. Network Construction from Text Data

We use AWS SageMaker JumpStart to download and parse SEC filings (10-K, 10-Q) and obtain specific sections as needed. AWS SageMaker provides a collection of tools for financial NLP, including downloading and parsing SEC filings into sections.[6] See also Cavar and Josefy (2018) for a nice coverage of tools for use with SEC filings. There is an extensive finance literature on generating insights from SEC filings (Li 2010; Brown and Tucker 2011; Loughran and Mcdonald 2011, 2014; Bonsall et al. 2017; Desola et al. 2019; Cao et al. 2020; Cohen et al. 2020; Bae et al. 2023). We intersect with this literature in using information from SEC filings to construct a network for graph machine learning.

Applying the `doc2vec`[7] algorithm to obtain document embeddings of the MD&A text for each firm, we construct an undirected and unweighted graph, as follows. First, we compute pairwise cosine similarity of MD&A text embeddings for all firms in the data set. Second, we use a threshold on cosine similarity for the network links, setting a symmetric, undirected, and unweighted link for all node pairs whose cosine similarity exceeds the threshold. Using this approach we create graphs for both the real and synthetic data sets.

For the 1,723 firms with complete data in the real data set, the distribution of cosine similarities is shown in Figure 1. The number of possible links is around 1.48 million, and after applying a cutoff of 0.5 for cosine similarity, we retain 29,126 links, with mean degree of 34. Figure 2 shows one rendering of the network, which exhibits clusters, although it is almost fully connected.

Figure 1. (Color online) Distribution of Cosine Similarities Between Document Embeddings of the MD&A Section of Firms' 10-K/Q Filings



Notes. The similarity of two firms is given by the following equation: $s = \frac{A \cdot B}{\|A\| \cdot \|B\|}$, where A, B are document embeddings. Because these embedding vectors may contain negative values, it is possible to return negative values. However, links in the network are based on values where $s > 0.5$.

This figure also shows the degree distribution, appearing as power law distributed, which is typical of hub and spoke networks, where a few nodes have a large number of links and most nodes have few.

For the synthetic data set, there are 3,285 nodes, where the number of possible links is around 5.30 million. After applying a cutoff of 0.5 for cosine similarity, we retain 180,961 links, with a mean degree of 110. We chose the cutoff of 0.5 a priori, without data snooping for the best cutoff, which can be a hyperparameter in the overall classifier. Later, we will vary this cutoff parameter to discover the cutoff that generates a graph that delivers the highest machine learning metrics. A graph with too many or too few links will be noninformative.

The choice of (i) embedding technique (`doc2vec`), (ii) distance metric, and (iii) cutoff is just one set of many available choices. We also tried other embeddings generators using transformers, including specialized financial transformers pretrained on Wiki text and SEC filings (these are denoted as RoBERTa-SEC models[8]). Unlike `doc2vec`, these transformer models resulted in networks that delivered lower accuracy in the GNNs we trained, possibly because we could only apply them sentence by sentence and then apply global average pooling. The MD&A sections in SEC filings are quite long (on average around 5,000 words) and are less amenable for use with transformers, which have low maximal sequence lengths. In addition to cosine similarity we also tried using Gaussian kernels for distance measurement with longformer transformers such as Big Bird (Zaheer et al. 2020), but this also did not deliver better results. Finally, our paper reports the results when different distance cutoffs are applied, and we note that this hyperparameter choice is an important one.

4.2. Classifiers

Our experiments implement a binary classification problem, where the graph data and tabular data are used to fit a model that classifies companies into investment grade (ratings AAA to BBB) or below investment grade (ratings BB and below). This is a canonical application in credit trading; moreover, the data sets are small and do not support multicategory classification. There are two ways in which graph information from `CorpNet` may be used to build credit risk rating models.

1. *Tabular ML*: We compute node-level (i.e., firm) attributes from the graph and add them to the tabular data described in Section 3—we include three graph-based attributes: degree centrality, eigenvector centrality, and clustering coefficient. These three metrics are used because they are already in use in the contagion literature (Poledna et al. 2015, Abbass et al. 2016, Das 2016, Helwege and Zhang 2016, Cai et al. 2018, Roukny et al. 2018, Wang et al. 2018), but they also capture notions of *interconnectedness*, which is specifically defined as one of the criteria for systemic risk in the context of systemically important financial institutions (SIFIs).[9]

Our three additional data columns for node features from the graph are as follows:

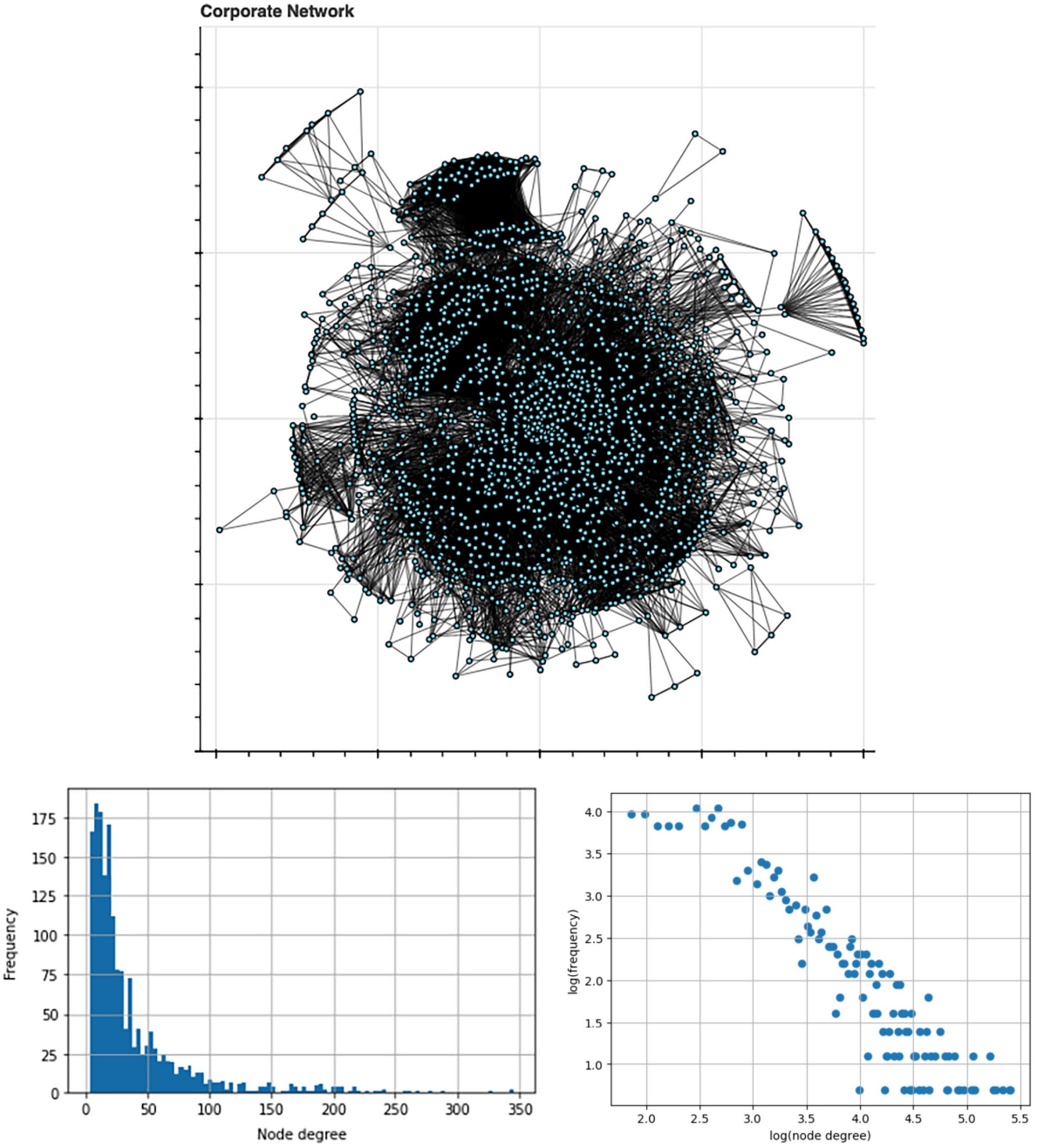
(a) Degree centrality[10] is the normalized number of links each node has, that is, the fraction of nodes a given node is connected to. For a network of n nodes, degree centrality of any node i is $|N(i)|/(n-1)$, where $|N(i)|$ is the size of the set of connected nodes of node i, that is, in its neighborhood $N(i)$;

(b) Eigenvector centrality[11] is a measure of connectedness of each node to other nodes, no matter how deeply connected. Eigenvector centrality computes the centrality for a node based on the centrality of its neighbors. This recursive relationship is captured in the eigensystem $\boldsymbol{A} \cdot \boldsymbol{x} = \lambda \boldsymbol{x}$, where $\boldsymbol{A} \in \mathcal{R}^{n \times n}$ is the adjacency matrix of the network, and $\boldsymbol{x} \in \mathcal{R}^n$ is an eigenvector solution of the equation, and λ is a scalar eigenvalue. Eigenvector centrality for each node is its value in the eigenvector x corresponding to the largest eigenvalue λ (of all possible solutions to the eigensystem).

(c) Clustering coefficient.[12] For unweighted graphs, the clustering coefficient of a node is the fraction of possible triangles through that node that exist. That is, if t_i is the number of closed triangles attached to a node i, and k_i is the number of nodes it is connected to, then the clustering coefficient is $\frac{2t_i}{k_i(k_i-1)}$.

We then train ML models on the extended tabular data set, using a large number of supervised learning models. GNNs are not used to train models in this

Figure 2. (Color online) Corporate Network with 1,723 Nodes and 29,126 Linked Node Pairs



Notes. The degree distribution is shown as well (lower plots) and displays the classic scale-free structure, that is, hub and spoke in nature, both in frequency (left) and log-log form (right). The mean degree across all nodes is 34. Using all nodes with degree > 1, we ran a regression of the log-log plot, which gives a power law coefficient of –1.22, $p = 0$, intercept = 6.8, $R^2 = 0.869$.

setting, and we may think of these tabular ML models as a baseline.

2. *Graph ML*: We use the original tabular data set as features combined with CorpNet to train a GNN for credit risk modeling. The goal here is to use the entire firm network with a graph machine learning model. This sets up a comparison with tabular ML models that use a few additional node-level graph metrics. In the finance literature, the use of GNNs is relatively new, whereas tabular ML using graph features has been used extensively, especially in the systemic risk literature (Billio et al. 2012, Poledna et al. 2015, Kok and Montagna 2016). The GNN model has many hyperparameters and we use hyperparameter optimization (HPO) to search over the number of hidden layers in the GCN, the number of neurons in the hidden layers,

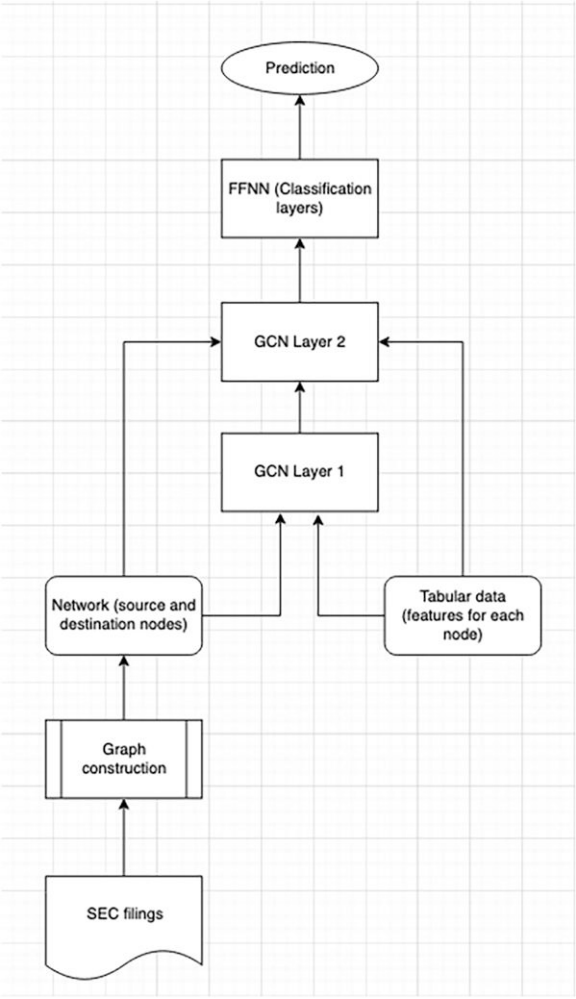
the extent of dropout applied, weight decay, epochs, aggregator (pooling) type, and learning rate. Details of the specific GNN used with equations are provided in Section 4.6.

We also ensembled the Tabular ML and Graph ML predictions by averaging the predicted probabilities from both models into a composite probability to make predictions. Results are reported for all Tabular ML models, the GNN model, and the ensembled model.

Overall, the Graph ML credit risk rating classifier involves two steps: (i) preparation of the corporate graph and (ii) combining the graph with tabular data for each firm using GCNs to train the GNN classifier. This is depicted in Figure 3.

In the next sections, we present the implementations of tabular and graph ML, that is, (i) AutoGluon for tabular machine learning, (ii) the GCN specification, (iii) discuss our use of inductive and transductive training, and (iv) describe node embeddings as feature representations used for graph ML.

Figure 3. Graph ML Diagram of the End-to-End Classifier



Notes. The SEC filings are used to create a network of related companies, stored as source and destination node lists. This graph information and tabular data are then processed through two GCN layers to get node embeddings, which are then used in a neural network to train a classifier.

4.3. AutoGluon for Implementing Tabular ML

We use AutoGluon, a highly performant open-source library for AutoML to fit our credit risk rating models on tabular data. This library "… enables easy-to-use and easy-to-extend AutoML with a focus on automated stack ensembling, deep learning, and real-world applications spanning image, text, and tabular data."[13] AutoGluon is facile to implement as well, so will enable other users to apply the ideas in this paper if needed.[14] AutoGluon is a powerful auto ML tool leveraging ensembling of several standard machine learning models.

AutoGluon is a good choice for this research for the following reasons. First, it is open source and hence available for free.[15] Second, it can be implemented on tabular data in just three lines of code.[16] Third, it is model agnostic because AutoGluon uses a wide variety of ML models (random forests, extreme randomized trees, k-nearest neighbors, LightGBM boosted trees, CatBoost trees, deep neural networks, etc., and also ensembles these models). Fourth, it is a hard benchmark to beat because AutoGluon uses automated deep learning, stack ensembling, and bagging to combine models, is highly performant, and has achieved a top 1% ranking on a wide collection of Kaggle competitions, as demonstrated in Erickson et al. (2020).[17] This sets a high benchmark for Graph ML to beat and also ensures a good tabular model that can be ensembled with the graph ML model.

The goal of the analysis aims to show that GNNs with no stack ensembling, or optimizations undertaken by an advanced AutoML algorithm such as AutoGluon, can achieve comparable, if not better results than these algorithms. This also assesses whether the graph construction methodology using large text from regulatory filings is successful in delivering state of the art machine learning performance through novel feature engineering. Later, in Section 5, we present the results from tabular and graph ML models.

4.4. GCN Specification for Implementing GNNs

We train GNNs in the models here. Because the data on SEC filings is public record, the network graph is known for all companies in the training and testing data sets. The GNN learns to predict the type of node (investment grade = 1 or below investment grade = 0) in the test data set from the tabular features in the training data and all graph relationships. This is a classic "node-prediction" problem, that is, if we know the type (0/1) of nodes in the training data set, then we learn how to label the remaining nodes (test data) in the graph. GNN training may be inductive or transductive, as discussed later in Section 4.5.

In a GNN, a node's neighborhood is used to define a computational graph and learn an embedding. In this approach, embeddings use both the structure of the graph and the features of the nodes. Embeddings are learned in an end-to-end fashion; thus, the predictions are a function of features of the target node and its ego network (the subgraph of the node and its neighbors to a chosen depth level) in contrast to shallow embedding methods that rely on the graph structure. Thus, GNNs can integrate feature information from topologically distant nodes in a nonlinear manner.

The implementation of our GNN uses graph convolution layers and hence is denoted a graph convolutional network or GCN. In our modeling, we used a GNN with two GCN layers. The GNN will ingest the graph and the features of each node and produce node embeddings from the trained graph. These node embeddings may then be used downstream for our classification task. We train the classifier end to end where we learn the weights on the GCN layers and the classification layers simultaneously. Once the GCN layers have been trained, they may be used for new nodes as long as they come with their SEC filings which are used, as described in Section 4.1, to determine which nodes in the network they will be linked to.

As an illustration, Figure 3 shows the overall classifier developed here. The SEC filings are fed into an algorithm that determines the CorpNet graph based on textual similarity in the MD&A sections of the filings. This is then combined with the tabular data in a GNN to generate node embeddings for each firm in the data. These embeddings are then used in a downstream neural network to train a classifier. The two GCN layers and neural net for classification are trained together.

To compare the GNN performance with other graph embedding generation techniques, we also implement node2vec. In node2vec, the embeddings are learned by training a neural network to predict the probability of a node given its context in a node sequence generated from a random walk on the graph. We use the deep graph library (DGL) to implement and train the node2vec model on the graph. Once the node embeddings are computed, we then take the produced node embeddings as static features that are fed into a downstream multilayer perceptron for the classification task. Thus, the node2vec approach consists of two distinct steps, one to train the embeddings and one to train the classifier, unlike the GNN which can be trained end to end.

4.5. Inductive vs. Transductive GNN Training

Our implementation of Graph ML may be transductive or inductive, and we implement both for completeness.

- *Transductive*. Training of the GNN uses the tabular features of all firms in both the train and test data sets, the complete graph, but only the labels from the training data set. The node embeddings of firms in the training data set are based on convolutions of tabular information from node neighbors that may include firms in the test data set as well. However, training does not use labels from the test data set. Even when GNNs are not used and only tabular data that contains graph node features (e.g., centrality) are used, the entire graph (train and test nodes) are needed to generate the graph-based features. To this extent, even papers that use graph features in tabular ML only, are in effect, implementing transductive training.
- *Inductive*. In this setting we train the GNN on the tabular data and subgraph of firms only in the training data set. Labels used are from the training data. We do not use the labels, tabular data, or graph information on firms in the test data set for model training.

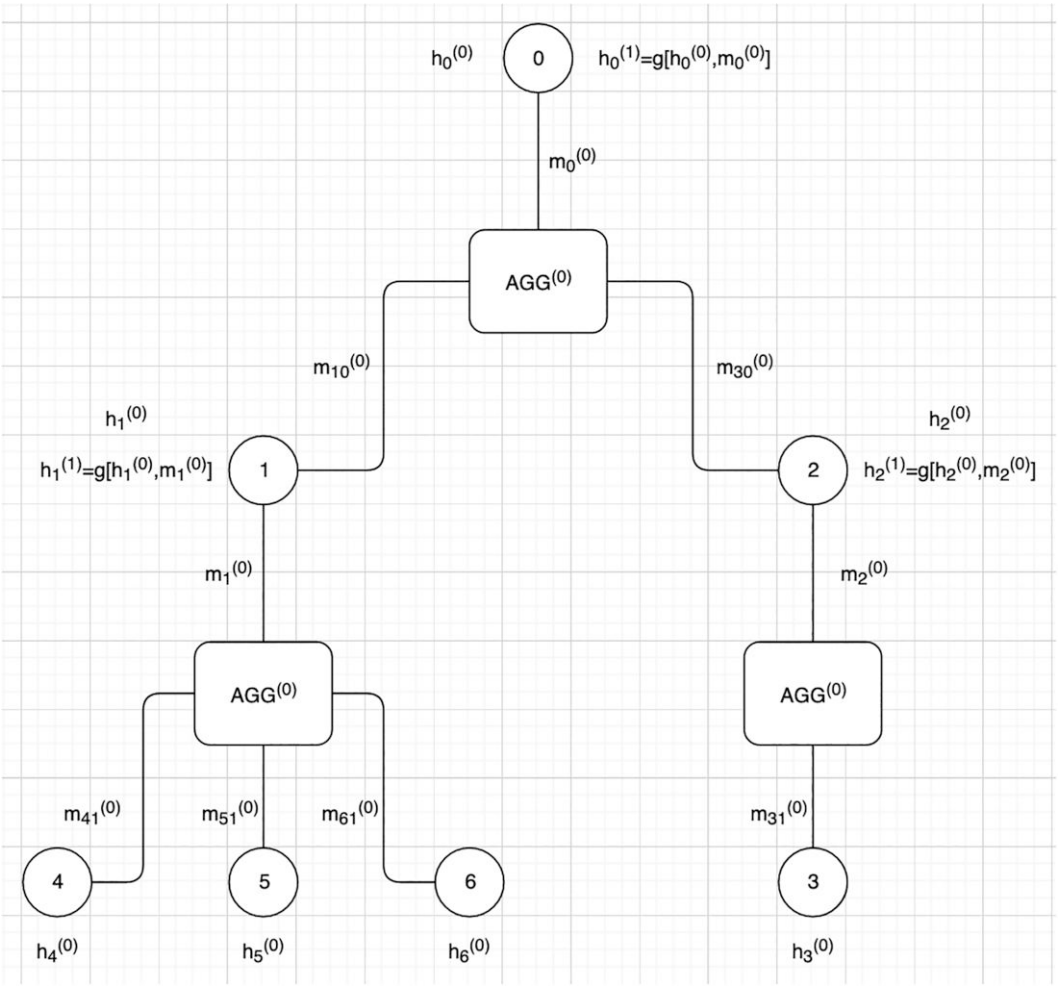
Whereas inductive models are more general, they eschew information available in the test data set (features only, not labels, which should never be used). Conversely, transductive models use feature information from the test data set and may be more accurate but may not generalize well. For a comparative discussion, see Ciano et al. (2022). It is to be noted that even when we use transductive training on the current train and test data sets, the model may still be used in an inductive manner for prediction on data that arrives subsequently, because the new firms may be added to the network and their tabular data will enable creating embeddings for the new nodes, with follow-on predictions.

4.6. Node Embeddings

In GNN training, node embeddings represent the information of a node and its neighbors, thereby comingling tabular and graph data. GNN models are fit on tabular features of the node itself and that of its direct neighbors and indirect neighbors to a chosen level of depth in the graph (the ego network). In the model implemented in this paper, the GNN layers convolve all the neighbor information (Kipf and Welling 2017)[18] and that of the node itself into an embedding for each node of size 32, which is a modeling choice. In the inductive setting, for example, GraphSAGE (Hamilton et al. 2017), only the neighbors in the training data set will be used to fit embedding, whereas in the transductive setting all nodes (without labels) will be used for convolution into a node's embedding.

Figure 4 shows an illustrative graph of seven companies. This is a subgraph for generating an embedding for node 0 using a GNN with two GCN layers, showing how data are processed for node 0 from connected nodes two layers deep. Nodes 1 and 2 feed into node 0, nodes 4, 5, and 6 feed into node 1, and node 3 feeds into node 2. The embeddings are denoted as $\boldsymbol{h}_i^{(r)}$, where i indexes nodes and (r) indexes the postlayer value. $\boldsymbol{h}_i^{(0)}$ denotes the initial data vector for each node, and $\boldsymbol{h}_i^{(1)}$ denotes the embedding after combining the initial data

Figure 4. Depiction of the GCN for Inductive Modeling



Notes. This shows how data are processed for node 0 from connected nodes two layers deep. Nodes 1 and 2 feed into node 0, nodes 4, 5, and 6 feed into node 1, and node 3 feeds into node 2. The embeddings are denoted as $\boldsymbol{h}_i^{(r)}$, where i indexes nodes and (r) indexes the postlayer value. $\boldsymbol{h}_i^{(0)}$ denotes the initial data vector for each node, and $\boldsymbol{h}_i^{(1)}$ denotes the embedding after convolving the initial data $\boldsymbol{h}_i^{(0)}$ with $\boldsymbol{m}_i^{(0)}$, the message from preceding nodes, using convolution $g[\cdot]$, the exact form of which is given by Equation (3). A message-passing framework is used, which is denoted as $\boldsymbol{m}_{ij}^{(r)}$, with i the "from" node and j is the "to" node. $AGG^{(r)}$ is a message aggregator (specific form shown in Equation (2)), and $\boldsymbol{m}_i^{(r)}$ is the combined message input from all preceding nodes into node i, which is used to update the original node embedding $\boldsymbol{h}_i^{(r)}$ to the new node embedding using network g, that is, $\boldsymbol{h}_i^{(r+1)} = g[\boldsymbol{h}_i^{(r)}, \boldsymbol{m}_i^{(r)}]$. Analogous structures may be used for GCN layers that are one layer deep or even three layers deep.

$\boldsymbol{h}_i^{(0)}$ with $\boldsymbol{m}_i^{(0)}$, the message from preceding nodes, using convolution $g[\cdot]$, the exact form of which is given by Equation (3). A message-passing framework is used, which is denoted as $\boldsymbol{m}_{ij}^{(r)}$, with i being the "from" node and j the "to" node. At the bottom end of the diagram are the initial embeddings, which may just be the feature vector at each node. $AGG^{(r)}$ is a message aggregator for these initial embeddings (specific form shown in Equation (2)). The aggregator weights its inputs and learns the weights during training. The output message $\boldsymbol{m}_i^{(r)}$ is the combined message input from all preceding nodes into node i, which is concatenated with the existing embedding of the node. This object is then used to update the original node embedding $\boldsymbol{h}_i^{(r)}$ to the new node embedding using network g, that is, $\boldsymbol{h}_i^{(r+1)} = g[\boldsymbol{h}_i^{(r)}, \boldsymbol{m}_i^{(r)}]$, and the weights of network $g[\cdot]$ are learned in training as well. Analogous structures may be used for GCN layers that are one layer deep or even three layers deep. The classifier is trained using the PyTorch[19] framework. The approach implemented here follows the GraphSAGE model of Hamilton et al. (2018).

More specifically, the GraphSAGE (SAGE = SAmple aggreGatE) approach of Hamilton et al. (2017) extends the original GCN of Kipf and Welling (2017) to inductive training in addition to transductive. Although the latter used semisupervised training, the former, newer approach allows for unsupervised, semisupervised, and supervised training. As shown in Figure 4, embeddings for a node i collect and integrate information from a set of neighbor nodes $N(i)$ to generate embeddings $\boldsymbol{h}_i^{(r)}$. This is done in two steps:

1. Collect and aggregate information from neighbor nodes into a representation. In GraphSAGE, this is called the aggregate (AGG) step, and Hamilton et al. (2017) offer three ways in which AGG may be implemented: (i) means (averaging neighbor embeddings), (ii) using LSTMs for combination, and (iii) max-pooling. Our analysis uses the last approach and the neighbor embeddings are combined as follows:

$$\boldsymbol{h}_{N(i)}^{(r)} = \max[(\sigma(\mathbf{W}_{\mathbf{pool}}\, \boldsymbol{h}_j^{(r-1)} + b),\ \forall j \in N(i))]. \quad (2)$$

In AGG, each neighbor node's embedding vector ($\boldsymbol{h}_j^{(r-1)}$) is fed through a fully connected neural net (weights $\mathbf{W}_{\mathbf{pool}}$, activation functions $\sigma(\cdot)$, usually ReLU, and bias term b) to get a representation, after which element-wise max-pooling is applied. AGG results in an intermediate representation.

2. Concatenate this representation with the representation from node i to create the node embedding, as follows:

$$\boldsymbol{h}_i^{(r)} = \sigma(\mathbf{W}^{(r)} \cdot CONCAT[\boldsymbol{h}_i^{(r-1)}, \boldsymbol{h}_{N(i)}^{(r)}]), \quad (3)$$

where the output of Equation (2) is applied, using weights $\mathbf{W}^{(r)}$ and activation function $\sigma(\cdot)$.[20]

In our specific application, Equation (3) is implemented using a neural network with two layers of 32 neurons each, and no dropout is applied (the learning rate is 0.02). Hyper-parameter optimization is undertaken to determine the number of epochs of training, with an initial number of 120, initial learning rate of 0.02, and weight decay parameter of 0.000294.

5. Analysis

In this section, we present the results of our credit rating models based on GNNs. We consider how the models perform with and without the additional graph information. As noted earlier, the experiments consider a binary classification problem, where the graph data and tabular data are used to fit a model that classifies companies into investment grade (label = 1) or below investment grade (label = 0). This is a classic decision made by asset managers who invest in credit markets. Such managers are interested in whether firms that are currently investment grade are predicted by the model to end up below investment grade and vice versa. Using the GNN methodology, we provide two sets of results: one with synthetic data and one with real data.

5.1. Results on the Synthetic Data Set

In addition to the AutoGluon benchmark, we fit the models using the `node2vec` approach, and the results are provided in all the tables. The results are shown in Table 1.

Several different tabular ML models are fit with target metric as the f1 score (this is standard, and results are

Table 1. Binary Classification Results on the Synthetic Data Set

Panel A

Model	F1	Accuracy	ROC_AUC	MCC	Mean recall	Precision	Recall
AutoGluon (WeightedEnsemble_L2)	0.731	0.682	0.749	0.357	0.672	0.676	0.798
	0.010	0.015	0.009	0.030	0.017	0.018	0.026
Node2Vec	0.790	0.774	0.852	0.544	0.772	0.7923	0.788
	0.003	0.003	0.001	0.006	0.003	0.004	0.003
GNN w/HPO (Transductive)	0.804	0.787	**0.864**	0.570	0.785	0.802	0.802
	0.003	0.010	0.001	0.001	0.002	0.003	0.009
GNN w/HPO (Inductive)	**0.816**	0.791	**0.874**	0.579	0.786	0.782	**0.852**
	0.008	0.010	0.010	0.021	0.011	0.012	0.009
Ensemble: GNN+Tabular	**0.813**	**0.796**	0.861	**0.589**	**0.794**	**0.808**	0.817
	0.004	0.009	0.002	0.007	0.003	0.003	0.011

Panel B

Model	F1	Accuracy	ROC_AUC	MCC	Mean recall	Precision	Recall
AutoGluon (WeightedEnsemble_L2)	0.755	0.732	0.808	0.459	0.729	0.749	0.762
	0.011	0.016	0.007	0.035	0.018	0.026	0.018
Node2Vec	0.782	0.765	0.834	0.523	0.764	0.787	0.777
	0.004	0.005	0.001	0.010	0.005	0.006	0.004
GNN w/HPO (Transductive)	**0.826**	**0.806**	**0.866**	**0.610**	**0.803**	**0.805**	**0.848**
	0.003	0.010	0.001	0.001	0.002	0.003	0.009
GNN w/HPO (Inductive)	0.807	0.778	0.863	0.555	0.772	0.765	**0.854**
	0.005	0.006	0.009	0.011	0.008	0.019	0.028
Ensemble: GNN+Tabular	0.817	0.796	0.862	0.588	0.792	0.794	0.843
	0.004	0.009	0.002	0.007	0.003	0.003	0.011

Notes. Panel A is run with only tabular features. Five replications are undertaken, and the mean and standard deviations are reported for metrics on the test data set (standard deviation values below the mean). The target objective is model f1 score. MCC is the Matthew's correlation coefficient (https://en.wikipedia.org/wiki/Phi_coefficient). Panel B uses the synthetic data set with tabular features and three graph node metrics. The better models are in bold font.

similar with accuracy or Area Under the Curve (AUC) as the target metric).[21] Table 1, Panel A, shows results where the tabular data has not been extended with the three tabular graph features, and the analysis with these features included is presented in Panel B. Detailed breakdowns of all models in Table 1 are shown in Tables A.2 and A.3. The results described below include all these tables. The main finding across all these tables is that the graph ML models are similar in their performance, and all of them are better than the tabular only models.

First, we discuss Table 1, Panel A (detailed in Table A.2). The data set has reasonable label balance, that is, 54% of the sample is investment grade firms and the remaining (46%) are below investment grade. The tabular models give f1 scores in the range of 67%–73%, accuracy levels in the range of 64%–68%, AUC values in the range of 68%–75%, and Matthew's Correlation Coefficient (MCC) values in the range of 27%–37%.

The graph neural network model is fit without ensembling and delivers better performance (f1 score = 80.4%, accuracy = 78.7%, AUC = 86.4%, MCC = 57%), showing an improvement from using the additional graph information over the tabular model. This demonstrates how the use of graph data can deliver better performance over tabular models, which is outside the margin of error, based on the standard deviations of the various metrics. The GNN is also ensembled with the best tabular models in a simple manner by averaging the predicted probabilities from the GNN and tabular models and then making a prediction. This slightly improves performance as well. It is interesting to note (see the appendix) that XGBoost is among the most popular models used in practice,[22] but its performance metrics are lower than many of the other models we used for the credit risk classification task here. Finally, the model also performs better than the node2vec based GNN (which is intuitive).

These models are trained in a transductive manner, where nodes (without labels) are used from the test data set. We also trained models in an inductive manner. Inductive models are trained on graphs that do not include nodes in the test data set. Hence, they may perform worse out of sample given they use less information. They may also perform better if the absence of test data set nodes results in lower noise and less overfitting. We see from Table 1, Panel A, that the inductive models show comparable, if not slightly better, metrics than when transductive training is used. This is not the case in Panel B, when the three additional graph features are included, where the inductive models perform worse than the transductive ones.

Second, we discuss Table 1, Panel B (detailed in Table A.3). We see that, with just three tabular features from the graph, we are able to increase the best f1 score of the tabular models to 76% from 73% without the three new graph-based tabular features (again, outside the margin of error, hence significant). The accuracy improves from 68% to 74%, AUC from 75% to 81%, and MCC from 37% to 48%. The GNN model does better than the improved tabular models (f1 = 83%, accuracy = 81%, AUC = 87%, and MCC = 61%). In this setting, the ensembling of best tabular models with GNN models gives somewhat impaired results because the poorer performance of AutoGluon relative to the GNN drags down the ensemble's quality. Finally, inductive training results are slightly attenuated, though comparable to those from transductive training. Overall, the results suggest that model improvements are possible with corporate networks generated from SEC filings coupled with the use of graph machine learning.

5.2. Results on the Real Data Set

The real data set is smaller than the synthetic one. The number of observations is 1,723 nodes and the graph has 29,126 links. The number of tabular features is much larger than that of the synthetic data and was discussed in Section 3.1. Ex ante, this makes it harder for additional information to generate improved models. The data comprise several financial ratios relating to liquidity, profitability, debt, operating performance, and cash flows. We also include the industry category in this data. Text from SEC filings (MD&A section of 10-K/Q forms) of these companies from the same quarter are used for the corporate network in the same way as for the synthetic data set.

We fit models in sequence in the same way as undertaken for the synthetic data set, that is, tabular models with and without node-level graph features, and additionally, GNN models, with and without ensembling. Table 2, Panel A, shows results where the tabular data has not been extended with the three tabular graph features, and the analysis with these features included is presented in Panel B. Table A.4 gives detailed results of all models run for Panel A, and Table A.5 presents detailed results of all models run for Panel B. The data set has low label imbalance, i.e., 57% of the sample is investment grade firms and the remaining (43%) are below investment grade. The results show that the ensembled tabular model does slightly better than the GNN. But now, in comparison with the synthetic data set, because both tabular and GNN models do well, ensembling both models provides superior performance to the best models without ensembling (the ensemble with the transductive GNN has better f1 score 89.5% versus 88%, accuracy of 87.5% versus 86%, AUC of 95% versus 93%, and MCC of 75% versus 72%). It is interesting to note that XGBoost's performance metrics are again slightly lower. In Table 2, even with the additional graph-based tabular features, performance of the ensemble and transductive GNN does not improve, in fact attenuates slightly. The pure inductive GNN model on real data performs only slightly below the transductive one, but does slightly better with the inclusion of graph features (Table 2, Panel B). Overall, we see a consistent

Table 2. Binary Classification Results on the Real Data Set

Panel A							
Model	F1	Accuracy	ROC_AUC	MCC	Mean recall	Precision	Recall
AutoGluon (WeightedEnsemble_L2)	0.880	0.859	0.933	0.711	0.850	0.851	0.911
	0.008	0.012	0.012	0.024	0.015	0.024	0.011
Node2Vec	0.813	0.779	0.843	0.547	0.769	0.784	0.844
	0.002	0.002	0.002	0.005	0.002	0.002	0.003
GNN w/HPO (Transductive)	0.861	0.835	0.904	0.662	0.824	0.823	0.903
	0.009	0.009	0.009	0.018	0.008	0.005	0.017
GNN w/HPO (Inductive)	0.837	0.811	0.887	0.614	0.804	0.821	0.854
	0.002	0.004	0.003	0.011	0.008	0.019	0.021
Ensemble: GNN+Tabular	**0.895**	**0.875**	**0.951**	**0.747**	**0.865**	**0.856**	**0.939**
	0.006	0.007	0.007	0.014	0.008	0.009	0.010
Panel B							
Model	F1	Accuracy	ROC_AUC	MCC	Mean recall	Precision	Recall
AutoGluon (WeightedEnsemble_L2)	0.871	0.850	0.927	0.693	0.843	0.851	0.892
	0.013	0.014	0.006	0.028	0.013	0.011	0.020
Node2Vec	0.810	0.774	0.844	0.536	0.762	0.774	0.851
	0.005	0.005	0.002	0.010	0.004	0.002	0.008
GNN w/HPO (Transductive)	0.850	0.820	0.885	0.632	0.809	0.810	0.893
	0.007	0.009	0.012	0.018	0.009	0.008	0.009
GNN w/HPO (inductive)	0.850	0.829	0.891	0.652	0.825	0.848	0.852
	0.014	0.015	0.010	0.030	0.014	0.010	0.023
Ensemble: GNN+Tabular	**0.888**	**0.867**	**0.942**	**0.729**	**0.857**	0.850	**0.929**
	0.006	0.007	0.011	0.015	0.008	0.007	0.008

Notes. Panel A is run with only tabular features. Five replications are undertaken, and the mean and standard deviations are reported for metrics on the test data set (standard deviation values below the mean). The target objective is model f1 score. MCC is the Matthew's correlation coefficient (https://en.wikipedia.org/wiki/Phi_coefficient). Panel B uses the real data set with tabular features and three graph node metrics. The better models are in bold font.

pattern where the bottom segment of the tables that includes graph ML models does better than the collection of tabular models in the upper portion of the tables, while also beating the `node2vec` benchmark easily.

5.3. Comparing Model Performance with Different Threshold Cutoffs

In the graphs used previously, a pair of firms would have a symmetric unweighted link in the graph if the cosine similarity between the two firms' MD&A embeddings exceeded a cutoff of 0.5. Using the synthetic data set, we see from Tables 3 and 4 that a graph based on a cutoff of 0.7 performs better than using graphs with other cutoffs. This suggests that a sparser graph leads to better models. This is intuitive: if the cutoff is very small, then all nodes will be linked and the graph would be mostly noninformative since all nodes would be linked to all other nodes and every node's neighborhood would be similar. Conversely, if the cutoff is very high, then it would lead to very few links in the network, and in the limit, there would be no graph at all. Hence, it is not surprising that when the cutoff is 0.9, the performance of the GCN declines from its best performing version when the cutoff is 0.7. The results here use a fixed hyperparameter set based on earlier optimization, and no HPO was undertaken; hence, results differ slightly from the earlier tables. These results suggest that for graph construction, using different cutoffs does make a difference to the GNN classifier, since the graph structure changes.

The results in this section also show that we can improve on the results in the preceding Tables 1 and 2, where the cutoff was chosen a priori to be 0.5. We chose

Table 3. Performance of GCN on Synthetic Data with Different Similarity Threshold Cutoffs 0.1, 0.3, 0.5, 0.7, and 0.9

	0.1	0.3	0.5	0.7	0.9
F1	0.733 ± 0.004	0.774 ± 0.006	0.813 ± 0.000	**0.842** ± 0.002	0.809 ± 0.005
Accuracy	0.697 ± 0.005	0.739 ± 0.005	0.783 ± 0.005	**0.835** ± 0.004	0.796 ± 0.002
MCC	0.386 ± 0.020	0.474 ± 0.009	0.565 ± 0.010	**0.672** ± 0.007	0.592 ± 0.007
Mean recall	0.690 ± 0.010	0.732 ± 0.004	0.776 ± 0.008	**0.837** ± 0.006	0.796 ± 0.002
Precision	0.699 ± 0.010	0.730 ± 0.001	0.763 ± 0.021	**0.876** ± 0.020	0.822 ± 0.024
Recall	0.770 ± 0.006	0.823 ± 0.011	**0.869** ± 0.023	0.810 ± 0.024	0.799 ± 0.032

Note. Results here use a fixed hyperparameter set based on earlier optimization, and no HPO was undertaken for the results in this table.

Table 4. Performance of GCN on Real Data with Different Similarity Threshold Cutoffs of 0.1, 0.3, 0.5, 0.7, and 0.9

	0.1	0.3	0.5	0.7	0.9
F1	0.827 ± 0.001	0.853 ± 0.002	0.841 ± 0.002	**0.866** ± 0.001	0.839 ± 0.011
Accuracy	0.797 ± 0.003	0.836 ± 0.004	0.813 ± 0.004	**0.845** ± 0.001	0.814 ± 0.010
MCC	0.585 ± 0.007	0.669 ± 0.011	0.616 ± 0.008	**0.683** ± 0.003	0.621 ± 0.020
Mean recall	0.788 ± 0.007	0.836 ± 0.006	0.802 ± 0.006	**0.839** ± 0.001	0.808 ± 0.008
Precision	0.804 ± 0.016	**0.868** ± 0.013	0.809 ± 0.009	0.848 ± 0.003	0.825 ± 0.002
Recall	0.852 ± 0.020	0.839 ± 0.007	0.878 ± 0.008	**0.885** ± 0.003	0.855 ± 0.022

Note. Results here use a fixed hyperparameter set based on earlier optimization, and no HPO was undertaken for the results in this table.

to use 0.5 before assessing different cutoff parameters, so as to be conservative and not hyperparameter optimize graph construction. After searching over cutoffs, we see a further improvement of around 2%–3% from tuning the cutoff parameter to a value of 0.7, for both data sets, as shown in Tables 3 and 4.

6. Concluding Discussion

This paper introduces a simple methodology for using graph information to extend current approaches that use tabular data for credit scoring. One contribution is a new approach for constructing corporate graphs using text from SEC filings, that is, a methodology to construct multimodal synthetic financial and graph data for companies and use these data to demonstrate how graph machine learning may be applied to improve credit scoring. After illustrating the methodology on synthetic data, we implement the approach on real data and report several model metrics when the corporate graph is blended with standard tabular financial data.

Whereas the application demonstrated here relates to building graph machine learning models for credit risk scoring, the same approach may be used for other use cases, such as modeling systemic risk using banking networks (Das et al. 2021), and for stock trading exploiting cross-correlation information among stocks by modeling their interconnectedness. Fraud detection is also an important and widespread practical use of such models.[23] A solution to productionize this is hosted on SageMaker JumpStart,[24] and related documentation[25] is also available.[26] Churn prediction is yet another application that may benefit as an extension to the work here and entity resolution in identity graphs.[27]

There are limitations and extensions of this work that are important to consider. The work on corporate graph construction has not settled on a generally accepted best graph. We provide one approach and show how it can improve credit risk modeling. However, this work may be extended to other approaches for graph construction depending on the availability of alternate data. One possible approach is to construct a graph of companies based on their supply chain relationships, if such data are available, as these relationships would also be good indicators of spillovers of credit risk. Other approaches are to use stock returns, as in Billio et al. (2012) and Das et al. (2019). Different approaches may offer varied results, depending on the use case in question. Another possible extension of the graph building approach in the paper is that using additional methods would enable extension of the CorpNet graph in this paper to be bipartite, such as in the work of Óskarsdóttir and Bravo (2021) who define a centrality score based on a multilayer network. Although the implementation in this paper used homogeneous graphs, the approach may also be applied to heterogeneous graphs.

The model may be extended to automatically eliciting graph explanations (Bracke et al. 2019, Ying et al. 2019), which is especially useful in the lending setting. Broadening this work to multiclass problems is an extension, generalizing the binary one in this paper. We do not have sufficient data to implement models on high label granularity. Therefore, we implemented a basic financial application, that is, predicting investment grade versus noninvestment grade ratings. Finally, using the model for link prediction, with a view to predicting dynamic changes in the graph, may also be an interesting extension that policy makers might be able to exploit in understanding how credit quality may change over time.

Appendix. Synthetic Data Set Generation

In Section 3.2, we provided the overview of four steps to generate the synthetic data set that corresponds to the real data set. This appendix provides more detail on the algorithm used to generate synthetic data.

A.1. Numerical Text Scores for Synthetic Data Generation

For synthetic data generation, to ensure that the ratings are correlated with the text, we undertake additional feature engineering by computing the "NLP scores" from the text. For each MD&A section in the SEC filings, we generate 11 scores for positivity, negativity, litigiousness, polarity, risk, readability, fraud, safety, certainty, uncertainty, and sentiment. Readability is based on the Gunning-Fog index, see Gunning (1952).[28] Sentiment is based on VADER (Valence Aware Dictionary and sEntiment Reasoner), a lexicon and rule-based sentiment analysis tool (Hutto and Gilbert 2014). The other scores are based on the percentage of words matched to word lists that were curated in a semiautomated manner.[29] The data frame is thus extended with 11 additional numeric columns.

We use five of these scores as positive signals (positivity, polarity, safe, sentiment, certainty) and another five as negative signals (fraud, litigious, negative, uncertainty, risk); we dropped the readability score as it has a different scale than the other scores. We obtain a net score by adding the positive signals and subtracting the negative signals. The net score is used to rank firms by credit quality. Synthetic credit ratings are then assigned to the ranked firms in proportions that are consistent with frequencies among rated companies, that is, approximately AAA (3% of companies), AA (17%), A (30%), BBB (24%), BB (13%), B (12%), and CCC and below (1%).

A.2. Simulating Company Financials

The model we use is a well-known bankruptcy prediction approach from the seminal paper by Altman (1968).[30] A brief summary is provided here.

The model uses eight inputs converted into five financial ratios: (i) current assets (*CA*); (ii) current liabilities (*CL*); (iii) total liabilities (*TL*); (iv) *EBIT* (earnings before interest and taxes); (v) total assets (*TA*); (vi) net sales (*NS*); (vii) retained earnings (*RE*); and (viii) market value of equity (*MVE*). There are strict relationships between these eight items, and we model them as stipulated in the Altman model through the following five ratios:

- (A) EBIT/total assets
- (B) Net sales/total assets
- (C) Market value of equity/total liabilities
- (D) Working capital/total assets
- (E) Retained earnings/total assets

The Z-score is then provided by the following formula, estimated using linear discriminant analysis (LDA), and the coefficients in the formula are widely published by Altman.

$$Zscore = 3.3A + 0.99B + 0.6C + 1.2D + 1.4E \qquad \text{(A.1)}$$

High-quality companies have high Z-scores, and low-quality companies have low ones. The usual ranges are as follows (different applications vary these ranges as needed; hence, this set of ranges is not the only one used in practice):

1. $Z > 3.0$: safe
2. $2.7 < Z < 2.99$: caution
3. $1.8 < Z < 2.7$: bankruptcy possible in two years
4. $Z < 1.8$: high chance of bankruptcy

These ranges may be stipulated as desired depending on the use case.

We calculate Z-scores from simulated financials calibrated to broad balance sheet, income statement, and market data using averages for the U.S. economy from the following sources:

- Balance sheet data: https://fred.stlouisfed.org/release/tables?rid=434&eid=196197
- Income statement data: https://fred.stlouisfed.org/release/tables?rid=434&eid=195208
- Price to book: http://pages.stern.nyu.edu/~adamodar/New_Home_Page/datafile/pbvdata.html

The numbers taken from these sources are used to extract values for U.S. companies for the eight inputs mentioned previously. In addition, we obtain the average price-to-book (*P2B*) ratio for U.S. companies, book equity value (*EQ*), and then generate market value of equity (*MVE*) as $P2B(EQ + RE)$. Working capital (*WC*) is as usual, $CA - CL$, the difference between current assets and current liabilities. We also normalized the data by total assets to obtain the averages for U.S. companies, shown in Table A.1.

These baseline values are used to simulate synthetic firms' values for each of these variables. Various checks are noted in Table A.1 to make sure the simulated balance sheets and income statement values are consistent with each other. Based on these random values, a simulated Z-score is generated

Table A.1. Normalized Average Financial Values for U.S. Companies Based on Data for 2021 Q1

Item	Percentage total assets	Simulation equation
Total assets (*TA*)	100.00	100
Current liabilities (*CL*)	19.37	$\tilde{CL} = CL(1 + x(u - 0.5))$
Current and L/T liabilities (*TL*)	58.67	$\tilde{TL} = TL(1 + x(u - 0.5))$
Equity (*EQ*)	12.75	$\tilde{EQ} = EQ(1 + x(u - 0.5))$
Retained earnings (*RE*)	28.58	$\tilde{RE} = RE(1 + x(u - 0.5))$
Current assets (*CA*)	26.54	$\tilde{CA} = CA(1 + x(u - 0.5))$
Net sales (*NS*)	13.99	$\tilde{NS} = NS(1 + x(u - 0.5))$
EBIT	1.32	$\tilde{EBIT} = EBIT(1 + x(u - 0.5))$
Price-to-book ratio (*P2B*)	4.75	$\tilde{P2B} = P2B(1 + x(u - 0.5))$
Market value of equity (*MVE*)	196.31	$\tilde{MVE} = \tilde{P2B}(\tilde{EQ} + \tilde{RE})$
Working capital (*WC*)	7.16	$\tilde{WV} = \tilde{CA} - \tilde{CL}$
Average Z-score (*Z*)	2.68	$\tilde{Z}$

Notes. These values are described in Section 3.2 and are used as a baseline to generate synthetic company data using a level of noise, x, combined with a random uniform variate u. We used $x = 0.3$ for the data generated in the paper. The following checks are carried out to make sure the simulated balance sheets and income statement values are consistent with each other: (1) $TL > CL$, (2) $TA \geq EQ + RE + TL$, (3) $TA > CA$, (4) $NS > EBIT$, (5) $P2B > 0$. If any of these checks fails, the simulated firm is discarded and not added to the data set.

Table A.2. Binary Classification Results on the Synthetic Data Set with Only Tabular Features

Model	F1	Accuracy	ROC_AUC	MCC	Mean recall	Precision	Recall
LightGBMXT	0.738	0.657	0.747	0.323	0.636	0.634	**0.889**
	0.008	0.032	0.012	0.052	0.039	0.039	0.056
NeuralNetFastAI	0.711	0.677	0.743	0.347	0.671	0.69	0.735
	0.011	0.012	0.013	0.025	0.014	0.021	0.037
WeightedEnsemble_L2	0.731	0.682	0.749	0.357	0.672	0.676	0.798
	0.010	0.015	0.009	0.030	0.017	0.018	0.026
NeuralNetTorch	0.733	0.670	0.753	0.342	0.655	0.659	0.836
	0.009	0.030	0.011	0.047	0.04	0.046	0.086
LightGBM	0.725	0.658	0.723	0.314	0.642	0.645	0.835
	0.006	0.017	0.010	0.028	0.025	0.032	0.066
LightGBMLarge	0.703	0.660	0.714	0.314	0.653	0.669	0.745
	0.018	0.015	0.020	0.029	0.019	0.026	0.067
ExtraTreesGini	0.714	0.679	0.740	0.351	0.674	0.691	0.739
	0.007	0.008	0.007	0.018	0.010	0.013	0.017
RandomForestGini	0.709	0.678	0.736	0.349	0.674	0.694	0.726
	0.011	0.008	0.009	0.015	0.007	0.009	0.027
CatBoost	0.721	0.691	0.750	0.376	0.687	0.707	0.735
	0.010	0.011	0.008	0.023	0.011	0.011	0.014
ExtraTreesEntr	0.716	0.683	0.740	0.359	0.678	0.696	0.738
	0.009	0.012	0.010	0.025	0.013	0.015	0.014
RandomForestEntr	0.711	0.679	0.736	0.351	0.675	0.694	0.729
	0.008	0.010	0.008	0.021	0.011	0.013	0.011
XGBoost	0.698	0.665	0.721	0.324	0.661	0.683	0.715
	0.013	0.011	0.018	0.021	0.010	0.009	0.024
KNeighborsUnif	0.675	0.643	0.684	0.279	0.639	0.665	0.686
	0.013	0.011	0.012	0.022	0.011	0.009	0.022
KNeighborsDist	0.674	0.641	0.684	0.275	0.637	0.664	0.684
	0.012	0.009	0.010	0.017	0.008	0.007	0.023
Node2Vec	0.790	0.774	0.852	0.544	0.772	0.7923	0.788
	0.003	0.003	0.001	0.006	0.003	0.004	0.003
GNN w/HPO (Transductive)	0.804	0.787	**0.864**	0.570	0.785	0.802	0.802
	0.003	0.010	0.001	0.001	0.002	0.003	0.009
GNN w/HPO (Inductive)	**0.816**	0.791	**0.874**	0.579	0.786	0.782	**0.852**
	0.008	0.010	0.010	0.021	0.011	0.012	0.009
Ensemble: GNN+Tabular	**0.813**	**0.796**	0.861	**0.589**	**0.794**	**0.808**	0.817
	0.004	0.009	0.002	0.007	0.003	0.003	0.011

Notes. Five replications are undertaken, and the mean and standard deviations are reported for metrics on the test data set (standard deviation values below the mean). The target objective is model f1 score. MCC is the Matthew's correlation coefficient (https://en.wikipedia.org/wiki/Phi_coefficient).

using Equation (A.1), after computing ratios A, B, C, D, and E of the Altman model. The firms are then ranked on Z-score and the simulated financials are concatenated to the text from Step 1. The final synthetic data set comprises more than 3,000 firms.

A.3. Numeric Results

The details of numeric results for Table 1 and 2 are shown in Tables A.2–A.5, respectively.

A.4. Note on Data and Code

We make this application available in various ways.

A.4.1. Solution in SageMaker JumpStart. The results in this paper on the synthetic data set have been deployed as an application in Amazon SageMaker JumpStart (https://docs.aws.amazon.com/sagemaker/latest/dg/studio-jumpstart.html). To use the code with the data, you can access it through the AWS Console, and the instructions are provided on the JumpStart page mentioned before.

After navigating to SageMaker JumpStart, search for the solution card named "Graph-Based Credit Scoring." The card appears as follows:

Click on the card to open the solution and on the landing page click on the orange button to "Deploy" the solution. Once deployed, the solution notebook will contain the entire workflow to run the model with the provided data set.

Table A.3. Binary Classification Results on the Synthetic Data Set with Tabular Features and Three Graph Node Metrics

Model	F1	Accuracy	ROC_AUC	MCC	Mean recall	Precision	Recall
NeuralNetTorch	0.755	0.730	0.803	0.455	0.727	0.743	0.767
	0.012	0.012	0.007	0.024	0.012	0.017	0.028
WeightedEnsemble_L2	0.755	0.732	0.808	0.459	0.729	0.749	0.762
	0.011	0.016	0.007	0.035	0.018	0.026	0.018
CatBoost	0.762	0.741	0.814	0.478	0.739	0.760	0.764
	0.006	0.008	0.006	0.018	0.009	0.014	0.011
ExtraTreesEntr	0.757	0.732	0.805	0.460	0.729	0.746	0.767
	0.007	0.007	0.006	0.015	0.008	0.013	0.019
ExtraTreesGini	0.755	0.731	0.803	0.457	0.728	0.745	0.766
	0.008	0.006	0.006	0.012	0.006	0.005	0.016
LightGBMLarge	0.743	0.719	0.792	0.433	0.716	0.736	0.752
	0.009	0.014	0.012	0.029	0.015	0.021	0.016
LightGBMXT	0.750	0.721	0.796	0.438	0.717	0.732	0.770
	0.010	0.017	0.016	0.036	0.020	0.031	0.030
XGBoost	0.743	0.722	0.801	0.440	0.720	0.745	0.742
	0.012	0.017	0.010	0.035	0.018	0.025	0.017
RandomForestGini	0.751	0.728	0.801	0.453	0.726	0.747	0.754
	0.010	0.010	0.008	0.020	0.010	0.014	0.023
LightGBM	0.747	0.726	0.803	0.448	0.724	0.749	0.746
	0.014	0.018	0.013	0.037	0.019	0.025	0.025
RandomForestEntr	0.748	0.725	0.803	0.446	0.722	0.743	0.754
	0.011	0.012	0.010	0.025	0.012	0.015	0.019
NeuralNetFastAI	0.748	0.728	0.792	0.454	0.727	0.753	0.743
	0.016	0.019	0.015	0.038	0.020	0.023	0.020
KNeighborsUnif	0.676	0.642	0.682	0.277	0.638	0.664	0.688
	0.016	0.014	0.013	0.027	0.013	0.010	0.025
KNeighborsDist	0.673	0.640	0.683	0.273	0.636	0.662	0.685
	0.014	0.011	0.01	0.022	0.010	0.008	0.026
Node2Vec	0.782	0.765	0.834	0.523	0.764	0.787	0.777
	0.004	0.005	0.001	0.010	0.005	0.006	0.004
GNN w/HPO (Transductive)	**0.826**	**0.806**	**0.866**	**0.610**	**0.803**	**0.805**	**0.848**
	0.003	0.010	0.001	0.001	0.002	0.003	0.009
GNN w/HPO (Inductive)	0.807	0.778	0.863	0.555	0.772	0.765	**0.854**
	0.005	0.006	0.009	0.011	0.008	0.019	0.028
Ensemble: GNN+Tabular	0.817	0.796	0.862	0.588	0.792	0.794	0.843
	0.004	0.009	0.002	0.007	0.003	0.003	0.011

Notes. Five replications are undertaken, and the mean and standard deviations are reported for metrics on the test data set (standard deviation values below the mean). The target objective is model f1 score. MCC is the Matthew's correlation coefficient (https://en.wikipedia.org/wiki/Phi_coefficient).

We hope the hosted model will enable other researchers to modify it for other datasets and use cases for graph machine learning.

A.4.2. GitHub. We also offer example code and synthetic data in the following repository: https://github.com/awslabs/sagemaker-graph-based-credit-scoring.

A.4.3. CodeOcean Capsule. The example code is also presented in a CodeOcean capsule: https://codeocean.com/capsule/5230264/tree/v2. This includes the code for generating the synthetic data set, and a notebook for running the tabular models and both, transductive and inductive GNNs. In the capsule title "GNN Paper," see the subfolder "sagemaker-graph-based-credit-scoring" in the "code" folder. The notebook to run is "notebook.ipynb."

Table A.4. Binary Classification Results on the Real Data Set with Only Tabular Features

Model	F1	Accuracy	ROC_AUC	MCC	Mean recall	Precision	Recall
ExtraTreesEntr	0.871	0.846	0.929	0.686	0.835	0.830	0.917
	0.012	0.015	0.014	0.031	0.017	0.020	0.019
LightGBMXT	0.882	0.863	0.929	0.719	0.856	0.861	0.904
	0.006	0.009	0.009	0.018	0.011	0.018	0.013
WeightedEnsemble_L2	0.880	0.859	0.933	0.711	0.850	0.851	0.911
	0.008	0.012	0.012	0.024	0.015	0.024	0.011

Table A.4. (Continued)

Model	F1	Accuracy	ROC_AUC	MCC	Mean recall	Precision	Recall
CatBoost	0.864	0.841	0.924	0.675	0.834	0.841	0.889
	0.008	0.010	0.010	0.020	0.011	0.016	0.018
ExtraTreesGini	0.877	0.854	0.928	0.703	0.845	0.843	0.915
	0.015	0.019	0.014	0.039	0.022	0.026	0.013
XGBoost	0.858	0.834	0.915	0.661	0.827	0.838	0.880
	0.007	0.010	0.006	0.022	0.013	0.023	0.021
LightGBM	0.864	0.841	0.920	0.675	0.834	0.842	0.887
	0.005	0.006	0.009	0.012	0.006	0.009	0.013
LightGBMLarge	0.853	0.828	0.912	0.649	0.821	0.832	0.877
	0.010	0.014	0.008	0.028	0.016	0.023	0.021
RandomForestGini	0.850	0.823	0.905	0.637	0.813	0.820	0.882
	0.010	0.012	0.011	0.025	0.013	0.018	0.024
RandomForestEntr	0.847	0.819	0.907	0.629	0.808	0.813	0.885
	0.008	0.009	0.010	0.019	0.010	0.017	0.025
KNeighborsDist	0.768	0.715	0.784	0.413	0.698	0.716	0.828
	0.021	0.026	0.019	0.055	0.026	0.020	0.024
KNeighborsUnif	0.758	0.705	0.759	0.391	0.688	0.709	0.815
	0.020	0.022	0.019	0.046	0.021	0.014	0.028
Node2Vec	0.813	0.779	0.843	0.547	0.769	0.784	0.844
	0.002	0.002	0.002	0.005	0.002	0.002	0.003
GNN w/HPO (Transductive)	0.861	0.835	0.904	0.662	0.824	0.823	0.903
	0.009	0.009	0.009	0.018	0.008	0.005	0.017
GNN w/HPO (Inductive)	0.837	0.811	0.887	0.614	0.804	0.821	0.854
	0.002	0.004	0.003	0.011	0.008	0.019	0.021
Ensemble: GNN+Tabular	**0.895**	**0.875**	**0.951**	**0.747**	**0.865**	**0.856**	**0.939**
	0.006	0.007	0.007	0.014	0.008	0.009	0.010

Notes. Five replications are undertaken, and the mean and standard deviations are reported for metrics on the test data set (standard deviation values below the mean). The target objective is model f1 score. MCC is the Matthew's correlation coefficient (https://en.wikipedia.org/wiki/Phi_coefficient).

Table A.5. Binary Classification Results on the Real Data Set with Tabular Features and Three Graph Node Metrics

Model	F1	Accuracy	ROC_AUC	MCC	Mean recall	Precision	Recall
LightGBMXT	0.878	0.857	0.932	0.708	0.850	**0.855**	0.902
	0.006	0.009	0.009	0.018	0.011	0.016	0.007
WeightedEnsemble_L2	0.871	0.850	0.927	0.693	0.843	0.851	0.892
	0.013	0.014	0.006	0.028	0.013	0.011	0.020
ExtraTreesGini	0.872	0.848	0.928	0.69	0.837	0.833	0.915
	0.016	0.018	0.010	0.038	0.019	0.002	0.032
ExtraTreesEntr	0.873	0.849	0.927	0.693	0.839	0.838	0.912
	0.009	0.011	0.012	0.023	0.012	0.017	0.022
XGBoost	0.863	0.839	0.918	0.671	0.831	0.836	0.891
	0.010	0.010	0.009	0.019	0.009	0.011	0.024
CatBoost	0.868	0.845	0.924	0.684	0.837	0.841	0.898
	0.009	0.011	0.010	0.023	0.011	0.012	0.015
LightGBM	0.860	0.838	0.922	0.669	0.832	0.845	0.876
	0.006	0.006	0.012	0.013	0.005	0.003	0.014
RandomForestEntr	0.847	0.819	0.909	0.629	0.808	0.812	0.886
	0.011	0.014	0.012	0.029	0.015	0.018	0.023
RandomForestGini	0.846	0.818	0.904	0.628	0.808	0.814	0.882
	0.005	0.005	0.009	0.009	0.006	0.012	0.020
LightGBMLarge	0.845	0.819	0.906	0.629	0.81	0.821	0.871
	0.010	0.011	0.014	0.022	0.012	0.017	0.025
KNeighborsDist	0.767	0.715	0.784	0.411	0.697	0.715	0.829
	0.021	0.026	0.019	0.055	0.026	0.02	0.026
KNeighborsUnif	0.758	0.704	0.759	0.389	0.687	0.708	0.816
	0.021	0.023	0.020	0.049	0.022	0.015	0.030
Node2Vec	0.810	0.774	0.844	0.536	0.762	0.774	0.851
	0.005	0.005	0.002	0.010	0.004	0.002	0.008
GNN w/HPO (Transductive)	0.850	0.820	0.885	0.632	0.809	0.810	0.893
	0.007	0.009	0.012	0.018	0.009	0.008	0.009

Table A.5. (Continued)

Model	F1	Accuracy	ROC_AUC	MCC	Mean recall	Precision	Recall
GNN w/HPO (inductive)	0.850	0.829	0.891	0.652	0.825	0.848	0.852
	0.014	0.015	0.010	0.030	0.014	0.010	0.023
Ensemble: GNN+Tabular	**0.888**	**0.867**	**0.942**	**0.729**	**0.857**	0.850	**0.929**
	0.006	0.007	0.011	0.015	0.008	0.007	0.008

Notes. Five replications are undertaken, and the mean and standard deviations are reported for metrics on the test data set (standard deviation values below the mean). The target objective is model f1 score. MCC is the Matthew's correlation coefficient (https://en.wikipedia.org/wiki/Phi_coefficient).

Endnotes

[1] See https://www.wallstreetmojo.com/altman-z-score/.

[2] An example of such an architecture is node2vec (https://snap.stanford.edu/node2vec/), which may be used to construct quality node embeddings using semisupervised learning (Grover and Leskovec 2016). Specialized neural networks using graph convolutional network (GCN) layers with ReLU activation functions may be applied to integrate network relationships with node features in a nonlinear manner (Kipf and Welling 2017).

[3] See https://www.kaggle.com/agewerc/corporate-credit-rating. The Github repo is https://github.com/Agewerc/ML-Finance. Credit ratings data in this repo is originally sourced from https://public.opendatasoft.com/ and financial information from https://financialmodelingprep.com/.

[4] See https://docs.aws.amazon.com/sagemaker/latest/dg/studio-jumpstart.html for documentation and an overview. See also the multimodal financial tools in JumpStart: https://aws.amazon.com/about-aws/whats-new/2021/09/amazon-sagemaker-jumpstart-multimodal-financial-analysis-tools/. The retrieval documentation is here: https://sagemaker-jumpstart-industry-pack.readthedocs.io/en/latest/notebooks/index.html.

[5] See https://www.investopedia.com/terms/a/altman.asp.

[6] See https://docs.aws.amazon.com/sagemaker/latest/dg/studio-jumpstart-industry.html.

[7] Based on doc2vec, https://radimrehurek.com/gensim/models/doc2vec.html, which uses word2vec, see https://www.tensorflow.org/tutorials/text/word2vec.

[8] See https://docs.aws.amazon.com/sagemaker/latest/dg/studio-jumpstart-industry.html\#studio-jumpstart-industry-models.

[9] See https://www.investopedia.com/terms/s/systemically-important-financial-institution-sifi.asp.

[10] See https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.degree_centrality.html.

[11] See https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.eigenvector_centrality.html\#networkx.algorithms.centrality.eigenvector_centrality.

[12] See https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cluster.clustering.html\#networkx.algorithms.cluster.clustering.

[13] See https://auto.gluon.ai/stable/index.html.

[14] See https://auto.gluon.ai/stable/tutorials/tabular_prediction/tabular-quickstart.html.

[15] See https://github.com/autogluon/autogluon.

[16] See https://venturebeat.com/ai/amazons-autogluon-produces-ai-models-with-as-little-as-three-lines-of-code/.

[17] See https://aws.amazon.com/blogs/opensource/machine-learning-with-autogluon-an-open-source-automl-library/.

[18] See https://tkipf.github.io/graph-convolutional-networks/.

[19] See https://pytorch.org.

[20] GraphSAGE extended the original GCN of Kipf and Welling (2017), where information is combined from neighboring nodes using the same layer-wise propagation approach in one step:

$$\mathbf{H}^{(r)} = \sigma([\mathbf{D}^{-1/2} \cdot \mathbf{A} \cdot \mathbf{D}^{-1/2}] \cdot \mathbf{H}^{(r-1)} \cdot \mathbf{W}^{(r)}),$$

where $\mathbf{H} \in \mathcal{R}^{n \times d}$ is the embedding matrix, and $\mathbf{D} \in \mathcal{R}^{n \times n}$ is the node degree diagonal matrix corresponding to the adjacency matrix $\mathbf{A} \in \mathcal{R}^{n \times n}$. The term $[\mathbf{D}^{-1/2} \cdot \mathbf{A} \cdot \mathbf{D}^{-1/2}] \in \mathcal{R}^{n \times n}$ is a normalization for the embeddings generated by the term $\mathbf{H}^{(r-1)} \cdot \mathbf{W}^{(r)}$. Here, $\mathbf{W}^{(r)} \in \mathcal{R}^{d \times d}$ is the trainable weights matrix. The zero elements in $\mathbf{A}$ automatically remove convolutions where the nodes are not neighbors. We may write the aggregation and convolution node-wise as follows:

$$h_i^{(r)} = \sigma\left(\sum_j \frac{1}{c_{ij}} h_j^{(r-1)} \cdot \mathbf{W}^{(r)}\right),$$

where c_{ij} is an element in $[\mathbf{D}^{-1/2} \cdot \mathbf{A} \cdot \mathbf{D}^{-1/2}]$.

[21] A reference taxonomy of models is provided at https://auto.gluon.ai/stable/api/autogluon.tabular.models.html, along with usage parameters.

[22] See https://www.kaggle.com/general/196541.

[23] For a cloud-based application of fraud detection, see https://aws.amazon.com/blogs/machine-learning/detect-fraudulent-transactions-using-machine-learning-with-amazon-sagemaker/, which shows how to detect fraudulent transactions using machine learning.

[24] See https://aws.amazon.com/sagemaker/jumpstart/.

[25] See https://docs.aws.amazon.com/sagemaker/latest/dg/jumpstart-solutions.html\#jumpstart-solutions-fraud-detection.

[26] See also the related Github repository: https://github.com/awslabs/sagemaker-graph-fraud-detection.

[27] See https://github.com/awslabs/sagemaker-graph-entity-resolution.

[28] This is documented here: https://sagemaker-jumpstart-industry-pack.readthedocs.io/en/latest/smjsindustry.nlp_scorer.html.

[29] See also other sources of such word lists such as the Loughran and McDonald lists: https://sraf.nd.edu/textual-analysis/resources/.

[30] See also the description at this site (and is also available on many other sites): https://www.creditguru.com/index.php/bankruptcy-and-insolvency/altman-z-score-insolvency-predictor.

References

Abbass P, Brownlees C, Hans C, Podlich N (2016) Credit risk interconnectedness: What does the market really know? *J. Financial Stability* 29:1–12.

Ahmed A, Shervashidze N, Narayanamurthy S, Josifovski V, Smola AJ (2013) Distributed large-scale natural graph factorization. *Proc. 22nd Internat. Conference on World Wide Web* (ACM, New York), 37–48.

Altman EI (1968) Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *J. Finance* 23(4):589–609.

Altman EI (2018) A fifty-year retrospective on credit risk models, the Altman Z-score family of models and their applications to financial markets and managerial strategies. *J. Credit Risk* 14(4):1–34.

Bae JW, Belo F, Li J, Lin X, Zhao X (2023) The opposing effects of complexity and information content on uncertainty dynamics: Evidence from 10-K filings. *Management Sci.* 69(10):5695–6415.

Bai C, Shi B, Liu F, Sarkis J (2019) Banking credit worthiness: Evaluating the complex relationships. *Omega* 83(C):26–38.

Ballen L (2021) Machine learning models and alternative data in credit scoring: Statistical and financial impact. MS Industrial Engineering thesis, Universidad de los Andes, Bogota, Columbia.

Barabasi A-L, Bonabeau E (2003) Scale-free networks. *Sci. Amer.* 288(5):60–69.

Benson AR, Gleich DF, Leskovec J (2016) Higher-order organization of complex networks. *Science* 353(6295):163–166.

Billio M, Getmansky M, Lo AW, Pelizzon L (2012) Econometric measures of connectedness and systemic risk in the finance and insurance sectors. *J. Financial Econom.* 104(3):535–559.

Black F, Scholes M (1973) The pricing of options and corporate liabilities. *J. Political Econom.* 81(3):637–654.

Blumauer A, Nagy H (2020) *The Knowledge Graph Cookbook* (Monochrom, Vienna, Austria).

Bonsall SB, Leone AJ, Miller BP, Rennekamp K (2017) A plain English measure of financial reporting readability. *J. Accounting Econom.* 63(2):329–357.

Bracke P, Datta A, Jung C, Sen S (2019) Machine learning explainability in finance: An application to default risk analysis. Staff Working Paper No. 816, Bank of England.

Brown S, Tucker JW (2011) Large-sample evidence on firms' year-over-year MD&A modifications. *J. Accounting Res.* 49(2): 309–346.

Cai J, Eidam F, Saunders A, Steffen S (2018) Syndication, interconnectedness, and systemic risk. *J. Financial Stability* 34(C):105–120.

Cao S, Lu W, Xu Q (2015) GraRep: Learning graph representations with global structural information. *Proc. 24th ACM Internat. Conf. on Information and Knowledge Management* (ACM, New York), 891–900.

Cao S, Jiang W, Yang B, Zhang AL (2020) How to talk when a machine is listening: Corporate disclosure in the age of AI. *Rev. Financial Stud.* 36(9):3603–3642.

Cavar D, Josefy M (2018) Mapping deep NLP to knowledge graphs: An enhanced approach to analyzing corporate filings with regulators. *Proc. 11th Internat. Conf. on Language Resources and Evaluation* (ELRA).

Chan-Lau JA (2006) Fundamentals-based estimation of default probabilities: A survey. Working paper, International Monetary Fund, Washington, DC.

Chen H, Perozzi B, Al-Rfou R, Skiena S (2018) A tutorial on network embeddings. Preprint, submitted August 8, https://doi.org/10.48550/arXiv.1808.02590.

Cheng D, Niu Z, Zhang Y (2020) Contagious chain risk rating for networked-guarantee loans. *Proc. 26th ACM SIGKDD Internat. Conf. on Knowledge Discovery & Data Mining* (ACM, New York), 2715–2723.

Ciano G, Rossi A, Bianchini M, Scarselli F (2022) On inductive–transductive learning with graph neural networks. *IEEE Trans. Pattern Anal. Machine Intelligence* 44(2):758–769.

Cohen L, Malloy C, Nguyen Q (2020) Lazy prices. *J. Finance* 75(3): 1371–1415.

Das SR (2016) Matrix metrics: Network-based systemic risk scoring. *J. Alternative Investment* 18(4):33–51.

Das SR, Hanouna P, Sarin A (2009) Accounting-based vs. market-based cross-sectional models of CDS spreads. *J. Bank. Finance* 33(4):719–730.

Das SR, Kim S, Ostrov D (2019) Dynamic systemic risk networks. *J. Financial Data Sci.* 1:141–158.

Das SR, Mitchener KJ, Vossmeyer A (2021) Bank regulation, network topology, and systemic risk: Evidence from the Great Depression. *J. Money Credit Bank* 54(5):1261–1312.

Das SR, Donini M, Zafar MB, He J, Kenthapadi K (2022) FinLex: An effective use of word embeddings for financial lexicon generation. *J. Finance Data Sci.* 8:1–11.

Desola V, Hanna K, Nonis P (2019) FinBERT: Pre-trained model on SEC filings for financial natural language tasks. Working paper, University of California, Berkeley.

Duffie D, Singleton KJ (1999) Modeling term structures of defaultable bonds. *Rev. Financial Stud.* 12(4):687–720.

Duvenaud DK, Maclaurin D, Iparraguirre J, Bombarell R, Hirzel T, Aspuru-Guzik A, Adams RP (2015) Convolutional networks on graphs for learning molecular fingerprints. Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R, eds. *Advances in Neural Information Processing Systems*, vol. 28 (Curran Associates, Inc., San Jose, CA), 2224–2232.

Dwyer D, Kocagil A, Stein R (2004) *The Moody's KMV RiskCalc v3. 1 Model: Next-Generation Technology for Predicting Private Firm Credit Risk* (Moody's KMV).

Erickson N, Mueller J, Shirkov A, Zhang H, Larroy P, Li M, Smola A (2020) AutoGluon-Tabular: Robust and accurate AutoML for structured data. *7th ICML Workshop on Automated Machine Learning*.

Falkenstein EG, Boral A, Carty LV (2000) Riskcalc for private companies: Moody's default model. Working paper, Moody's Inc., New York.

Fortunato S (2010) Community detection in graphs. *Phys. Rep.* 486(3):75–174.

Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. *Proc. 34th Internat. Conf. on Machine Learning* (PMLR), 1263–1272.

Goyal P, Ferrara E (2018) Graph embedding techniques, applications, and performance: A survey. *Knowledge Base System* 151: 78–94.

Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. *Proc. 22nd ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining* (ACM, New York), 855–864.

Gunning R (1952) *The Technique of Clear Writing* (McGraw-Hill, Toronto, Canada).

Gupta S, Kumar P (2021) A constrained agglomerative clustering approach for unipartite and bipartite networks with application to credit networks. *Inform. Sci.* 557:332–354.

Hamilton WL (2020) *Graph Representation Learning* (Morgan & Claypool, San Rafael, CA).

Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. *Proc. 31st Internat. Conf. on Neural Inform. Processing Systems* (Curran Associates, Red Hook, NY), 1025–1035.

Hamilton WL, Ying R, Leskovec J (2018) Inductive representation learning on large graphs. Preprint, submitted September 10, https://doi.org/10.48550/arXiv.1706.02216.

Helwege J, Zhang G (2016) Financial firm bankruptcy and contagion. *Rev. Finance* 20(4):1321–1362.

Hilscher J, Wilson M (2016) Credit ratings and credit risk: Is one measure enough? *Management Sci.* 63(10):3414–3437.

Hutto C, Gilbert E (2014) VADER: A parsimonious rule-based model for sentiment analysis of social media text. *Proc. Internat. AAAI Conf. on Web and Social Media 8*.

Jarrow RA, Turnbull SM (1995) Pricing derivatives on financial securities subject to credit risk. *J. Finance* 50(1):53–85.

Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. *Proc. 5th Internat. Conf. on Learn. Representations* (ICLR, Toulon, France).

Kok C, Montagna M (2016) Multi-layered interbank model for assessing systemic risk. ECB Working Paper No. 1944, European Central Bank.

Li F (2010) The information content of forward-looking statements in corporate filings–A naïve Bayesian machine learning approach. *J. Accounting Res.* 48(5):1049–1102.

Li Y, Tarlow D, Brockschmidt M, Zemel RS (2016) Gated graph sequence neural networks. Bengio Y, LeCun Y, eds. *Proc. 4th Internat. Conf. on Learn. Representations* (ICLR, San Juan, Puerto Rico).

Loughran T, Mcdonald B (2011) When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *J. Finance* 66(1):35–65.

Loughran T, McDonald B (2014) Measuring readability in financial disclosures. *J. Finance* 69(4):1643–1671.

Ma Y, Tang J (2021) *Deep Learning on Graphs* (Cambridge University Press, Cambridge, UK).

Merton RC (1974) On the pricing of corporate debt: The risk structure of interest rates. *J. Finance* 29(2):449–470.

Muñoz-Cancino R, Bravo C, Ríos SA, Graña M (2021) On the combination of graph data for assessing thin-file borrowers' creditworthiness. Preprint, submitted November 26, https://doi.org/10.48550/arXiv.2111.13666.

Ohlson JA (1980) Financial ratios and the probabilistic prediction of bankruptcy. *J. Accounting Res.* 18(1):109–131.

Óskarsdóttir M, Bravo C (2021) Multilayer network analysis for improved credit risk prediction. *Omega* 105:102520.

Ou M, Cui P, Pei J, Zhang Z, Zhu W (2016) Asymmetric transitivity preserving graph embedding. *Proc. 22nd ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining* (ACM, New York), 1105–1114.

Perozzi B, Al-Rfou R, Skiena S (2014) DeepWalk: Online learning of social representations. *Proc. 20th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining* (ACM, New York), 701–710.

Poenaru-Olaru L (2020) Credit scoring prediction using graph features. MS thesis, Delft University of Technology, Delft, Netherlands.

Poledna S, Molina-Borboa JL, Martínez-Jaramillo S, van der Leij M, Thurner S (2015) The multi-layer network nature of systemic risk and its implications for the costs of financial crises. *J. Financial Stability* 20(C):70–81.

Rios J, Zhao K, Street WN, Tian H, Zheng X (2020) A hybrid deep learning model for dynamic stock movement predictions based on supply chain networks. *Workshop on Information Technology and Systems (WITS)* (Social Science Research Network, Rochester, NY).

Roukny T, Battiston S, Stiglitz JE (2018) Interconnectedness as a source of uncertainty in systemic risk. *J. Financial Stability* 35:93–106.

Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2009) The graph neural network model. *IEEE Trans. Neural Networks* 20(1):61–80.

Shumovskaia V, Fedyanin K, Sukharev I, Berestnev D, Panov M (2021) Linking bank clients using graph neural networks powered by rich transactional data. *Internat. J. Data Sci. Anal.* 12(2):135–145.

Sobehart JR, Stein R, Mikityanskaya V, Li L (2000) Moody's public firm risk model: A hybrid approach to modeling short term default risk. Moody's Investors Service, Global Credit Research, Rating Methodology, March.

Surana A, Kumara S, Greaves M, Raghavan UN (2005) Supply chain networks: A complex adaptive systems perspective. *Internat. J. Production Res.* 43(20):4235–4265.

Tobback E, Moeyersoms J, Stankova M, Martens D (2016) Bankruptcy prediction for SMEs using relational data. Working paper, University of Antwerp, Faculty of Business and Economics.

Wang G-J, Jiang Z-Q, Lin M, Xie C, Stanley HE (2018) Interconnectedness and systemic risk of China's financial institutions. *Emerging Marketing Rev.* 35:1–18.

Wei Y, Yildirim P, Van den Bulte C, Dellarocas C (2016) Credit scoring with social network data. *Marketing Sci.* 35(2):234–258.

Ying Z, Bourgeois D, You J, Zitnik M, Leskovec J (2019) GNNExplainer: Generating explanations for graph neural networks. *Advances in Neural Information Processing Systems*, vol. 32 (Curran Associates, Red Hook, NY), 9240–9251.

Zaheer M, Guruganesh G, Dubey A, Ainslie J, Alberti C, Ontanon S, Pham P, et al. (2020) Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, vol. 33, 17283–17297.

Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, et al. (2020) Graph neural networks: A review of methods and applications. *AI Open* 1:57–81.