Product Competition Prediction in Engineering Design Using Graph Neural Networks

Faez Ahmed
Department of Mechanical Engineering,
Massachusetts Institute of Technology,
Cambridge, MA 02139
e-mail: faez@mit.edu

Yaxin Cui
Department of Mechanical Engineering,
Northwestern University,
Evanston, IL 60208-3111
e-mail: yaxincui2023@u.northwestern.edu

Yan Fu
Insight and Analytics Ford Motor Company,
Dearborn, MI 48124
e-mail: yfu4@ford.com

Wei Chen[1]
Department of Mechanical Engineering,
Northwestern University,
Evanston, IL 60208-3111
e-mail: weichen@northwestern.edu

Understanding relationships between different products in a market system and predicting how changes in design impact their market position can be instrumental for companies to create better products. We propose a graph neural network-based method for modeling relationships between products, where nodes in a network represent products and edges represent their relationships. Our modeling enables a systematic way to predict the relationship links between unseen products for future years. When applied to a Chinese car market case study, our method based on an inductive graph neural network approach, GraphSAGE, yields double the link prediction performance compared to an existing network modeling method—exponential random graph model-based method for predicting the car co-consideration relationships. Our work also overcomes scalability and multiple data type-related limitations of the traditional network modeling methods by modeling a larger number of attributes, mixed categorical and numerical attributes, and unseen products. While a vanilla GraphSAGE requires a partial network to make predictions, we augment it with an "adjacency prediction model" to circumvent the limitation of needing neighborhood information. Finally, we demonstrate how insights obtained from a permutation-based interpretability analysis can help a manufacturer understand how design attributes impact the predictions of product relationships. Overall, this work provides a systematic data-driven method to predict the relationships between products in a complex network such as the car market. [DOI: 10.1115/1.4054299]

Keywords: artificial intelligence, machine learning, graph neural networks, design automation, design for market systems

1 Introduction

Complex engineering systems contain multiple stakeholders and individual entities, which exhibit complex interactions and interconnections. Modeling and predicting relationships between these entities is key to studying them. An example of a complex engineering system is the car market, where there are many interactions between stakeholders. The success of a new car depends not only on its engineering performance but also on its competitiveness relative to similar cars and factors such as perceived market position. Customers from different geographies may prefer different types of vehicles. A design intervention in the car market, either by introducing changes in existing cars or launching a new car design, may encourage customers to change their driving behavior. When these changes happen, a manufacturer needs to understand which products their car models will compete in the new situation and what they can do to improve their market position. It is also important to consider the complex relationship among customers, such as the social network between customers and the complex relationships among products, such as the competitive relationship between products.

Network analysis is a crucial method for statistical analysis of complex systems in many scientific, social, and engineering domains [1–6]. Descriptive network analysis methods have been frequently applied in the engineering design field to study the engineering system [7]. More topological metrics from networks were tested by Haley et al. [8], who showed that they are applicable to describe complex engineering systems. Furthermore, a few studies have begun exploring the capability of statistical network models in modeling complex customer–product relationships [9,–12]. The underlying assumption in using the network-based approach is that customers and products can be viewed as components of a complex socio-technical system. Such a system can be analyzed using social network theories and computational techniques.

Researchers have employed exponential random graph models (ERGMs) as a statistical inference framework to interpret complex customer–product relations. ERGMs have been employed in the literature to study customers' consideration behaviors [13,–15]. These studies illustrated the benefits of using the network-based preference model for predicting the outcome of design decisions. However, ERGMs have a few limitations. First, they are typically appropriate for small to medium-sized networks with a few attributes. For large datasets, the Markov chain Monte Carlo (MCMC) approach to estimate ERGM parameters does not converge [16]. This leads to an important limitation for product manufacturers, who now want to make the most of massive datasets but still want statistical models to help them understand what is happening inside these models. In addition, previously published research shows that future market forecasts based on ERGMs are not sufficiently accurate at capturing the true network [17]. Poor forecasts can affect the manufacturer's market position as inaccurate predictions of the market competition can lead a manufacturer to wrongly estimate their future market position when they introduce a new car or change a feature in an existing car. If manufacturers rely on poor predictions to introduce design changes, the result will also affect the customers, as the new choices present in the market may lack what they desire. If car manufacturers have a method to predict competition for future years accurately, they can also use these predictions to identify competitors and incorporate them in designing their strategy for product placement, marketing, or redesign of the car. Manufacturers can also estimate how their market position may change when competitors introduce changes in existing attributes. This paper presents such an approach by modeling networks using neural networks, which does not face the issues highlighted above.

[1]Corresponding author.
Manuscript received October 18, 2021; final manuscript received April 4, 2022; published online May 9, 2022. Assoc. Editor: Mark M. Derriso.

Graph neural networks (GNNs) have recently become popular because they can model both discrete and continuous representations and have large expressive power [18]. Hence, they have a wide range of applications in domains that can harness graph structures out of their data. By supporting interpretability, causality, and inductive generalization, GNNs offer fundamental advantages over more traditional unstructured machine learning (ML) methods. Learning graph representations and performing reasoning and prediction have achieved impressive progress in applications ranging from drug discovery [19], image classification, natural language processing, and social network analysis [20]. A few of the well-known applications of GNNs are Uber Eats [21], which used them to recommend food items and restaurants, and Alibaba, which used them to model millions of nodes for product recommendation [22]. Within engineering design, although GNN usage is less common, researchers have recently used them for product tolerance design [23], machining feature recognition [24], and understanding mechanical device function [25]. These successes motivated us to use them for studying product relationships.

1.1 Our Contributions.

We use a GNN approach for predicting product relationships due to the inherent complexity of predicting whether two products are related to each other or not. In our approach, the products are nodes, and product relationships (such as product association and market competition) are links between those nodes. Hence, predicting relationships between products is posed as a graph link (or edge) prediction problem. Our approach uses GraphSAGE [26], a type of GNN method, which allows the modeling of design attributes. GraphSAGE represents a graph (network) structure in lower-dimension vectors and employs local neighborhoods to estimate these vectors for unseen nodes. These vectors are then utilized as the input for a downstream classification task. In this work, we also employ a permutation-based method to examine the feature importance to assist design decisions. In summary, the contributions of this study are as follows:

(1) Propose a GNN-based method for modeling a product relationship network and enabling a systematic way to predict the relationship links between unseen products for future years.
(2) Show that the link prediction performance of GNNs is better than existing network modeling methods.
(3) Demonstrate the scalability of the GNN method by modeling the effect of a large number of continuous and categorical attributes on link prediction.
(4) Uncover the importance of attributes to help make design decisions using permutation-based methods.

Below, we discuss the related work, our methodology, results, and discussion. A previous version [27] of this paper was presented at ASME IDETC 2021 conference.

2 Related Work

This paper applies GNNs to product relationship networks for link prediction and uncovers the importance of engineering design attributes using permutation-based analysis. This work focuses on the product co-consideration relation as a demonstration, but the method can be generalized to other product relationships, such as product association relationships. Below, we discuss related work on product co-consideration networks, GNNs, and interpretable machine learning.

2.1 Product Co-consideration Networks.

Co-consideration of products describes the situation where customers consider multiple products at the same time before making a purchase [28]. The consideration behavior involves the comparison and evaluation of product alternatives and is accordingly a critical step in the customer's decision-making process [29]. At the same time, product co-consideration also implies a market competition relationship between products. As a single product may be chosen by a customer considering two or more products, those products can increase their market share by understanding which alternatives are also being considered and introducing changes in their products such that the customers prefer them over their competitors. Therefore, successfully modeling the product co-consideration relationship can help companies understand the embedded market competition and provide them with new opportunities to formulate design solutions to meet customer needs.

Researchers have developed multiple methods and models of customer consideration behaviors to understand the underlying patterns of customer consideration behaviors. Some models of customer consideration set composition are based on the marginal benefits of considering an additional product [30,31]. Other pioneering works have built models for investigating the impact of the consideration stage on the customer decision-making process [32,33]. Many works use both online and offline customer activity data to infer the product co-consideration behavior [34]. In recent years, the network-based approach has emerged to understand the product competition by describing the product co-consideration relation based on customer cross-shopping data [16,17,28]. Depicted in a simple network graph, where nodes represent individual products and edges represent their co-consideration relation based on aggregated customer preference, network-based analysis views co-consideration relationships from the lens of network theories, where the underlying social processes explain the links in the observed network.

Several works that investigate the product co-consideration network are based on survey data collected from customers who purchased cars. Wang et al. [35] have applied the correspondence analysis methods and network regression methods to investigate the formation of car co-consideration relationships. Sha et al. [17] have applied ERGMs in understanding the underlying customer preference in car co-consideration networks. However, the previous explorations are restricted to using the traditional network-based statistical methods, which leads to a low computation efficiency, a low prediction accuracy for the future market competition, the inability to model many design attributes, and an inability to model both categorical and continuous design attributes simultaneously. To overcome the limitations of the ERGMs, we have developed a new method to investigate the underlying effect of customers' consideration behavior by using GNN methods. Applied to the same dataset, a comparison of the ERGMs and this work is summarized in Table 1. These comparisons will be further elaborated in the Results section, where the specifics of the experiments are provided.

2.2 Graph Neural Networks.

Network data can be naturally represented by a graph structure that consists of nodes and links. Recently, research on analyzing graphs with ML has grown

Table 1 Comparison of this work with prior studies on modeling car relationship using ERGM models

Topic	Past work using ERGM model	This work using GNN model
Test nodes	Only common cars between training (2013 data) and test data (2014 data). Trained on 296 cars	All cars in the training set. Trained on 388 cars. Tested on 403 cars from 2014 and 422 cars from 2015
Unseen data	Predictions restricted to cars in the training data	Predictions possible for entirely new cars too (107 unseen cars in 2014)
Attributes	Six design attributes restricted to numerical values	29 design attributes, including categorical attributes
Interpretability	Coefficient-based	Permutation-analysis based

rapidly. The graph-based ML tasks in networks typically include tasks such as node classification, link prediction, community detection, network similarity, anomaly detection, and attribute prediction [36]. In this paper, we focus on the link prediction task by representing product relationships as links between nodes.

In a graph, each node is naturally defined by its features and the neighborhood of connected nodes. The behavior of a node is often directed not only by its features but also by its neighbors, which makes it challenging to do feature engineering for graphs, that is, manually defining all features that may be important in making predictions about nodes. In such cases, graph representation learning methods are helpful, as they provide a way to automatically discover the vector representation of nodes that captures their graph structure and features from raw data. The outcome is a node embedding that can be viewed as learned features (or attributes) of a node. Ideally, similar nodes (ones with similar neighbors, connectivity, and similar features) should have similar node embeddings. Two nodes in a co-consideration network can uniquely define each edge, so edge embeddings can be calculated using the corresponding node embeddings. Using an appropriately defined loss function in their ML model, one can encourage all edges to have similar edge embeddings compared to nonexistent (negative) edges. Therefore, learning the representation of nodes in a graph, called *node embedding*, is an essential part of downstream tasks such as classification and regression.

There exist two major classes of the embedding algorithms: transductive learning and inductive learning. Transductive learning estimates the values of the remaining nodes and edges while knowing the ground truth of some nodes and edges on the graph. It refers to predicting connected unknown nodes and edges by using supervised learning with known nodes and edges. Node embedding models, such as the ones using spectral decomposition [37,38] or matrix factorization methods [39,40], are transductive. Inductive learning trains a model on a graph and then makes predictions for nodes and links on an entirely new graph. Although transductive approaches do not efficiently generalize to unseen nodes in the same graph (say for dynamically evolving graphs) and cannot learn to generalize across different graphs, they are still the most common method used in practice. Unlike most transductive graph learning methods, GraphSAGE method, which was proposed in 2017 [26], is an efficient inductive method that leverages the node attributes of neighboring nodes to generate representations on previously unseen data. GraphSAGE aggregates feature from a sample of the node's local neighborhood. Hence, training a GraphSAGE model on an example graph can generate node embeddings for previously unseen nodes too, as long as they have the same set of attributes as the training data (that is, no new attributes are introduced). GraphSAGE is advantageous for graphs with many node attributes, which is often the case for product networks.

2.3 Interpretable Machine Learning. In addition to using ML models for prediction, it is important for engineering applications to interpret what a model has learned so that these interpretations can throw some light on how different inputs affect the outcome. Interpretable ML methods present an effective tool to explain or present the model results in a way that is understandable to humans [41,42].

Identifying feature importance is a type of interpretable ML methods that can help with this goal. It indicates the statistical contribution of each feature to the underlying model [43]. Among the techniques to estimate the feature importance, model-agnostic methods [44] have the advantage that they can work with any ML model as they treat a model as a black-box and do not inspect internal model parameters. As graph neural networks are a type of black-box ML method, we focus on model-agnostic interpretable methods to explain their modeling results.

We use the *permutation feature importance measurement* in our work. This model-agnostic approach was introduced by Breiman [45] for random forests and then further developed by Fisher et al. [46]. The underlying idea behind this approach is to randomly permute a single feature in the dataset while keeping all other features intact. Then use a pretrained machine learning model to make predictions. If the feature is important, the predictions will get significantly worst after permuting the feature. Hence, the importance of a feature can be quantified by measuring how much the prediction metric changes [47]. The permutation-based feature importance method has been applied to bioinformatics [48], engineering [49], and political science [50] to provide insights into ML models. Our study uses permutation-based methods to examine which product attributes are important for link prediction between cars.

3 Methodology

We establish a product co-consideration network to model product competition behavior and use a GNN approach to predict future product competition. The methodology of the training and prediction process for the link existence is shown in Fig. 1.

Our methodology comprises five main components as follows:

(1) *Representing products and their relationships as a graph* (Sec. 3.1): this step involves the data processing and transformation to construct a network with products as nodes and their relationships as links.

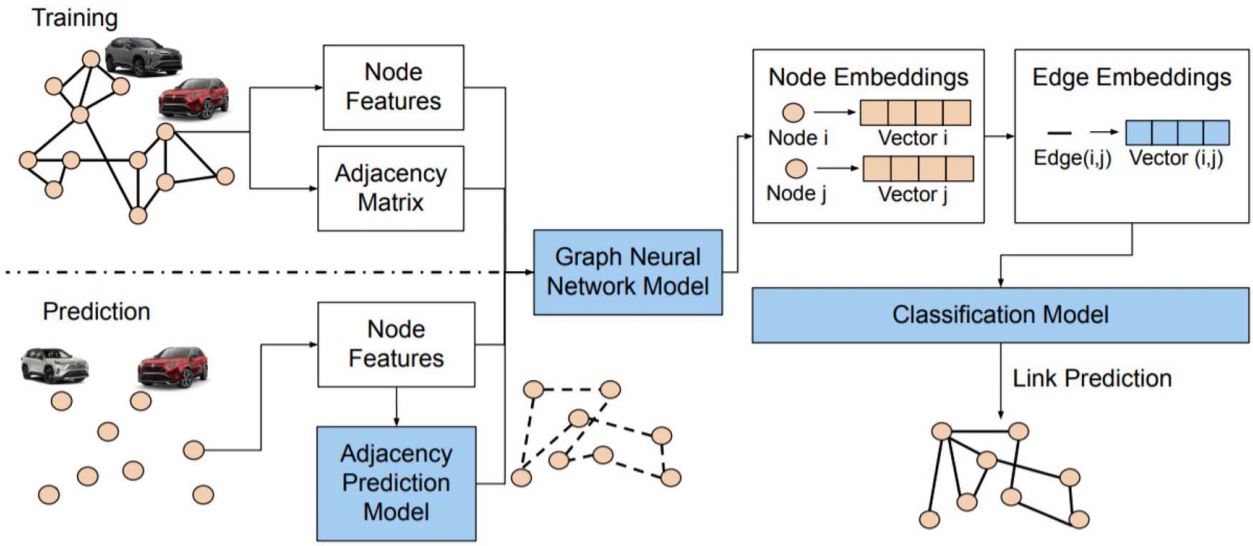

Fig. 1 The methodology of predicting the link existence in a car competition network using a graph neural network model

(2) *Training the GNN to learn the graph structure* (Sec. 3.2): this step finds a low-dimensional embedding of nodes and edges in the contracted graph.
(3) *Training classification models to make predictions* (Sec. 3.5): this step takes the graph embeddings as input to train a classification model on link existence.
(4) *Creating an adjacency prediction model to augment the GNN for unseen data* (Secs. 3.6, 3.7, and 3.8): for validation, the model is tested on the held-out network and unseen network. A proposed adjacency prediction model is applied in the unseen network prediction.
(5) *Interpreting the importance of design attributes* (Sec. 3.9): based on the model, this step investigates the importance of the features and provides useful insight for the engineering design.

The detailed components of each step are described in detail.

3.1 Network Construction. We translate customer survey data to a network, which simultaneously models products as nodes and relationships between them as edges. Before purchasing a product, customers often consider multiple products and select one or more products among them. When the same customer simultaneously considers two products in their decision-making process, we define this relationship as a co-consideration relationship. Assuming the customer only buys one product in the end, co-considered products are assumed to compete in this paper. Note that there are many different methods to measure competition between any two products, and the methods we describe next generalize to any measure of choice. Next, we discuss how a graph is created for co-considered products.

We studied a unidimensional product network that can reveal product market competition by describing products' co-consideration relationships. Each product corresponds to a unique node. Each node is associated with attributes such as price, fuel consumption, and engine power. The product co-consideration network is constructed using data from customers' consideration sets. The presence of a co-consideration binary link between two nodes (products) is determined by the number of customers who consider them together:

$$E_{i,j} = \begin{cases} 1, & n_{i,j} \geq c \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $E_{i,j}$ refers to the edge connected by node i and node j. $n_{i,j}$ is the number of customers who have considered products i and j together. c or the cutoff is a domain-dependent threshold, which defines the strength of the relationship considered in the analysis. In other words, we define an undirected link between node i and node j, if at least c customers consider both products i and j together. Based on Eq. (1), the network adjacency matrix is symmetric and binary.

3.2 Inductive Representation Learning on Networks. Many GNN models can learn functions trained on a graph and generate the embeddings for a node, which sample and aggregate feature and topological information from a node's neighborhood. However, engineering applications require methods that can make predictions about completely new nodes too. This need inspired us to employ GraphSAGE—a representation learning technique for dynamic graphs, which learns aggregator functions that can calculate new node embedding based on the features and neighborhood of a node.

As illustrated in Fig. 2, GraphSAGE learns node embeddings for attributed graphs (where nodes have features or attributes) through aggregating neighboring node attributes. The aggregation parameters are learned by the ML model by encouraging node pairs co-occurring in short random walks to have similar representations.

The detailed algorithm of GraphSAGE from Ref. [26] is shown in Algorithm 1.

Algorithm 1 GraphSAGE embedding generation (i.e., forward propagation) algorithm from Ref. [26]

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; nonlinearity σ; differentiable aggregator functions $aggregate_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output: Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow aggregate_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot concat(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

In GraphSAGE, it is assumed that every node can be defined by its neighbors, which means that the embedding for a node can be calculated by some combination of the embedding vectors of its neighbors. At the beginning of the training, every node's embedding is set equal to its feature vectors. The algorithm follows two main steps—aggregate and update (Steps 4 and 5 in Algorithm 1). The aggregate step uses any differentiable function to aggregate the embedding of neighbors to find the embedding of the target node. A

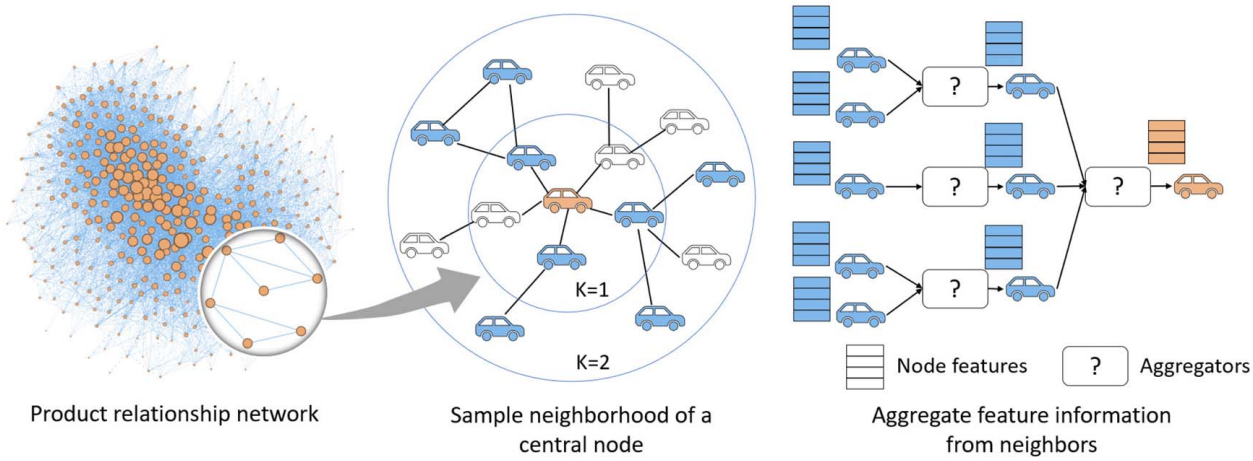

Fig. 2 Illustration of sampling and aggregation in GraphSAGE method. A sample of neighboring nodes contributes to the embedding of the central node.

typical example of the aggregate step can be a simple averaging of neighbors. The update step uses a differentiable function to combine the new aggregated representation for the target node with its previous representation. The K parameter tells the algorithms how many neighborhoods or hops to use to compute the representation for the target node. The aggregation can occur for first neighbors ($K = 1$) or from neighbors that are further away ($K \geq 1$). However, if too many neighbors at different depths are used that may dilute the effect of a local neighborhood. On the other hand, if only the first neighbors are considered, the method will be equivalent to using a simple neural network. Interested readers are encouraged to read [26] for details of the algorithm.

3.3 Node Embeddings. To train a GraphSAGE model, the inputs are the product attributes (i.e., node features) and the network structure (i.e., adjacency matrix) of the product co-consideration network. Then for each node, the GNN models can encode nodes into lower-dimensional space in the node embedding stage. For example, as illustrated in Fig. 1, nodes i and j can be represented by vectors i and j, which carry the information of node i's and j's features and local neighborhoods, respectively.

3.4 Edge Embeddings. Using a GNN-trained embedding for nodes, one can also learn the representation for all possible links (edges) in the network. Learning link representations is done by aggregating every possible pair of node embeddings. We use the dot product of vectors i and j to find the edge embeddings. Note that other symmetric operations such as averaging can also aggregate node embeddings to give an edge embedding. Our experiments found that the dot product gave slightly better results than the averaging operator (same F-1 score and 0.07 higher area under curve (AUC) score), which led us to select the dot product as the aggregation method in this paper. Once we learn the edge embeddings, they can be used as an input to any ML model, which can be trained to predict whether an edge exists or not, which is discussed next.

3.5 Classification Model for Link Prediction. The link prediction problem can be posed as a binary classification problem, where the goal is to predict whether a link candidate exists in the network (Class 1 or a positive edge) or does not exist (Class 0 or a negative edge). During the GNN model training, we can also train a downstream classification model to predict link existence, given the edge embedding as an input.

For each pair of nodes, the classification model takes the edge embeddings as input and whether the link exists or not as labels. Any classification model (such as logistic regression, k-nearest neighbors, and naive Bayes classifiers) can be integrated with the GNN model to predict the link existence. We used a multilayer perceptron (MLP) model for this work. Note that the GNN model and the MLP-based classification model are trained simultaneously for the supervised learning task in the training process. To avoid imbalanced training of the classification model for networks with very few edges, we balance the two classes by subsampling the negative edges (an edge that does not exist in the training data).

3.6 Validation Networks. After the training was completed, we tested the model's performance in predicting links for an unseen network. The model can be tested on two different types of networks. In one case, the initial network was divided into two parts by randomly sampling edges. The GNN model was tested to predict links for the held-out links. In the second case, we trained the model on one network and tested it on another completely unseen network. However, this presents new challenges, which are discussed next.

3.7 Adjacency Prediction Model. While GNN-based link prediction methods are typically used to find missing links from a graph, they cannot be directly applied to a completely unknown network. However, in engineering design applications, when design interventions (changing existing product attributes or launching new products) occur, it is desired to train a model in the current year and make predictions about the following year, which may have new products and evolved versions of previous products. Applications may require that predictions for product links are made where training and testing networks belong to different domains, periods, or locations. Making such predictions presents a circularity problem, as a typical GNN, including a vanilla GraphSAGE, needs at least a partial adjacency matrix as an input to predict the complete adjacency matrix.

We overcame this issue by developing a method to predict an approximate adjacency matrix using a separate ML model, which is referred to as the adjacency prediction model in Fig. 1. The predicted adjacency is used to identify a few neighbors of each node, used in the GNN as a partial adjacency matrix. There are several ways of predicting the adjacency matrix, given the node attributes. A naïve way would be to find all the nodes in the new graph, which also appeared in the training dataset, and copy their adjacency information. However, such a model performs poorly, as all the new nodes have no neighbors; therefore, the GNN cannot make accurate predictions about them.

Instead, we used a similarity-based k-nearest neighbor method in the adjacency prediction model. The similarities among product nodes are measured by the cosine distance of all car features. By using the similarities for each node with other nodes, the top K most similar nodes from the graph are selected as neighbors. This gives us the approximate adjacency matrix, where each node is connected to its k-nearest neighbors. The benefit of this approach is that all nodes in the co-consideration network are connected to some other nodes. While the choice of K is subject to the modeler, we seek an appropriate number to keep the network's density comparable with a typical co-consideration product network in the training network.

Note that other ML methods can also be used to output an approximate adjacency matrix. For instance, one can train a classification model with the average car attributes as input and a binary output corresponding to link existence. Our preliminary analysis showed that classification models (e.g., logistic regression) did not perform as well as the nearest-neighbor approach. This may be attributed to classification models not finding sufficient neighbors for all nodes. Our method overcame this limitation by assigning the same number of neighbors to all nodes, yielding good empirical results.

3.8 Metrics for Link Prediction. With the trained GNN model and classification model, we predicted the co-consideration network in the subsequent years based on the new node features and the approximate adjacency prediction model. The link prediction can be regarded as a binary classification model, which predicts the probability of the target link's existence to be Yes or No. To evaluate the performance of the classification model, we analyzed the confusion matrix (which describes the performance of a classifier) and the receiver operating characteristic (ROC) curve, which plots the true positive rate (TPR) and false positive rate (FPR). A confusion matrix consists of four values: true positives (TP) refer to the cases in which the model predicted "yes" and they are actually "yes," true negatives (TN) refer to the cases in which the model predicted "no" and they are actually "no," false positives (FP) refer to the cases in which model predicted "yes" but they are actually "no," and false negatives (FN) refer to the cases in which the model predicted "no" but they are actually "yes." We can further calculate other metrics such as precision, recall, and F1 score based on the confusion matrix. Precision is the ratio of true positives and total predicted positives (true positives and false positives). Recall is the ratio of true positives and total actual positives (true positive and false negatives). F-1 score, which is a widely used

metric for classification models, is the harmonic mean of precision and recall.

In our paper, to compare different models, we also used the AUC metric, which measures the area underneath the ROC curve, and provides an aggregated measure of the performance across all possible classification thresholds. The AUC ranges in value from 0 to 1. A higher AUC value indicates a better classification model. As a secondary metric, we also calculate the F-1 score, which also ranges from 0 to 1, and a higher F1 score indicates a better classification model.

3.9 Permutation-Based Feature Importance. Besides forecasting future market competition in the engineering design domain, it is important to understand the dominant features in product competition. Therefore, we investigated the importance of different design attributes in the GNN method using "Permutation feature importance" [51].

We used the method outlined in Ref. [51] to measure the importance of a feature by calculating how much a model's prediction error increases on average when a particular feature is permuted randomly. A feature was considered "important" if shuffling its values significantly increased the model error. This implied that the model relied on this feature to make accurate predictions, as measured by less prediction error. A feature was considered "unimportant" if shuffling its values left the model error unchanged. This implied that the model ignored the feature for the prediction and was not dependent on it to make good predictions. The outline of the permutation importance algorithm is described in section 8.1 in Ref. [51]. Interested readers are encouraged to refer to the related work for details of the algorithm.

Some other methods to calculate feature importance suggest removing features, retraining the model, and then comparing the model error. In contrast, permutation feature importance does not require retraining the model. Since the retraining of an ML model can take a long time, only permuting a feature can save time and inform us of the importance of features for that particular model. This technique is independent of what ML model is used and generally, several different permutations are used to estimate the metric. One also needs to define what metric (such as the AUC value for a classification model) they are using to calculate the change in performance. This metric does not reflect the intrinsic predictive value of a feature by itself. Instead, it shows how important the feature is for a particular model.

It is noteworthy that the permutation methods on feature importance can be applied to either training data or test data. Applying it to training data will help understand how much the model relies on each feature for making predictions (training data). Applying it to test data will help understand how much the feature contributes to the performance of the trained ML model on unseen test data. Our analysis uses it for the training data as the feature importance found using test data can change if the model is tested on different test sets.

4 Results and Discussion

In this section, we demonstrate the use of the GNN approach to study the Chinese car market. We used car survey data provided by the Ford Motor Company as a test example. We show that by training a GraphSAGE model, we can predict the future market competition even though cars in the future may have new attributes such as increased engine size or new products may be introduced. We also show how statistical methods can be employed to calculate the importance of each attribute for the relationship prediction task. This information can be reported back to designers to make strategic design changes.

4.1 Data Description. Our dataset contains customer survey data from 2012 to 2016 in the China market. In the survey, more than 40,000 respondents each year specified which cars they purchased and which cars they considered before making their final car purchase decision. Each customer indicated at least one and up to three cars which they considered. More concretely, in both year 2013 and year 2014, around 18% of respondents only considered one car, around 57% of respondents considered two cars, and around 25% of respondents considered three cars. The survey did not allow customers to report more than three cars that they considered. When two cars are co-considered, they are assumed to have a co-consideration relationship—predicting which is one of the goals of our proposed methods. In this study, we assume that a pair of cars co-considered by a respondent have an equal strength of relationship irrespective of the number of other cars that this respondent also considered with them. It is possible that when the respondents considered only two cars, it reflects a stronger co-consideration relation between those two cars than when respondents considered three cars. However, we use this assumption for our work due to a lack of data and quantitative evidence supporting different strengths of relationships between cars. We also aggregated the customers who considered a different number of cars to create the network. This aggregation may hide differences between different sets of customers (say those considering two cars versus those considering three cars), studying which is outside the scope of this work. In future work, we will explore this topic by training separate models for networks created by each set of people and identifying the differences between the network structures. The dataset resulting from the survey also contains attributes for each car (e.g., price, power, brand origin, and fuel consumption) and many attributes for each customer (e.g., gender, age).

4.2 Link Prediction for Car Co-Consideration Network. In this part, we used our method to build a model that predicts co-consideration links in the car dataset. We treat this problem as a supervised link prediction problem on a homogeneous network (nodes of only one type) with nodes representing cars and links corresponding to a car–car co-consideration relationship. Each node is also associated with attributes, listed in Table 6.

4.2.1 Network Construction. To study car co-consideration, we started by creating a car co-consideration network based on customers' survey responses in the 2013 survey data. The network consists of 388 unique car models represented as network nodes. The link between a pair of nodes (denoting cars) is allocated based on the car co-consideration by at least M customers. Unless otherwise specified, we use $M = 1$ for the experiments and later also show how the model performs for different values of M.

4.2.2 The Input Car Attributes. As demonstrated in the Methodology section, the car attributes and co-consideration network adjacency matrix serve as the input of the GNN and classification models, and the link existences are labels to judge the training performance. Our experiment studied manually chosen 29 car attributes. The list of attributes contains all the practical engineering attributes (e.g., fuel consumption, engine size) and car types (e.g., body type, market segmentation) available in the survey dataset. The attributes are listed in Table 6. Note that the attributes are both continuous and categorical. The categorical variables are transformed via a one-hot encoder which converts categorical variables into vectors (after one-hot encoding, 29 features lead to 210 features), and the continuous variables are normalized to vary between 0 and 1.

4.2.3 Experimental Settings. In the training stage, we built a model using the Stellar Graph library [52] with the following architecture. First, we built a two-layer GraphSAGE model ($K = 2$ in Algorithm 1) that trained on a network with node attributes and the binary adjacency matrix of the network (corresponding to co-consideration links) as inputs. The intermediate output of the model is node embeddings for all the cars in the training data. The node embeddings were then transformed to link embeddings

Table 2 Confusion matrix in predicting 2013 with 29 features

	2013 training prediction		2013 test prediction on held-out links	
	0	1	0	1
Actual class				
0	5390 (TNR 53.90%)	4610 (FPR 46.10%)	609 (TNR 54.82%)	502 (FPR 45.18%)
1	592 (FNR 5.92%)	9408 (TPR 94.08%)	75 (FNR 6.75%)	1036 (TPR 93.25%)

Note: Average F1-score for 2013 is 0.74. AUC for 2013 train is 0.84 and test is 0.84. TNR and TPR are shown in brackets.

by using a dot product for every pair of nodes.[2] The resultant link embedding was used to input a classification model comprising a dense neural network layer. For every candidate link, the classification layer outputs a probability, which measures whether a link exists or not. The entire model was trained end-to-end by minimizing the binary cross-entropy between predicted link probabilities and true link labels. The binary cross-entropy loss function was selected, as it encourages positive links to get high probabilities and negative links to receive low probabilities. The loss function is commonly used to measure the performance of a classification model whose output is a probability value between 0 and 1, which corresponds to link existence/nonexistence in our case. The model was trained on mini batches of positive and negative links from the training data, and the stochastic gradient descent (SGD) method was used to update the model parameters.

GraphSAGE algorithm requires setting up a few parameters, which are described next. We set the size of each minibatch (which measures the number of links shown to the model in each minibatch during training) to 20 and the number of epochs for training the model to 100. These values were empirically derived from experience and by observing how quickly the loss function converged for different settings. We set the number of 1-hop and 2-hop neighbor samples for GraphSAGE to be 5 and 5, respectively. These values determine how many neighbors are used to estimate the embedding of a node in Step 4 of Algorithm 1.

For the aggregator functions inside the GraphSAGE method, we selected hidden layer sizes of 5 for both the GraphSAGE layers and a bias term. To reduce overfitting, we used a dropout rate of 0.3. We stacked the GraphSAGE and classification layers for end-to-end supervised learning to minimize the binary cross-entropy. During prediction, we used the five nearest neighbors in the adjacency prediction model. Note that most of the above parameters were chosen based on an initial analysis of a validation set. Different choices of these parameters may be required for different problems. Our code is made public on Github[3] along with an anonymized dataset for other researchers to replicate our results.

4.2.4 Predicting Missing Links in the Same Year. In this part, we test our method for predicting held-out links from a network of cars from the 2013 data. We split the network into two parts to train the model by sampling a subset of links—the training graph and the test graph. Both the graphs contain the same nodes and do not contain any isolated nodes. For the training graph, an equal number of positive and negative edges were sampled to ensure that the model is trained on a balanced dataset. The test graph was used for evaluating the model's performance on held-out data.

A confusion matrix first measures the prediction performance along with the training performance in Table 2. The right-hand part of Table 2 shows the confusion matrix of 2013 test prediction on held-out links. It includes four different combinations of predicted and actual classes. The 609 in the top-left cell is the true negative (the model predicted negative, and it was true), and the 502 in the top right is the false positive (the model predicted positive, and it was false). The associated percentages indicate that for all pairs of nodes without link existence (actual class = 0), 54.82% are predicted correctly, whereas 45.18% are not. Meanwhile, the 75 in the bottom left is the false negative (model predicted negative and it was false), and 1036 in the bottom right is the true negative (the model predicted negative and it was true), which suggests that for all pairs of nodes with link existence (actual class = 1), 93.25% are predicted correctly while 6.75% are not. We further calculate other evaluation metrics to quantify classification performance. The F1 score, which measures the test accuracy in an unbalanced class, was *0.74* for the predicted missing links (the range of the F1 score was [0, 1]), while the AUC was *0.84* for both training set and held-out test set. The higher the AUC, the better the model is—it tells how capable the model is when distinguishing between classes. We note that overfitting is avoided because the AUCs for both the training and test sets are comparable. While the results in Table 2 are promising, they are of less practical usage. This is because a car manufacturer may care less about predicting relationships between cars in the year of survey completion and more about future predictions, enabling them to make strategic design decisions.

4.2.5 Predicting Entire Network for the Following Year. Once the trained model is converged, the learned parameters for the GNN model and the classification model can predict the co-consideration network in the future years. As a test dataset, the car co-consideration network in 2014 is predicted. First, the 2014 car model set, which has an intersection with the 2013 car set and has newly emerged cars, acts as the input of the prediction process without any link information. Then, an approximate adjacency matrix based on the similarities of nodes is generated through the adjacency prediction model. Next, the node features and approximate prediction model are fed into the GNN model, followed by the classification model. The link existence of each pair of nodes is forecasted with a certain probability threshold.

The performance of GNNs in predicting future networks is one of the most important results in this paper, which is highlighted in Table 3. We show the confusion matrix for the predicted 2014 co-consideration network in Table 3. Furthermore, we scoped out the AUC–ROC curve (in Fig. 3). The overall AUC for this curve is 0.80. To test the repeatability of our results, we conducted ten runs and found all runs to give results between 0.80 and 0.81. Later, we also discuss how the results generalize to networks created using different cutoff values and the number of neighbors.

4.2.6 Predicting Entire Network for the Year After Next. So far, we have predicted the 2014 co-consideration network based on the training data in 2013. However, as 2014 succeeded in 2013, the market structure did not change dramatically. Among 389 cars in 2013 and 403 cars in 2014, there are 296 cars in common. Therefore, to further assess the prediction capability for the model, we predict the 2015 co-consideration network using the trained model (2013 training data) with the car attributes and similarity-based adjacency matrix.

The predicted results are recorded and evaluated in Table 3 and Fig. 3, where the F1 score is 0.65 and AUC is 0.80. Compared to

[2]Other symmetric transformations can also be used such as the average or element-wise product of the node embeddings.

[3]https://github.com/Yaxin-Cui/Graph-Neural-Network

Table 3 Confusion matrix in predicting 2014 and 2015 with 29 features

	2014 test prediction on unseen network		2015 test prediction on unseen network	
	0	1	0	1
Actual class				
0	42633 (TNR 61.73%)	26435 (FPR 38.27%)	45735 (TNR 61.28%)	28893 (FPR 38.72%)
1	1811 (FNR 15.17%)	10124 (TPR 84.83%)	2195 (FNR 16.43%)	11167 (TPR 83.57%)
AUC	0.80		0.80	
F1 score	0.65		0.65	

Note: F1-score for 2014 is 0.65 and 0.65 for 2015. AUC for 2014 is 0.80 and 0.80 for 2015.

the prediction results in 2014, the prediction in 2015 maintains an equivalent performance, indicating model robustness.

4.2.7 Comparison With Existing Statistical Network Models. In this section, we compare the GNN method with an existing statistical network modeling method—ERGM. In order to make a fair comparison with the literature, we used the same set of input attributes (only six attributes in [16]) and compared the AUC of each model. Besides, as previous studies used a subset of cars and did not predict newly emerged car models, we also took the intersection of 2013 and 2014 cars (296 cars in total) for our analysis.

When only six car features were utilized in the training and prediction model, we found that the prediction results for 2014 data for GNN are significantly better than that of ERGM, as shown in Table 4. In the confusion matrix, we observed that in ERGM, the true positive rate (the ratio of true positive to all actual positive) is 79.81% and the true negative rate (TNR) (the ratio of true negative to all actual negative) is 40.51%. Both the values are lower than those predicted by GNN. Furthermore, the F1 score of the ERGM is merely 0.31, which is almost half the 0.60 F1 score of the GNN model. The AUC for ERGM prediction is 0.68, which is also less than the corresponding value of 0.78 for the GNN model. All of the evidence suggested that the prediction model of GNN performs better than the traditional statistical network models. It is also important to note that, unlike ERGM, GNN can model a large number of attributes (29 attributes) and unseen data. These benefits prove its effectiveness in modeling networks in comparison to other statistical methods.

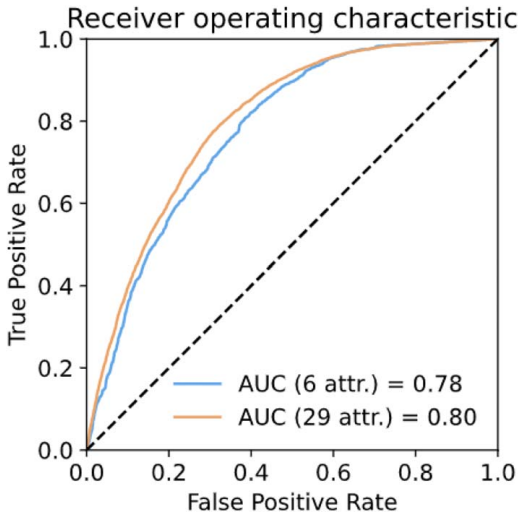

Fig. 3 AUC–ROC curve to predict 2014 co-consideration network with six attributes and 29 attributes

Then we summarized all the AUCs to compare in Table 5. Notice that the ERGM with 29 attributes does not associate with an AUC value because the model does not converge with many attributes. Meanwhile, we did not run the six attributes prediction for the 2015 data on the GNN because the common car set for 2013 and 2014 is no longer suitable for the 2015 car market. It is apparent from the comparison that the GNN models perform better than the ERGM model with a higher AUC and F1 score, and GNN models can accommodate larger networks with more design attributes and introduction of unseen nodes in the study of product relationships.

4.3 Interpretability of Attributes. We applied the permutation method to inspect the feature importance to find the decrease in a model score when a single feature value is randomly shuffled. We ran 50 permutations for each feature in the training data and calculated the drop in performance. These repeats in the process with multiple shuffles were done to ensure accuracy. The results are shown in Table 6. We found that the make of the car, the body type, and the segment are the most critical attributes for the GNN to predict ties.

Table 6 shows that 14 of the 29 attributes have no positive effect on the model prediction. Note that negative values are returned when a random permutation of a feature's values results in a better performance metric than before a permutation is applied. This means the model does not rely on features that have negative values when predicting links for the training data. We observe that most continuous values, such as engine size, price, fuel consumption, and power, are not important. This behavior may either reflect a trend in the data captured by the GNN model or may be

Table 4 Confusion matrix in predicting 2014 with six features and 296 cars for using the GNN method and the ERGM method

	2014 prediction class GNN		2014 prediction class ERGM	
	0	1	0	1
Actual class				
0	20336 (TNR 54.95%)	16675 (FPR 45.05%)	14993 (TNR 40.51%)	22018 (FPR 59.49%)
1	867 (FNR 13.04%)	5782 (TPR 86.96%)	1384 (FNR 20.82%)	5265 (TPR 79.18%)
AUC	0.78		0.68	
F1 score	0.60		0.31	

Note: F1 score is 0.60 for the GNN model and 0.31 for the ERGM model, and the AUC is 0.78 for the GNN model and 0.68 for the ERGM model.

Table 5 Comparing train AUC and test AUC in different years, different models, and different sets of attributes

Number of attributes	Train AUC (2013)	Test AUC (2014)	Test AUC (2015)	Test AUC (ERGM)
29 attributes	0.84	0.80	0.80	NA
Six attributes	0.81	0.78	NA	0.68

Note: AUC in link prediction. The goal is to predict the entire network (all existing and nonexisting edges) in a 0/1 classification task.

caused by a methodology limitation of the applicability of permutation-based methods for mixed (continuous and discrete) data. Understanding the cause of this trend is an exciting direction of research, which can be explored in future work on interpretability analysis.

4.4 Effect of Network Density and Neighbors. In the analysis so far, we created a binary network by assuming that a relationship exists between two cars if at least one person considers them together. However, there is no strong motivation to use this specific cutoff, and different applications may quantify relationships using different methods. For example, a car manufacturer may also decide to use a higher cutoff value to create a different binary network with fewer links. To test the generalizability of our method for different networks, we train our models for networks with different cutoff values, as shown in Table 7. The table reports the AUC value for link predicting for unseen 2014 networks when the model is trained on networks derived from 2013 data. The results show that our method consistently achieves high AUC values, even when trained on different networks.

To further establish the generalizability of the approach, we test how choices of the number of neighbors in the adjacency prediction model affect the final classification performance. We use a different number of neighbors (denoted by N) to create an input adjacency matrix for the test data. The AUC results in Table 7 show that the choice of the number of neighbors does not significantly impact the prediction performance of our models. The model accurately predicts the relationships consistently for different networks and different parameter choices. However, we would caution that while these generalizability tests look promising for the Chinese car market data, the results may not necessarily translate to other domains with different network structures and attributes.

Finally, one may question, why not use the adjacency prediction model directly to predict links instead of using it as an input to the GNN? As the adjacency prediction model does not consider the neighbor's effect and the graph connectivity, it is often inaccurate. For instance, the AUC of our adjacency prediction model is 0.52, which, when input to the GNN model, leads the GNN to have an AUC value of 0.80 (Table 3). However, improvements in the accuracy of the adjacency prediction model will also lead to improvements in the GNN performance. To test the maximum improvement in prediction performance that the adjacency prediction model can provide, we tested the predictions by using the true adjacency matrix as an input (equivalent to a perfect input matrix provided by the most accurate adjacency prediction model

Table 6 Car attributes type and feature importance

Attribute	Variable Type	Importance	Sample Values
Make	Categorical, Nominal	$2.44 \cdot 10^{-2}$	Audi, Ford
Body Type and Doors	Categorical, Nominal	$1.26 \cdot 10^{-2}$	2 Door Coupe
Segment (Detailed)	Categorical, Nominal	$1.15 \cdot 10^{-2}$	CD Premium Car
Segment Number	Categorical, Nominal	$8.9 \cdot 10^{-3}$	1, 2, 3
Segment (Combined)	Categorical, Nominal	$8.1 \cdot 10^{-3}$	B, C
Market Category	Categorical, Nominal	$5.9 \cdot 10^{-3}$	Small Size
Body Type	Categorical, Nominal	$5 \cdot 10^{-3}$	Coupe
Community	Categorical, Nominal	$4.5 \cdot 10^{-3}$	1, 2, 3
Brand Origin	Categorical, Nominal	$4.1 \cdot 10^{-3}$	European, Japanese
Import	Categorical, Binary	$3.8 \cdot 10^{-3}$	[0, 1]
Lane Assistance	Categorical, Binary	$1.5 \cdot 10^{-3}$	[0, 1]
Third row of seats	Categorical, Binary	$1.4 \cdot 10^{-3}$	[0, 1]
Park Assistance	Categorical, Binary	$6 \cdot 10^{-4}$	[0, 1]
AWD	Categorical, Binary	$3 \cdot 10^{-4}$	[0, 1]
Leather Seats	Categorical, Binary	$1 \cdot 10^{-4}$	[0, 1]
EngineSize log	Numerical, Continuous	0	10.4409
Alloy Wheels	Categorical, Binary	$-2 \cdot 10^{-4}$	[0, 1]
Fuel Consumption	Numerical, Continuous	$-2 \cdot 10^{-4}$	8.216
Fuel per Power	Numerical, Continuous	$-2 \cdot 10^{-4}$	0.066
Luxury	Categorical, Binary	$-3 \cdot 10^{-4}$	[0, 1]
Autotrans	Categorical, Binary	$-3 \cdot 10^{-4}$	[0, 1]
Year of Data	Numerical, Discrete	$-4 \cdot 10^{-4}$	2013,2014
Price log	Numerical, Continuous	$-4 \cdot 10^{-4}$	16.0406
Stability Control	Categorical, Binary	$-5 \cdot 10^{-4}$	[0, 1]
Fuel Type	Categorical, Nominal	$-5 \cdot 10^{-4}$	ICE
Power log	Numerical, Continuous	$-7 \cdot 10^{-4}$	6.7535
Side Airbags	Categorical, Binary	$-7 \cdot 10^{-4}$	[0, 1]
Navigation	Categorical, Binary	$-8 \cdot 10^{-4}$	[0, 1]
Turbo	Categorical, Binary	$-1.4 \cdot 10^{-3}$	[0, 1]

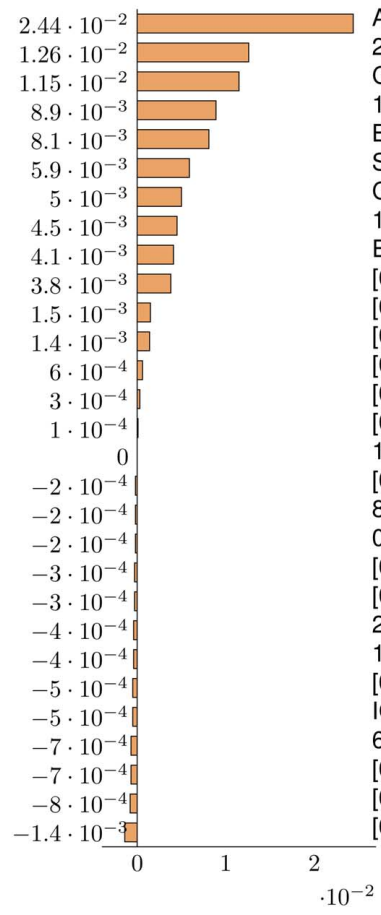

Table 7 Test AUC for 2014 for training networks of different densities

Cutoff (M)	# of links	$N=2$	$N=5$	$N=10$	$N=20$
1	11111	0.81	0.81	0.81	0.81
5	2546	0.87	0.88	0.86	0.84
10	1293	0.88	0.87	0.86	0.85
15	833	0.84	0.87	0.85	0.85
20	561	0.84	0.86	0.86	0.85

Note: GNNs perform well for different binary networks derived by using different cutoff values. We also observe that the effect of the choice of the number of neighbors does not have a large impact on the prediction performance.

with AUC equal to 1). Using this matrix as input, the AUC for predicting 2014 by the GNN increased from 0.80 (Table 3) to 0.83. In contrast, to test the case when the least informative adjacency matrix is input to the GNN, we input an adjacency matrix where all nodes are connected to all other nodes (input matrix with AUC equal to 0.5). In that case, the GNN's AUC drops to 0.72. As one can note, improvements in the input adjacency matrix, equivalent to more accurate identification of neighbors, also lead to improvements in the GNN performance.

5 Discussion

A car is an expensive commodity, and customers usually consider multiple options before deciding which car to buy. This decision may be influenced by many factors, such as the customer's budget, driving needs, required and necessary features, the popularity of nearby car models, brand, experience, and the influence of cars owned or recommended by family and friends. From a manufacturer's perspective, it is crucial to understand the market competition and develop strategies to improve their market share. The proposed model can support designers in the following aspect:

First and foremost, the prediction capability of the GNN model facilitates the forecast of future market competition when a new car is introduced or the attributes of an existing car change. Designers can use the model to anticipate the outcomes of a design change or a design release. For example, when a new car is released, the model can predict whether it will be competitive in the car market and what other cars will be potential competitors of the new car model (considered concurrently) by predicting the co-consideration link existence. Therefore, designers or manufacturers can use this information to develop their design strategies. For example, designers would like to improve the performance of a current model given a fixed amount of budget (either upgrade the power level or add more user-friendly features). They can forecast the derived market competition given the design changes and choose the strategy that maximizes its competitiveness. Meanwhile, as the model can predict the potential competitors of the changed/new car model, it would be beneficial to set marketing plans accordingly to highlight the car models among the competitors.

To evaluate the overall performance of the prediction, F1 and AUC scores are adopted, where F1 serves as a comparison indicator between precision (catch the true positive—the existence of link) and recall (indication of not missing true positive) at a certain threshold and AUC scores measure the average prediction performance with different thresholds. It is striking that the F1 score in the GNN model (0.60) is almost double of the ERGM model (0.31), which substantially increases the prediction capability. In addition, among different metrics, we are specifically interested in the true positive rate (also referred to as sensitivity or recall), which measures the proportion of existing co-consideration links that are correctly identified. The correct identification of link existence can prepare a design team with potential competitors and adjust design and marketing strategy. It is noticeable that the true positive rate for the prediction is over 80% for all the results shown, which shows that there is a considerably high probability that an actual link exists that will test positive. This indicates that the prediction model can well capture competition in the future.

Second, the feature importance results shed light on understanding the critical features in the co-consideration network formation. The results of the feature importance in Table 6 show that some features, such as make, body type, import, lane assistance, third row, park assistance, and AWD, have a higher impact on the product co-consideration network, whereas other features, such as turbo and navigation, are not critical factors in making predictions. Knowing these factors and introducing interventions to change them for future product iterations can enable a car manufacturer to affect the competition relationships, leading to a larger market share. However, we should warn that it is imprudent to make definitive conclusions from regression models without real-world validation. Models should first be validated by an expert's opinion of the plausibility of model results. There may be a discrepancy between what customers reported in the survey and how they behave, which is essential to consider while interpreting the results. One method to validate the model's predictions is to show the customers the forecasts and collect their feedback on how well it aligns with them. Nevertheless, our analysis sheds light on critical factors that customers may be considering while making their purchase decisions.

This study also has potential limitations. As a data-driven method based on deep learning, the model can only discover market competition patterns in general circumstances, which is based on the assumption that the underlying customer preference keeps unchanged. When a tremendous change occurs (such as the global pandemic in 2020) and customer preference changes sharply, the model will not learn such external influences. Besides, the performance of such deep learning models in general also relies on many parameters set based on the modeler's experience and experiments. It is possible that the model becomes overly sensitive for some settings of these parameters or performs poorly. Therefore, to leverage the proposed model, it would be essential to be mindful that the premise is that the underlying network does not change dramatically and find the optimal set of parameters.

6 Conclusions

This paper presents a systematic method to study and predict the relationship between products by using the inductive graph neural network models. With a focus on product co-consideration relation as the aggregated result of customer preference from survey data, we exhibit the efficiency of GNNs in modeling and prediction relationships. The method also enables designers to forecast the market changes under design intervention and figure out the critical features in the market competition.

This work has three main contributions. First, we show that inductive graph neural network models, which can embed each node of a graph into a real vector, can capture node feature and graph structure information simultaneously to enable ML applications on complex networks. They also enable us to model categorical, ordinal, and numerical attributes simultaneously. Second, we show that GNN models have better link prediction performance than ERGMs, both for held-out links from the same year and predicting the entire network structure for future years. Third, we overcome a limitation of GNNs when applied to predicting an unseen network by proposing a new graph adjacency prediction model to enable link prediction between unseen products for future years. Meanwhile, we show the scalability of the GNN method by modeling the effect of a large number of continuous and categorical attributes on link prediction. In addition, we show how permutation-based methods can find the importance of attributes to help design decisions.

While the GNN method provides many advantages over existing network models, there are a few limitations and practical challenges that need more work: First, this study is limited by the nature of

survey data. The survey data only samples a small portion of the actual car market, which leads to a sparse network and an unbalanced dataset with most links classified as 0. We randomly selected a subset of samples from the original dataset to match the samples coming from both classes in the training process to overcome the issue. Another limitation lies in the possibility of multicollinearity in features when calculating their importance. Suppose two features are correlated, and we permute only one of these features. In this case, the information about the permuted feature is still available to the model through its correlated counterpart, leading to inaccurate assessments by a permutation-based method relying on one-at-a-time permutations. The problem of correlations among features is common in many interpretable ML problems, and our work is no exception. Future work will investigate methods to overcome these issues.

The findings of this study have several important implications for future practice. In future work, we aim to predict the product relationship strength and extend the current work on more complex network structures to investigate the relationship between customers and products. The current GNN work will be extended to modeling multidimensional customer–product complex relations in the future.

Acknowledgment

We are thankful to the d'Arbeloff career development chair for supporting Faez Ahmed's work. The authors are also grateful to the support from the National Science Foundation (Grant Nos. CMMI-2005661 and CMMI-2005665), the Intersection Science Fellowship and the Ford-Northwestern Alliance Project.

Conflict of Interest

This article does not include research in which human participants were involved. Informed consent not applicable.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

References

[1] Simon, H. A., 1977, *The Organization of Complex Systems*, Vol. 54, Springer, pp. 245–261.
[2] Wasserman, S., and Faust, K., 1994, *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*, Cambridge University Press.
[3] Holling, C. S., 2001, "Understanding the Complexity of Economic, Ecological, and Social Systems," Ecosystems, **4**(5), pp. 390–405.
[4] Newman, M. E., 2003, "The Structure and Function of Complex Networks," SIAM Rev., **45**(2), pp. 167–256.
[5] Braha, D., Suh, N., Eppinger, S., Caramanis, M., and Frey, D., 2006, *Unifying Themes in Complex Systems*, Springer, Berlin/Heidelberg, pp. 227–274.
[6] Hoyle, C., Chen, W., Wang, N., and Koppelman, F. S., 2010, "Integrated Bayesian Hierarchical Choice Modeling to Capture Heterogeneous Consumer Preferences in Engineering Design," ASME J. Mech. Des., **132**(12), p. 121010.
[7] Sosa, M. E., Eppinger, S. D., and Rowles, C. M., 2007, "A Network Approach to Define Modularity of Components in Complex Products," ASME J. Mech. Des., **129**(11), pp. 1118–1129.
[8] Haley, B. M., Dong, A., and Tumer, I. Y., 2016, "A Comparison of Network-Based Metrics of Behavioral Degradation in Complex Engineered Systems," ASME J. Mech. Des., **138**(12), p. 121405.
[9] Wang, M., Chen, W., Fu, Y., and Yang, Y., 2015, "Analyzing and Predicting Heterogeneous Customer Preferences in China's Auto Market Using Choice Modeling and Network Analysis," SAE Int. J. Mater. Manuf., **8**(3), pp. 668–677.
[10] Fu, J. S., Sha, Z., Huang, Y., Wang, M., Fu, Y., and Chen, W., 2017, "Modeling Customer Choice Preferences in Engineering Design Using Bipartite Network Analysis," Proceedings of the ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Cleveland, OH, Aug. 6–9.
[11] Sha, Z., Huang, Y., Fu, S., Wang, M., Fu, Y., Contractor, N., and Chen, W., 2018, "A Network-Based Approach to Modeling and Predicting Product Co-Consideration Relations," Complexity, **2018**, pp. 1–14.
[12] Ghosh, D., Olewnik, A., Lewis, K., Kim, J., and Lakshmanan, A., 2017, "Cyber-Empathic Design: A Data-Driven Framework for Product Design," ASME J. Mech. Des., **139**(9), p. 091401.
[13] Sha, Z., Saeger, V., Wang, M., Fu, Y., and Chen, W., 2017, "Analyzing Customer Preference to Product Optional Features in Supporting Product Configuration," SAE Int. J. Mater. Manuf., **10**(3), pp. 320–332.
[14] Wang, M., Chen, W., Huang, Y., Contractor, N. S., and Fu, Y., 2016, "Modeling Customer Preferences Using Multidimensional Network Analysis in Engineering Design," Des. Sci., **2**, pp. 1–28.
[15] Wang, M., Sha, Z., Huang, Y., Contractor, N., Fu, Y., and Chen, W., 2016, "Forecasting Technological Impacts on Customers' Co-consideration Behaviors: A Data-Driven Network Analysis Approach," ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Paper No. V02AT03A040.
[16] Cui, Y., Ahmed, F., Sha, Z., Wang, L., Fu, Y., and Chen, W., 2020, "A Weighted Network Modeling Approach for Analyzing Product Competition," International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 84003, American Society of Mechanical Engineers, Paper No. V11AT11A036.
[17] Sha, Z., Huang, Y., Fu, J. S., Wang, M., Fu, Y., Contractor, N., and Chen, W., 2018, "A Network-Based Approach to Modeling and Predicting Product Coconsideration Relations," Complexity, **2018**, pp. 1–14.
[18] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M., 2020, "Graph Neural Networks: A Review of Methods and Applications," AI Open, **1**, pp. 57–81.
[19] Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N. M., MacNair, C. R., French, S., Carfrae, L. A., and Bloom-Ackermann, Z., 2020, "A Deep Learning Approach to Antibiotic Discovery," Cell, **180**(4), pp. 688–702.
[20] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y., 2020, "A Comprehensive Survey on Graph Neural Networks," IEEE Trans. Neural Netw. Learn. Syst., **32**(1), pp. 4–24.
[21] Jain, A., Liu, I., Sarda, A., and Molino, P., 2019, "Food Discovery With Uber Eats: Using Graph Learning to Power Recommendations." . Accessed March 1, 2021.
[22] Wang, J., Huang, P., Zhao, H., Zhang, Z., Zhao, B., and Lee, D. L., 2018, "Billion-Scale Commodity Embedding for E-Commerce Recommendation in Alibaba," Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, pp. 839–848.
[23] Li, K., Gao, Y., Zheng, H., and Tan, J., 2021, "A Data-Driven Methodology to Improve Tolerance Allocation Using Product Usage Data," ASME J. Mech. Des., **143**(7), p. 071101.
[24] Cao, W., Robinson, T., Hua, Y., Boussuge, F., Colligan, A. R., and Pan, W., 2020, "Graph Representation of 3D CAD Models for Machining Feature Recognition With Deep Learning," ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Virtual Conference, Aug. 17–19.
[25] Wang, J., Chiu, K., and Fuge, M., 2020, "Learning to Abstract and Compose Mechanical Device Function and Behavior," ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Virtual Conference, Aug. 17 –19.
[26] Hamilton, W. L., Ying, R., and Leskovec, J., 2017, "Inductive Representation Learning on Large Graphs". arXiv preprint.
[27] Ahmed, F., Cui, Y., Fu, Y., and Chen, W., 2021, "A Graph Neural Network Approach for Product Relationship Prediction," ASME International Design Engineering Technical Conferences, Virtual Conference, Aug. 17–20.
[28] Wang, M., Sha, Z., Huang, Y., Contractor, N., Fu, Y., and Chen, W., 2018, "Predicting Product Co-consideration and Market Competitions for Technology-Driven Product Design: A Network-Based Approach," Des. Sci., **4**, p. e9.
[29] Shocker, A. D., Ben-Akiva, M., Boccara, B., and Nedungadi, P., 1991, "Consideration Set Influences on Consumer Decision-Making and Choice: Issues, Models, and Suggestions," Mark. Lett., **2**(3), pp. 181–197.
[30] Hauser, J. R., and Wernerfelt, B., 1990, "An Evaluation Cost Model of Consideration Sets," J. Consumer Res., **16**(4), pp. 393–408.
[31] Roberts, J. H., and Lattin, J. M., 1991, "Development and Testing of a Model of Consideration Set Composition," J. Mark. Res., **28**(4), pp. 429–440.
[32] Gaskin, S., Evgeniou, T., Bailiff, D., and Hauser, J., 2007, "Two-Stage Models: Identifying Non-Compensatory Heuristics for the Consideration Set Then Adaptive Polyhedral Methods Within the Consideration Set," Proceedings of the Sawtooth Software Conference, Santa Rosa, CA, Oct. 17–19, Vol. 13, Citeseer, pp. 67–83.
[33] Dieckmann, A., Dippold, K., and Dietrich, H., 2009, "Compensatory Versus Noncompensatory Models for Predicting Consumer Preferences," Judg. Deci. Making, **4**(3), pp. 200–213.
[34] Damangir, S., Du, R. Y., and Hu, Y., 2018, "Uncovering Patterns of Product Co-consideration: A Case Study of Online Vehicle Price Quote Request Data," J. Interact. Mark., **42**, pp. 1–17.
[35] Wang, M., Huang, Y., Contractor, N., Fu, Y., and Chen, W., 2016, "A Network Approach for Understanding and Analyzing Product Co-consideration Relations in Engineering Design," DS 84: Proceedings of the DESIGN 2016 14th International Design Conference, Cavtat, Dubrovnik, Croatia, May 16–19, pp. 1965–1976.
[36] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M., 2018, "Graph Neural Networks: A Review of Methods and Applications." arXiv preprint.

[37] Kipf, T. N., and Welling, M., 2016, "Semi-supervised Classification With Graph Convolutional Networks." arXiv preprint.
[38] Atwood, J., and Towsley, D., 2015, "Diffusion-Convolutional Neural Networks." arXiv preprint.
[39] Cao, S., Lu, W., and Xu, Q., 2016, "Deep Neural Networks for Learning Graph Representations," Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, Feb. 12–17.
[40] Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., and Tang, J., 2018, "Network Embedding As Matrix Factorization: Unifying Deepwalk, Line, Pte, and Node2vec," Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Marina Del Rey, CA, Feb. 5–9, pp. 459–467.
[41] Doshi-Velez, F., and Kim, B., 2017, "Towards a Rigorous Science of Interpretable Machine Learning." arXiv preprint, arXiv:1702.08608.
[42] Molnar, C., 2022, "A Guide for Making Black Box Models Explainable," *Interpretable Machine Learning*, 2nd ed.
[43] Du, M., Liu, N., and Hu, X., 2019, "Techniques for Interpretable Machine Learning," Commun. ACM, **63**(1), pp. 68–77.
[44] Ribeiro, M. T., Singh, S., and Guestrin, C., 2016, ""Why Should I Trust You?" Explaining the Predictions of Any Classifier," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, Aug. 13–17, pp. 1135–1144.
[45] Breiman, L., 2001, "Random Forests," Mach. Learn., **45**(1), pp. 5–32.
[46] Fisher, A., Rudin, C., and Dominici, F., 2019, "All Models Are Wrong, But Many Are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously," J. Mach. Learn. Res., **20**(177), pp. 1–81.
[47] Altmann, A., Toloşi, L., Sander, O., and Lengauer, T., 2010, "Permutation Importance: A Corrected Feature Importance Measure," Bioinformatics, **26**(10), pp. 1340–1347.
[48] Putin, E., Mamoshina, P., Aliper, A., Korzinkin, M., Moskalev, A., Kolosov, A., Ostrovskiy, A., Cantor, C., Vijg, J., and Zhavoronkov, A., 2016, "Deep Biomarkers of Human Aging: Application of Deep Neural Networks to Biomarker Development," Aging (Albany NY), **8**(5), p. 1021.
[49] Matin, S., Farahzadi, L., Makaremi, S., Chelgani, S. C., and Sattari, G., 2018, "Variable Selection and Prediction of Uniaxial Compressive Strength and Modulus of Elasticity by Random Forest," Appl. Soft Comput., **70**, pp. 980–987.
[50] Farinosi, F., Giupponi, C., Reynaud, A., Ceccherini, G., Carmona-Moreno, C., De Roo, A., Gonzalez-Sanchez, D., and Bidoglio, G., 2018, "An Innovative Approach to the Assessment of Hydro-Political Risk: A Spatially Explicit, Data Driven Indicator of Hydro-Political Issues," Global Environ. Change, **52**, pp. 286–313.
[51] Molnar, C., 2022, *Interpretable Machine Learning*, 2nd ed..
[52] Data61, C., 2018, Stellargraph Machine Learning Library. .