entropy

MDPI

Article

Returnformer: A Graph Transformer-Based Model for Predicting Product Returns in E-Commerce

Qian Cao [1,*], Ning Zhang [1] and Huiyong Li [2]

[1] School of Computer and Artificial Intelligence, Beijing Technology and Business University, Beijing 100048, China; 2331102067@st.btbu.edu.cn
[2] School of Artificial Intelligence (Institute of Artificial Intelligence), Beihang University, Beijing 100191, China; lihuiyong@buaa.edu.cn
* Correspondence: caoqian@th.btbu.edu.cn

Abstract

E-commerce retailers bear substantial additional costs arising from high product return rates due to lenient return policies and consumers' impulsive purchasing. This study aims to accurately predict product return behavior before payment, supporting proactive return management and reducing potential losses. Based on the Graph Transformer, we proposed a novel return prediction model, Returnformer, which focuses on capturing user–product connections represented in topological structures of bipartite graphs. The Returnformer first integrates global topological embeddings into original node features to alleviate structural information loss caused by graph partitioning. It then employs a Graph Transformer to capture long-range user–item dependencies within local subgraphs. In addition, a graph-level attention mechanism is introduced to facilitate the propagation of global return patterns across different subgraphs. Experiments on a real-world e-commerce dataset show that the Returnformer outperforms four machine learning models in terms of prediction accuracy, demonstrating superior performance compared to the state-of-the-art models. The proposed model enables retailers to identify potential return risks prior to payment, thereby supporting timely and proactive preventive interventions.

Keywords: return prediction; e-commerce; graph algorithm; topological structure

1. Introduction

With the continuous updating of e-commerce platform technology, global online sales have been rising in recent years [1]. However, the product return rate has continued to rise due to a relaxed return policy [2], the discrepancy between advertising and the actual product [3], and delayed delivery [4]. This is especially obvious in the fashion industry, where the return rate ranges from 13% to 96%, with an average of 53% [5]. The high return rate has significantly eroded the profits of retailers. What's more, product returns generate 2 to 16 times more greenhouse gases than the production and distribution of standard products [6]. Therefore, returns become a challenge to both the economy and ecology.

To effectively address the return problem, an in-depth understanding of customer behavior is essential, since return decisions are ultimately made by customers. Suppose retailers can predict the likelihood of product returns before customers pay, they may be able to take preventive measures to reduce this behavior, such as adjusting payment methods or offering reduced discounts. Therefore, a model that can estimate the probability of returns before payment can help retailers take timely measures to avoid the risk of returns.

Check for updates

Received: 28 November 2025
Revised: 23 December 2025
Accepted: 6 January 2026
Published: 8 January 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Recent studies on forecasting return intention are mainly categorized into machine learning technology [7–9] and graph representation learning [10–12]. Compared with traditional machine learning, graph representation learning, which efficiently extracts complex potential patterns and relational features from data, performs satisfactorily in modeling users' preferences [13]. However, due to the random division of the graph, the structural information of subgraphs becomes fragmented, which may hinder the transmission of information. Moreover, existing graph algorithms don't make more use of the structural information of large-scale graphs. In addition, most present studies focus on the feature similarity between any customers and between any products, while ignoring the interrelationship between nodes in subgraphs, which represents the similarity of return patterns. To address these problems, we designed a novel return prediction model, called Returnformer, based on the Graph Transformer. The model works at the level of individual product variants. The main contributions of this study are summarized as follows:

1. We introduced Returnformer, a new return prediction model based on the Graph Transformer. By using the self-attention layer, it can effectively aggregate high-order information in the customer–product interaction graph. Furthermore, we have also introduced the Kolmogorov–Arnold Network (KAN) as a decoder to improve classification performance by constructing complex decision-making boundaries through nonlinear transformation.
2. The topological embeddings generated by Node2Vec are utilized to supplement the structural information. And then we adopt a dual-path feature fusion method, which combines the structural embedding with the original feature embedding, to obtain the final fusion embedding as the input feature.
3. We use the graph-level attention mechanism to capture the relations between customers and products in different subgraphs. This method enables similar nodes in different subgraphs to interact, thereby capturing the global characteristics and patterns of customer return behaviors.

The rest of this paper is as follows. Section 2 reviews the related works on the return of products in online shopping. Subsequently, the methodology of this paper is outlined in Section 3, which describes the dataset used and details the developed Returnformer. The validity of Returnformer is verified, and a discussion is provided in Section 4, followed by the conclusions and future works in Section 5.

2. Related Works

There has been extensive research on product returns. Some focus on exploring the factors affecting the return of products, such as consumer behavior [3,14,15], product characteristics [16–19], retailer reputation [20], logistics [21,22], and return policy [2]. Some even conduct pricing decision-making research on different return policies [23]. However, forecasting product returns is the key to this study. Therefore, this section reviews related studies based on their prediction targets and modeling approaches.

2.1. Prediction of Product Return Rate and Return Volume

Research on return rate and return volume prediction aims to forecast the overall return trend. This helps enterprises manage inventory and formulate production strategies. Dzyabura et al. [5] explored a predictive model that can forecast the product return rate based on the image before the product goes on the market. Rajasekaran and Priyadarshini [24] introduced product feedback scores and divided the return rate into five levels to help e-commerce platforms identify products with high return rates. However, for manufacturers, the return volume is more important because it has a significant impact

on the production plan. Cui H et al. [25] used a LASSO model with main effects and interaction effects to forecast the future returns volume of automotive manufacturers.

Compared with traditional manufacturers, predicting the amount of returns in advance is more significant for remanufacturers since they need to make secondary use of returned products. Studies mentioned in [26,27] respectively adopted Grey-GERT and a dynamic prediction model selection algorithm to forecast the return of recyclable products, so as to improve the profitability of the remanufacturing industry. Chou et al. [28] further incorporated historical demand and sales data into their return forecasting model, supporting more adaptive remanufacturing strategies.

2.2. Prediction of Product Return Propensity

The forecast of the return volume and return rate usually focuses on the overall product return trend, rather than the forecast of a single product, and also, these predictions are made after purchase. Whereas consumers are often uncertain about whether to keep or return products. Previous studies tried to predict this uncertainty by predicting customers' tendency to return. These studies can be roughly divided into two categories according to different strategies. The former adopts traditional machine learning to predict returns, while the latter focuses on using graph algorithms to predict the probability of returns.

2.2.1. Traditional Machine Learning Models for Return Prediction

In terms of feature construction, Fu Y et al. [29] developed a general return tendency model by using the potential vector obtained from the decomposition of product returns to forecast the return tendency of consumers. Urbanke P [30] designed a feature extraction method for high-dimensional sparse data to achieve feature dimension reduction and predict product returns. In addition, Duong Q H et al. [31] mainly extracts the intrinsic and external attributes of the product from customer reviews to forecast the probability of returns. Notably, Hofmann A [32] proved that return behaviors can be effectively predicted based only on basic order characteristics such as product name, price, and quantity.

From a modeling perspective, some studies regard the problem of consumer returns as a second classification problem. By setting a threshold, they convert the predicted return probability into binary prediction results to forecast whether consumers will return the goods. These studies systematically compare various traditional machine learning models [7,8], and even evaluated the model performance under different optimizations [9] to balance predictive accuracy with interpretability.

2.2.2. Leveraging Graph Representation for Return Prediction

The inherent structure of consumer-product interactions naturally exhibits graph-like characteristics. Therefore, there is a growing trend of research that explores the potential of graph representation learning on product returns. Different studies have modeled consumer purchase and return records using various graph algorithms, including hypergraphs [10], weighted graphs [11], heterogeneous graphs [12,33], and bipartite graphs [13,34,35].

Li et al. [10] and Zhu et al. [11] employed hypergraph and weighted hybrid graph modeling, respectively, to compute feature similarities and identify similar shopping baskets, customers, or products. They then utilized transition probabilities from random walk and its variant algorithms to forecast return likelihood, and further return behaviors. Similarly, Li et al. [12] incorporated order-user similarity into a trust network, then applied random walk algorithms for prediction.

In contrast, Joshi et al. [34] divided consumer-product interaction networks into various behavior communities via community detection, employing Support Vector Machines (SVM) for prediction. Kedia et al. [35] integrated industry-specific characteristics and users'

body shape by combining MF-BPR with Skip-gram models, and then deep neural networks were adopted to forecast returns. The similarity of these approaches is that they extract features before final classifying through predefined algorithmic rules, such as walking, partitioning, and similarity calculation.

The emergence of graph neural networks (GNNs) [36–39] has provided a new way to learn node representation, which can aggregate neighbor information through a message-passing mechanism to complete the update of node representation. McGowan et al. [13] demonstrated the superior performance of GNN for returns. On this basis, Ma and Wang [33] proposed an inductive HGNN that integrates user, product, and order features to construct a return prediction framework.

Although GNNs offer greater advantages over traditional graph algorithms, these approaches still have a deficiency of local information aggregation [40]. Furthermore, e-commerce interaction graphs are typically large-scale. To improve computing efficiency, this graph is usually randomly divided into countless subgraphs, which will lead to the destruction of global structural information. Additionally, these studies primarily focus on the features of individual nodes and neglect the correlation of different nodes between subgraphs. Table 1 compares the return prediction research using different graph representation methods.

Table 1. The comparison of different studies on return prediction using graph representation methods.

Studies	Method	Comparison of Predictive Method Considerations			
		Similarity	**Topological Structure**	**Graph Partitioning**	**Inter-Graph Correlation**
Li et al. [10]	random walks	A truncated random walk	–	Local graph partitioning	–
Zhu et al. [11]	A random walk-based local algorithm	Hybrid similarity	–	LoGraph algorithm	–
Li et al. [12]	A trust-aware random walk model	Enhanced Pearson similarity	–	–	Trust-aware subgraph fusion
Ma and Wang [33]	Heterogeneous Graph Neural Network	–	Message passing	Heterogeneous graph sampling algorithm	–
Joshi et al. [34]	community detection +SVM	–	–	–	Multi-view subgraph fusion
Kedia et al. [35]	MF–BPR + skip-gram	–	–	–	–
McGowan et al. [13]	Graph Neural Network	–	Message passing	–	–
This study	Returnformer	Topological interaction similarity	Node2Vec to data Augmentation	Balanced-edge partitioning	Graph-level Attention Mechanism

Inspired by the performance of Graph Transformer [41] on large-scale graphs, we develop a new return forecasting model, Returnformer, based on the Graph Transformer to address these constraints. Our method explicitly integrates the embedding of topological structure. What's more, it employs the graph-level attention mechanism [42] to capture the return pattern between different subgraphs. we ultimately introduce the Kolmogorov–Arnold Networks (KAN) [43] as a classifier to get the final return probability.

3. Methodology

We first introduce the dataset used in this study, along with the bipartite graph established herein. And subsequently, a detailed exposition of prediction model is provided.

To better illustrate the research process, Figure 1 summarizes the main steps of this study. The process begins with data description and preprocessing, followed by the construction of a customer–product bipartite graph. Global structural features is extracted and fused with original features. And then the fused features are used as inputs to the proposed Returnformer for training. The model performance is finally evaluated on the test set using standard classification metrics.

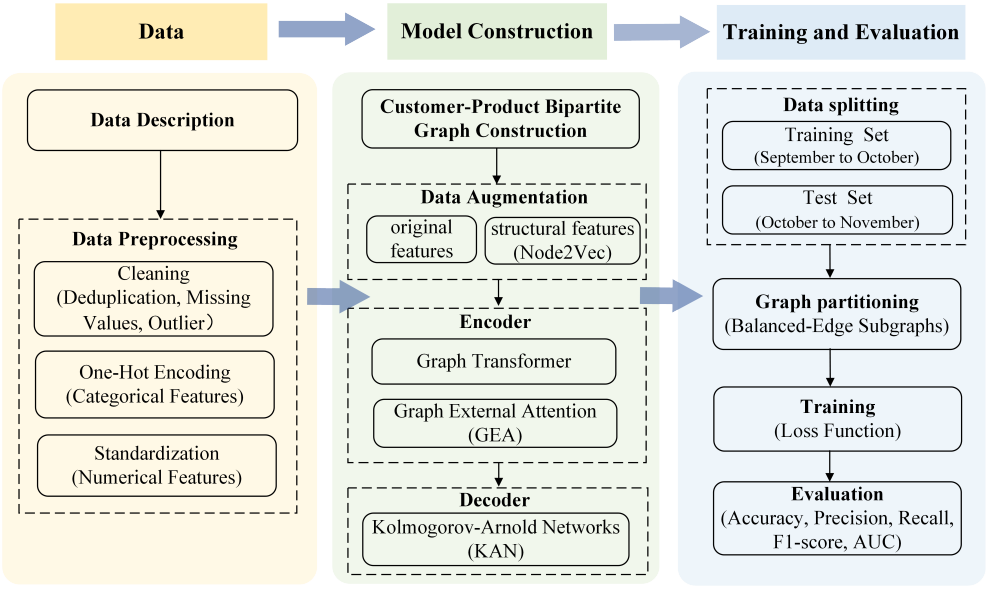


Figure 1. The research flowchart.

3.1. Data and Customer–Product Bipartite Graph Construction

3.1.1. Data Description

Due to the high return rate of clothing products and the significant differences in their customer behaviors, this study utilizes the clothing dataset to construct the customer–product bipartite graph and verify our model. The dataset encompasses records of customer keep and return from September to November 2021, provided by the UK fast fashion e-commerce company ASOS. To prevent data leakage, these customers' keep and return records are segmented by month. The training set encompasses historical data from September to October, while the test set covers records from October to November.

Both the training set and the test set contain three tables: an event table, a customer table, and a product table. These encompass not only approximately 1.8 million customer keep–return events, but also over 1.08 million unique users and more than 300,000 product variants. To protect privacy, all sensitive data is anonymized. We regard customer return prediction as a binary classification task. Returned events are labeled as 1, and kept events are labeled as 0. Each event in the dataset is marked accordingly.

3.1.2. Data Preprocessing

We first checked and preprocessed the data, including identifying duplicates, missing values, and outliers. Then we deleted duplicates and recoded the customer ID and the product ID. Subsequently, the customer's birth year is converted into age to enrich the user portrait. However, about 30,000 customers were over 90 years old, and these outliers were excluded to minimize errors. The final training set consisted of 939,537 events, while the test set contained 858,526 events. To match these events, there are 1,084,504 users and 338,076 product variants. We assessed the sample distribution of the final training and test sets. The ratio of positive to negative samples is approximately 1.2:1, indicating no

significant class imbalance. Additionally, we performed one-hot encoding on categorical variables and standardized numerical variables to facilitate model training.

3.1.3. Customer–Product Bipartite Graph Construction

Customer return behavior does not arise solely from isolated customer or product attributes, but from repeated interactions between customers and products over time. A customer might repeatedly purchase the same item or return multiple products. A product may also be purchased by multiple customers.

To preserve these interaction patterns, we construct a customer–product bipartite graph to represent the connection relationship between customers and items. Since customers and products are two types of entities with different attributes, we define them as two different types of nodes. Each type of node has its own attributes as shown in Table 2.

Table 2. The node attributes of the bipartite graph.

Entity	Type	Attributes
customer	node	customer ID, age, gender, country, membership status, historical purchase volume, historical return volume, user return rate, the proportion of different return reasons
product	node	variant ID, brand, product type, average product price, average discounted product price, product sales volume, product return volume, product return rate, the proportion of different return reasons

The events occurring between customers and products are represented as edges connecting the corresponding nodes. In this study, there are two types of edges: returned edges and kept edges. They are marked as 1 or 0 respectively. This bipartite graph denoted as $G = (U, I, E)$, where U, I, and E represent the sets of user nodes, product nodes, and edges, respectively. As a result, our return prediction task in the bipartite graph is an edge-level classification task. The customer–product bipartite graph is illustrated in Figure 2, which respectively display the specific values of a customer and a product.

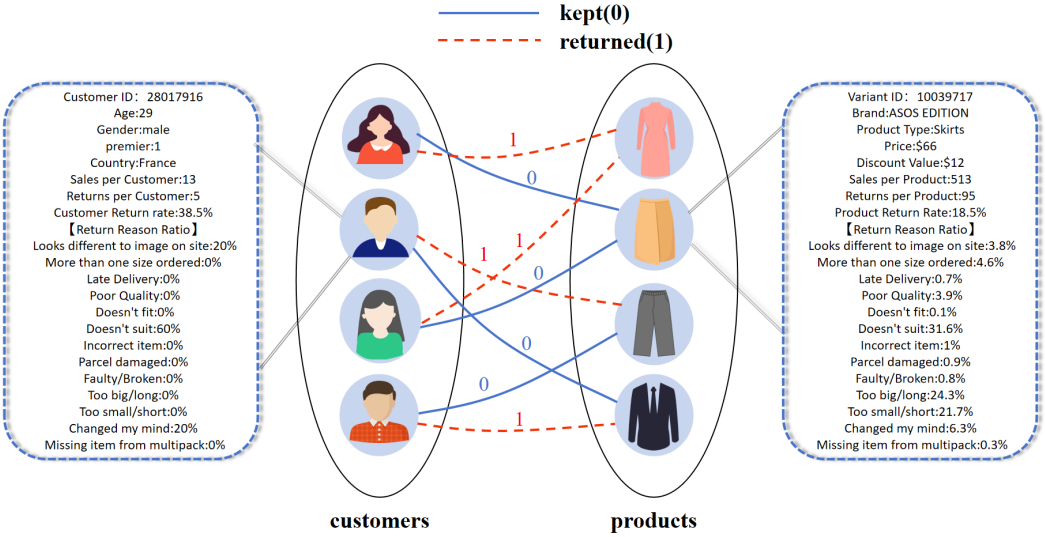


Figure 2. The customer–product bipartite graph.

3.2. Proposed Model

We propose the Returnformer, a return prediction model based on the Graph Transformer. The Returnformer framework is illustrated in Figure 3, which comprises three primary components: data augmentation, encoder, and decoder.

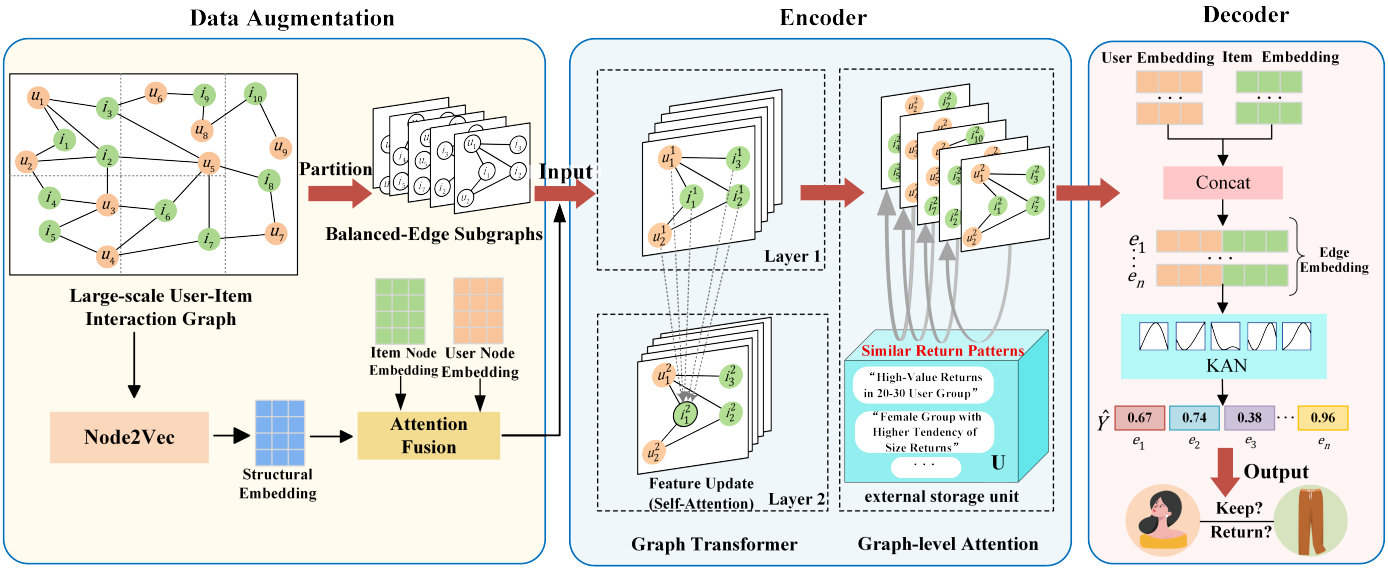


Figure 3. The Returnformer framework.

3.2.1. Data Augmentation

The large-scale user–product interaction graph in e-commerce often comprises millions of nodes and edges. To make computation feasible, the graph is often partitioned into smaller subgraphs. However, this approach may disrupt the overall graph's topological structure. For instance, when connections between high-return-rate products are fragmented across different subgraphs, this may potentially lead to the loss of critical relationships between users and products.

To address this problem, we leverage Node2Vec to obtain the global topological structure of the user–product interaction graph. Node2Vec controls breadth-first (BFS) and depth-first (DFS) search behaviors during random walks by adjusting hyperparameters p and q. It can capture the structural information and encode it into low-dimensional vectors. And the process is shown in Figure 4.

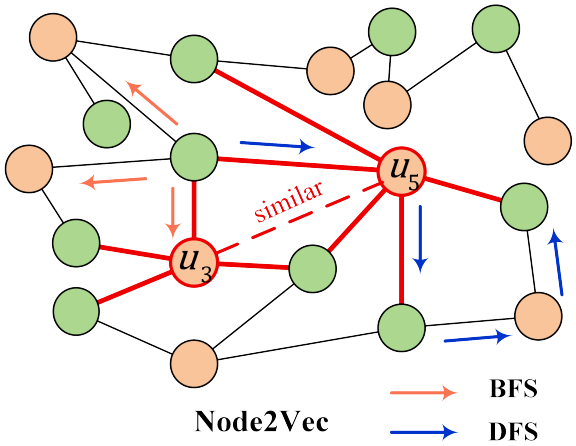


Figure 4. Node2Vec.

Meanwhile, we make use of an attention mechanism to fuse original and structural features dynamically. Unlike static fusion methods, which may obscure important features, our approach adaptively combines features based on their importance. The detailed fusion process is shown in Figure 5.

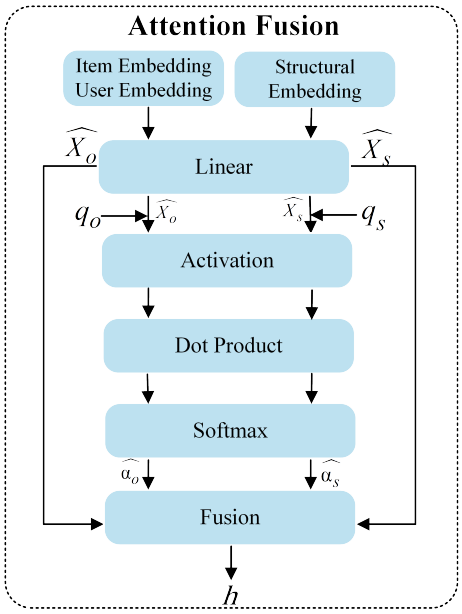


Figure 5. The attention feature fusion mechanism.

Formally, let $X_o, X_s \in \mathbb{R}^{N\times d}$ be the input features, N be the number of nodes, and d be the feature dimension. We first apply linear transformations:

$$\begin{cases} \widehat{X_o} = X_o W_o \\ \widehat{X_s} = X_s W_s \end{cases} \tag{1}$$

where $W_o, W_s \in \mathbb{R}^{d\times d}$ are the learnable parameter. X_o represents the original features, and X_s denotes the structural features generated by Node2Vec.

Next, we compute attention weights for the original and structural features,

$$\begin{cases} \alpha_o = \tanh\left(\widehat{X_o} q_o\right) \\ \alpha_s = \tanh\left(\widehat{X_s} q_s\right) \end{cases} \tag{2}$$

where $q_o, q_s \in \mathbb{R}^d$ are the attention vectors.

Furthermore, following softmax normalization, the attention weights are expressed as

$$\begin{cases} \widehat{\alpha_o} = \frac{\exp(\alpha_o)}{\exp(\alpha_o)+\exp(\alpha_s)} \\ \widehat{\alpha_s} = \frac{\exp(\alpha_s)}{\exp(\alpha_o)+\exp(\alpha_s)} \end{cases} \tag{3}$$

Ultimately, the fused embedding h can be represented as

$$h = \widehat{\alpha_s}\widehat{X_s} + \widehat{\alpha_o}\widehat{X_o} \tag{4}$$

3.2.2. Encoder

The encoder learns representations from graph data by mapping nodes, edges, or the whole graph into a low-dimensional vector space. The encoder of this model primarily adopts a Graph Transformer and incorporates a graph-level attention mechanism to enhance the model's expressive capacity.

(1) Graph Transformer

The Graph Transformer uses self-attention to let each node interact with all other nodes in its subgraph. This captures long-range dependencies and allows user and product information to propagate globally within the current subgraph. The proposed model simplifies the Graph Transformer framework by removing the edge feature processing. Meantime,

the above graph structure is integrated, which makes it more suitable for modeling sparse user–item interactions. Figure 6 illustrates the detailed process of Graph Transformer.

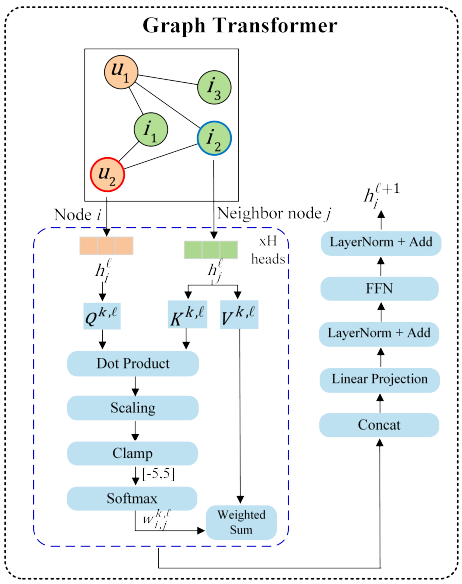


Figure 6. Graph Transformer.

To obtain the node features of the updated $(\ell+1)$-th layer, we need to use Equation (5) to calculate Q, K, and V of each attention head in the ℓ-th layer,

$$\begin{cases} Q_i^{k,\ell} = Q^{k,\ell} h_i^{\ell} \\ K_j^{k,\ell} = K^{k,\ell} h_j^{\ell} \\ V_j^{k,\ell} = V^{k,\ell} h_j^{\ell} \end{cases} \tag{5}$$

where $Q^{k,\ell}, K^{k,\ell}, V^{k,\ell} \in \mathbb{R}^{d_k \times d}$ are the learnable weight matrices, and $\mathrm{h}_i^{\ell}, \mathrm{h}_j^{\ell}$ are the input features of the computing node i and j its neighbor nodes at the ℓ-th layer, respectively. The fusion features h from Equation (4) serves as the initial input of the 0-th layer.

Subsequently, the attention weight of a node i to its neighbor nodes $j \in \mathcal{N}_i$ can be expressed as

$$w_{ij}^{k,\ell} = \mathrm{softmax}\left(\frac{Q_i^{k,\ell} \cdot K_j^{k,\ell}}{\sqrt{d_k}} \right)^T \tag{6}$$

where d_k represents the dimension of each head. To ensure numerical stability, the input of Softmax is clamped to the range of -5 to $+5$.

After obtaining the attention weights for each head, all the matrices are concatenated and processed through the output projection matrix O_h^{ℓ} to obtain the multi-head attention output $\hat{h}_i^{\ell+1}$. This process is implemented by Equation (7),

$$\widehat{h}_i^{\ell+1} = O_h^{\ell} \| _{k=1}^{H} \left(\sum_{j \in N_i} w_{ij}^{k,\ell} V^{k,\ell} h_j^{\ell} \right) \tag{7}$$

where the range of k is from 1 to H, and k represents the number of attention heads, $O_h^{\ell} \in \mathbb{R}^{d \times d}$.

Finally, through two normalizations and residual connections, as well as a two-layer feedforward network, the updated node features $h_i^{\ell+1}$ are obtained as follows,

$$\tilde{h}_i^{\ell} = \mathrm{LayerNorm}(h_i^{\ell}) + \widehat{h}_i^{\ell+1} \tag{8}$$

$$\bar{h}_i^{\ell} = W_2^{\ell} \mathrm{GELU}(W_1^{\ell} \tilde{h}_i^{\ell}) \tag{9}$$

$$h_i^{\ell+1} = \text{LayerNorm}(\tilde{h}_i^{\ell}) + \bar{h}_i^{\ell} \tag{10}$$

where $W_1 \in \mathbb{R}^{d_{\text{ff}} \times d}$, $W_2 \in \mathbb{R}^{d \times d_{\text{ff}}}$ are the weight matrices, and d_{ff} is the hidden dimension of the feed-forward network.

(2) Graph-level Attention Mechanism

We partition the large interaction graph into several subgraphs with the same number of edges. The Graph Transformer encode within a single subgraph, ignoring the potential correlations between subgraphs. For instance, users A and B who belong to different subgraphs may exhibit similar behaviors, such as frequently returning the same type of goods.

To address this, we introduce a graph-level attention mechanism to enhance the Graph Transformer, which called Graph External Attention (GEA). This mechanism integrates information from external graphs into the current graph's representation learning. It enhances the model's ability to capture both local embeddings and global inter-graph correlations. The GEA computes attention weights from node features in the current graph to external key-value units by Equations (11) and (12). Algorithm 1 demonstrates the pseudo-training algorithm of GEA.

Algorithm 1 Framework of the GEA

Input: Graph with node set V, node embeddings X
Output: Updated node representations X_{out}

```
Initialize model parameters: external key-value units U_k, U_v, and number of heads H_ext
for each node ∈ X do
    X_reshaped = Reshape(X, (n, H_ext, d/H_ext))
    Q = X_reshaped U_k^T
    A = DNorm(Q)
    X_attn = A U_v
    X_out = Reshape(X_attn, (n, d))
end for
return X_out
```

The attention matrix A_{ext} is obtained by two steps. First, matrix multiplication on the node embedding matrix $X \in \mathbb{R}^{n \times d}$ and a learnable parameter matrix $U_k \in \mathbb{R}^{m \times d}$ is performed. Then, dual normalization technique $DNorm(\cdot)$ is applied:

$$A_{ext} = DNorm(XU_k^T) \tag{11}$$

where U_k can be regarded as an external key storage unit containing m virtual nodes. The similarity between the current node and the m virtual nodes in the external unit is calculated. When computing the attention matrix A_{ext} for the first time, we use the node features $h_i^{\ell+1}$ updated by Equation (10) as the node feature matrix X.

The dual normalization first computes the similarity $\tilde{S}_{i,j}$ between nodes i and neighbor nodes j through Equation (12).

$$\tilde{S}_{i,j} = (XU^T)_{i,j} \tag{12}$$

Then Equations (13) and (14) are used to normalize each column and each row, respectively, to improve the stability of the values.

$$\widehat{S}_{i,j} = \frac{\exp\left(\widetilde{S}_{i,j}\right)}{\sum_{k=0}^{n} \exp\left(\widetilde{S}_{k,j}\right)} \tag{13}$$

$$\alpha_{i,j} = \frac{\widehat{S}_{i,j}}{\sum_{k=0}^{S} \widehat{S}_{i,k}} \tag{14}$$

The final node features are obtained by multiplying the normalization attention matrix and the external value unit U_v, as demonstrated in Equation (15).

$$X_{out} = A_{ext} U_v \tag{15}$$

3.2.3. Decoder

The decoder uses the encoded embeddings for link prediction. Specifically, the Kolmogorov–Arnold Networks (KAN) is introduced as the decoder. Traditional GNNs typically employ inner products or shallow multilayer perceptrons (MLPs) for decoding. Whereas, these methods not only make it difficult to model complex decision boundaries in user–product interactions, but also result in a substantial number of parameters.

In contrast, the KAN adaptively learns nonlinear feature relationships via B-spline functions. Specifically, each learnable function is obtained by combining different parameters of B-splines. By placing these functions on edges rather than nodes, the KAN captures complex interaction patterns with fewer parameters. Figure 7 illustrates its application in our model.

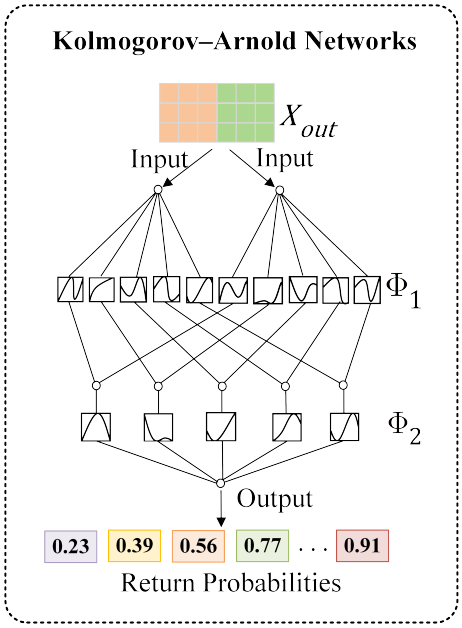


Figure 7. Kolmogorov–Arnold Networks.

Equations (16) and (17) detail MLP, while Equations (18) and (19) describe the KAN. MLP achieves nonlinear mapping by repeatedly applying linear transformations using fixed activation functions, described as follows:

$$\mathrm{MLP}(X_{out}) = (F_L \circ F_{L-1} \circ \cdots \circ F_1)(X_{out}) \tag{16}$$

The transformation of its ℓ-th layer is shown in Equation (17), which consists of the weight matrix W_ℓ, the bias b_ℓ, and the activation function σ.

$$F_\ell(z) = \sigma(W_\ell z + b_\ell) \tag{17}$$

The KAN defines each layer as a trainable univariate function matrix, which can be expressed as

$$\mathrm{KAN}(X_{out}) = (\Phi_L \circ \Phi_{L-1} \circ \cdots \circ \Phi_1)(X_{out}) \tag{18}$$

Φ_ℓ is the function matrix of the ℓ-th layer, as shown in Equation (19). $\phi_{\ell,p,q}$ is the trainable univariate function from the p-th input to the q-th output in the ℓ-th layer,

$$\Phi_\ell = \left\{\phi_{\ell,p,q}\right\}, p = 1, \cdots, M_{in}^{(\ell)}, q = 1, \cdots, M_{out}^{(\ell)} \tag{19}$$

where, $M_{in}^{(\ell)}$ and $M_{out}^{(\ell)}$ are the input dimension and the output dimension in the ℓ-th layer, respectively.

The above demonstrates the specific processes of each part of the Returnformer. We outline the pseudo-training algorithm of Returnformer using Algorithm 2.

Algorithm 2 Framework of Returnformer

Input: User–item bipartite graph $G = (U, I, E)$; user features X_u; item features X_i; Node2Vec embeddings X_{n2v}
Output: Edge-level prediction return probabilities $\hat{Y}$

Initialize model parameters θ randomly
Feature fusion: $X_f = \text{Attention Fusion}(X_u, X_i, X_{n2v})$
Initialize node embeddings: $H^{(0)} = [X_{fu} \| X_{fi}]$
for each edge $\in H^{(0)}$ **do**
  $V_u = X_{fu}[u_{\text{idx}}],\ V_i = X_{fi}[i_{\text{idx}}]$
  $H_u^{l+1}, H_i^{l+1} = \text{Graph Transformer}(V_u, V_i)$
  $H_u^{\text{out}}, H_i^{\text{out}} = \text{GEA}(H_u^{l+1}, H_i^{l+1})$
  $X_{\text{out}} = [H_u^{\text{out}} \| H_i^{\text{out}}]$
  $\hat{Y} = \text{KAN}(X_{\text{out}})$
end for
return $\hat{Y}$

4. Experiments

In this section, we evaluate our proposed model using the processed return data. The experiments are implemented in Python 3.9, utilizing open-source libraries such as sklearn, pandas, and numpy.

4.1. Experimental Environment Setup

We employed the original distribution of the training set and the test set to develop and evaluate our Returnformer. Due to the limitations of the GPU, nearly one million events were divided into multiple batches for training. We set the return probability threshold to 0.5 and use the cross-entropy loss function to quantify the difference between the predicted value and the actual value. The model is optimized utilizing the Adam optimizer and the automation method Optuna, which helps us automatically search for the best hyperparameters and quickly converge to the minimum value of the loss function. Specifically, the search ranges for batch size and embedding dimension are set to [64, 128, 256] and [16, 32, 64, 128, 256], respectively. The learning rate and dropout rate are searched within the ranges of [1×10^{-5}, 1×10^{-3}] and [0.2, 0.5]. The validation set is obtained by randomly sampling 10% from the training set to evaluate different hyperparameters. Finally, the parameter configuration used in this study is presented in Table 3.

Table 3. The parameter settings used in the proposed model.

Parameter	Definition	Setting
p	return parameter in Node2Vec	0.8
q	in-out parameter in Node2Vec	0.8
ℓ	number of layers in the Graph Transformer	2
k	number of attention heads in the Graph Transformer	4
m	number of virtual nodes in the memory unit of the GEA	20

Table 3. *Cont.*

Parameter	Definition	Setting
H_{ext}	number of attention heads in the GEA	1
d	embedding dimension	128
lr	learning rate	2×10^{-5}
$dropout$	dropout rate	0.45
$batch_size$	batch size	128

4.2. Comparison Results

Four machine learning models and three graph neural network models were used as comparison models to assess the performance of Returnformer. These models include Multi-Layer Perceptron (MLP), Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), Categorical Boosting (CatBoost), Graph Convolutional Network (GCN), Graph Attention Network (GAT), and Graph Sample and Aggregate (GraphSAGE). Besides, we employed multiple metrics, including accuracy, precision, recall, F1-score, and area under the Receiver Operating Characteristic curve (AUC), to evaluate the model's performance.

Figure 8 presents the performance metrics for all models on the dataset. The Returnformer surpassed all baselines in all metrics except precision. It achieved a recall of 86.75% and an F1-score of 78.87%, indicating that it has pleasant performance.

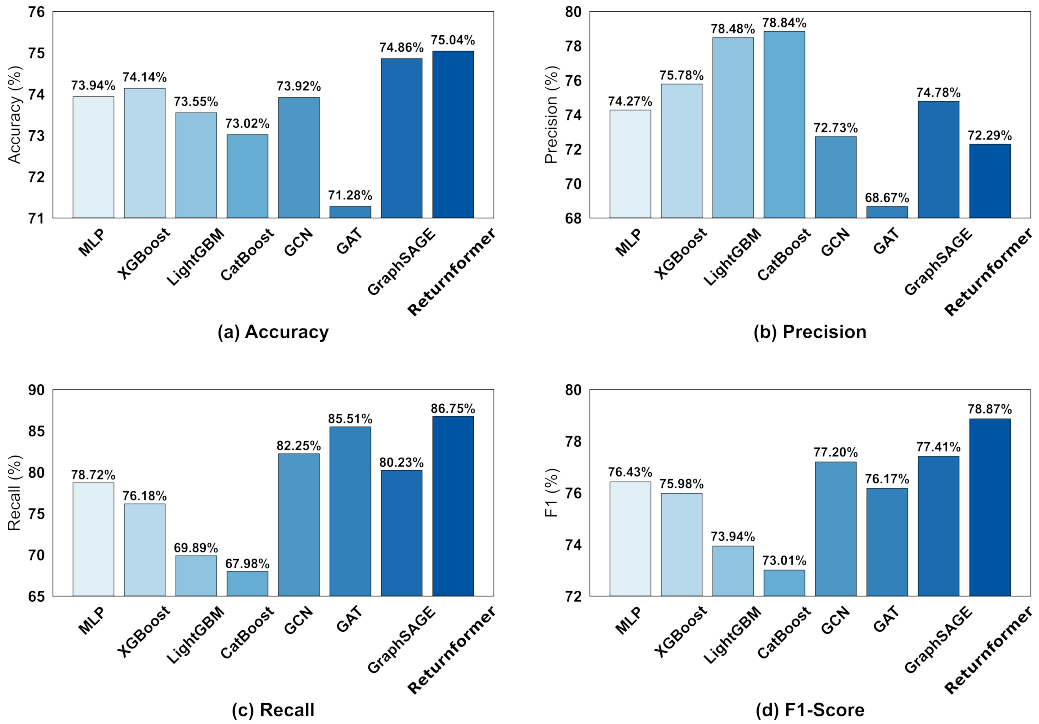


Figure 8. All comparison results on the test set. (**a**) Accuracy. (**b**) Precision. (**c**) Recall. (**d**) F1-score.

The Receiver Operating Characteristic (ROC) curves of all models are displayed in Figure 9. The closer AUC is getting to 1, the stronger the model's classification capability will be. The ROC curve of the Returnformer is closest to the upper left corner, with an AUC of 0.844, indicating that its classification ability surpass that of other models. Moreover, GNNs significantly outperformed machine learning models. Notably, the F1-score for GraphSAGE is 77.41%, which is the highest among all comparison models. This highlights the advantage of the graph representation learning method in forecasting user–item interaction behaviors.

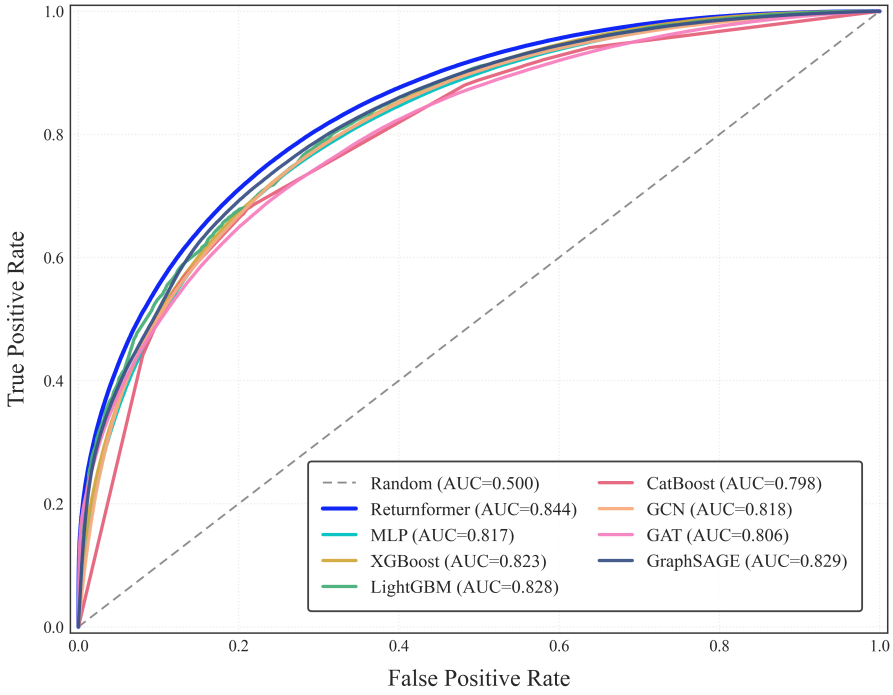


Figure 9. Receiver Operating Characteristic (ROC) curves.

Generally, customers with higher return rates tend to have higher expectations for products and are more likely to abuse the return policy [12]. To further validate the Returnformer's capability in modeling users with high return rates, we conducted an additional experiment on the same training set. However, the scope of the test set is different. It includes only events from users with a return rate of 50% or higher. This specific test set can make a more rigorous assessment of the Returnformer.

As shown in Table 4, the Returnformer achieved an F1-score of 87.63% on this specialized test set, which is 8.76% higher than its F1-score on the entire test set. On the one hand, this displays that the Returnformer can make more accurate predictions about the return behavior of users with high return rates. On the other hand, it demonstrates the ability of the model to predict the return behavior of special users.

Table 4. The comparison results on the test set of users with high return rates (return rates $\geq$ 50%).

Model	Accuracy	Precision	Recall	F1-Score	AUC
Returnformer (all tests)	75.04%	72.29%	86.75%	78.87%	84.42%
Returnformer (high-return customers)	79.23% **(+4.19%)**	80.80% **(+8.51%)**	95.71% **(+8.96%)**	87.63% **(+8.76%)**	78.21%

The bold numbers indicate the improvement in Returnformer's metrics on the high-return customers subset.

4.3. Ablation Analysis

We conducted ablation experiments by systematically removing each component to assess its contribution in the Returnformer after comparative experiments.

(1) *w/o* data augmentation: We eliminated the data augmentation component, meaning that the model only relies on the original feature of nodes without integrating the global structural information. (2) *w/o* graph-level attention: We removed the graph-level attention mechanism that captures the inter-graph correlations. Consequently, after two layers of Graph Transformer encoding, the feature is directly partitioned and concatenated. (3) *w/o*

KAN: We omitted the KAN decoder from the model, replacing it with a single-layer linear classifier for the features after splicing.

The results of ablation experiments are presented in Figure 10. Except for precision, other metrics of Returnformer perform better than those of all simplified versions. Each component plays the essential role in improving the ability to return predictions.

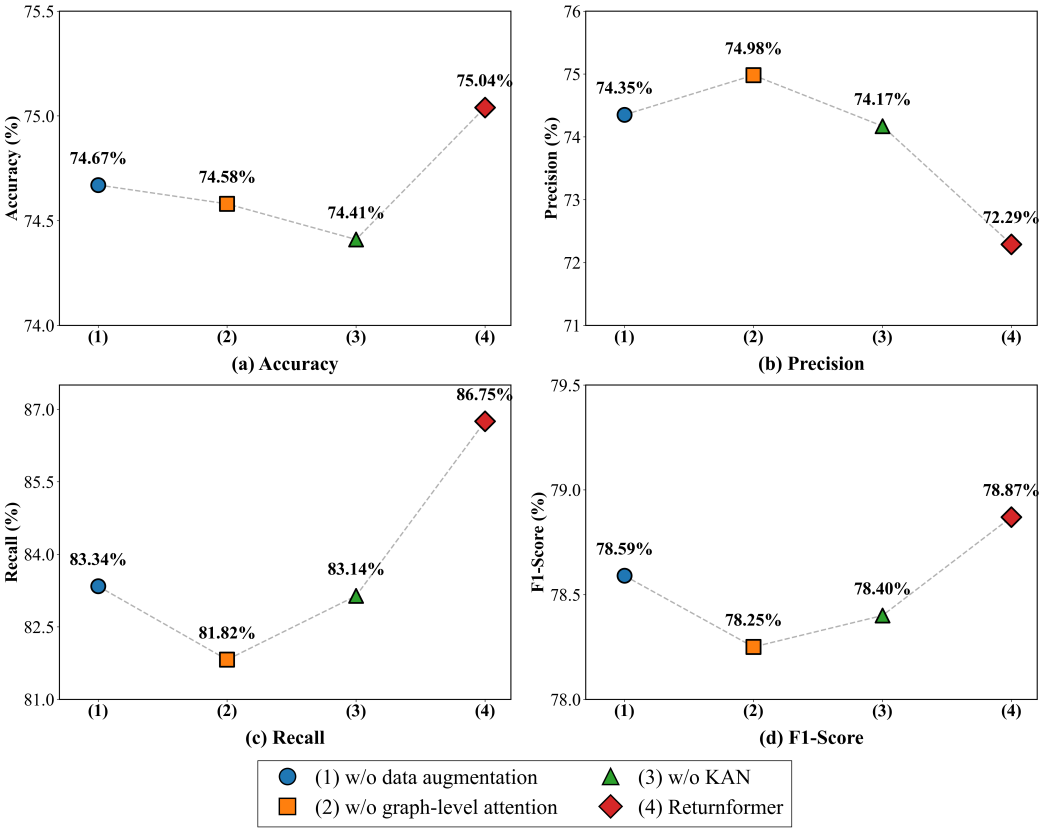


Figure 10. Results of Ablation Experiments.(**a**) Accuracy. (**b**) Precision. (**c**) Recall. (**d**) F1-score.

The data augmentation integrates the global topological structure into the original features, effectively compensating for the structural information lost during graph partition. This enhances the model's representation ability. Additionally, the graph-level attention mechanism automatically learns the potential connections between subgraphs. This mechanism allows the model to learn the return patterns in subgraphs and apply them to the node representation in the new subgraph. Moreover, with its strong nonlinear fitting ability, the KAN accurately models complex decision-making boundaries in user–product interactions, which contributes to the final classification.

4.4. Sensitivity Analysis

To further verify the robustness of the proposed model, we conducted a sensitivity analysis on several key hyperparameters, mainly including the number of layers and the number of attention heads in the Graph Transformer, as well as the number of attention heads in the GEA. These parameters are critical since they are located in the encoding layers and directly control the model's ability to capture high-order information and propagate information across subgraphs.

Specifically, the number of Graph Transformer layers varied from 1 to 4, while the number of attention heads in the Graph Transformer was set to 1, 2, 4, 8. In addition, we evaluated the GEA with 1, 2, 4, and 8 attention heads. For each configuration, the model was trained under the same experimental settings described in Section 4.1, and all

configurations were evaluated using the same test set. The corresponding F1-score and AUC values under different configurations are illustrated in Figure 11.

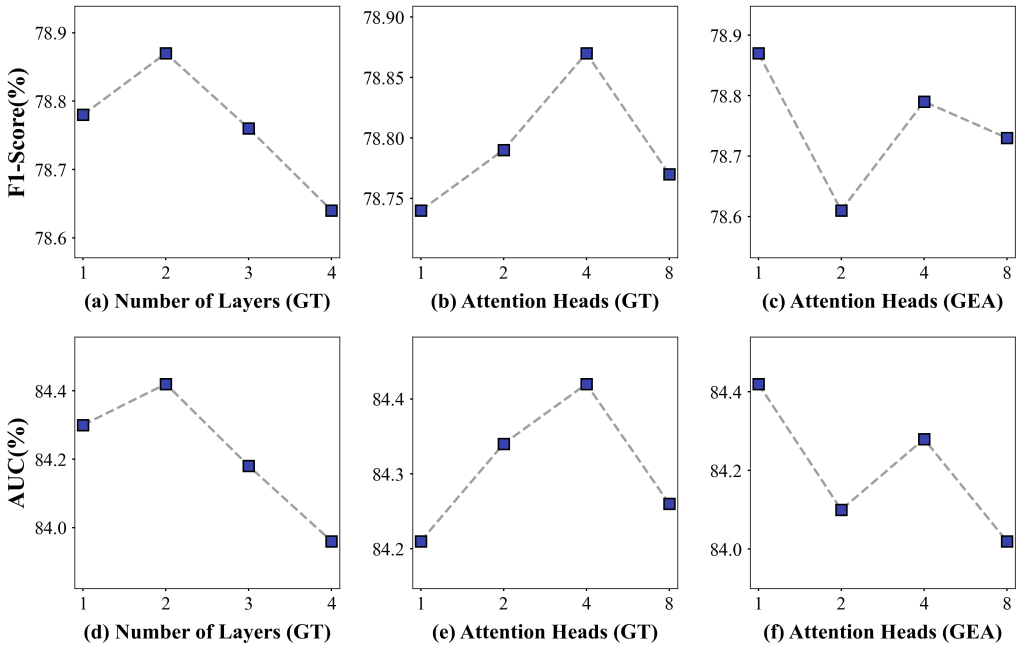


Figure 11. The comparison of F1-score and AUC under Different Configurations. (**a**) F1-Score of Graph Transformer(GT) with different numbers of layers; (**b**) F1-Score of Graph Transformer with different numbers of attention heads; (**c**) AUC of GEA with different numbers of attention heads; (**d**) AUC of Graph Transformer with different numbers of layers; (**e**) AUC of Graph Transformer with different numbers of attention heads; (**f**) F1-Score of GEA with different numbers of attention heads.

It can be observed that the model performs stably across different hyperparameter settings, demonstrating the effectiveness of the proposed model. When the number of Graph Transformer layers exceeds two, the performance gradually degrades, which may be attributed to overfitting. Moreover, when the number of attention heads increases beyond four, redundancy in attention computation may occur. As the number of attention heads in the GEA varies, both the F1-score and AUC first decrease, then increase, and finally decrease again. Overall, this sensitivity analysis confirms that the proposed model is robust to appropriate variations in key hyperparameters. It also suggests that the predictive performance is mainly determined by the model architecture and input features, rather than strongly relying on specific hyperparameter settings.

4.5. Discussion

This study aims to forecast customers' return risk in advance by learning from their historical transaction data. The comparison results in Figure 8 demonstrate that the Returnformer outperforms seven baseline models. Moreover, the experimental results also indicate that, compared to traditional machine learning methods, graph representation learning has stronger learning and prediction capabilities for data with natural graph structures. This is consistent with previous research [13], as graph representation learning is easier to learn complex patterns in the data, especially in this type of user–item interaction graph. The predictive results for users with a high return rate, presented in Table 4, demonstrate that the model we proposed is better suited for such users.

The results of the ablation experiments in Figure 10 also further confirm the validity of each component of the model. The topology structure in the user–item interaction graph may contain predictive information beyond the original node features. Additionally, the

graph-level attention mechanism can adequately catch the commonality of users' return patterns across different subgraphs, making an apparent contribution to return prediction. Figure 11 illustrates that the proposed model maintains stable performance under different hyperparameter settings, indicating good robustness.

Through the analysis of various experiments, we found that the recall of our model is obviously higher than its precision. The Returnformer can perceive consumers' return tendency more comprehensively. Although its precision is not ideal, this may lead the model to misclassify some low-risk customers as high-risk. However, return prediction is a risk-sensitive task, where failing to identify a true return customer generally incurs a higher operational cost than incorrectly labeling a low-risk customer. Moreover, the model outputs a continuous return probability, while the precision and recall reported in our experiments are calculated using a fixed decision threshold of 0.5. To further examine the trade-off between precision and recall of the proposed model, we plot the precision-recall(PR) curves of all models in Figure 12. The PR curves illustrate the performance of different models in terms of precision and recall across varying decision thresholds.

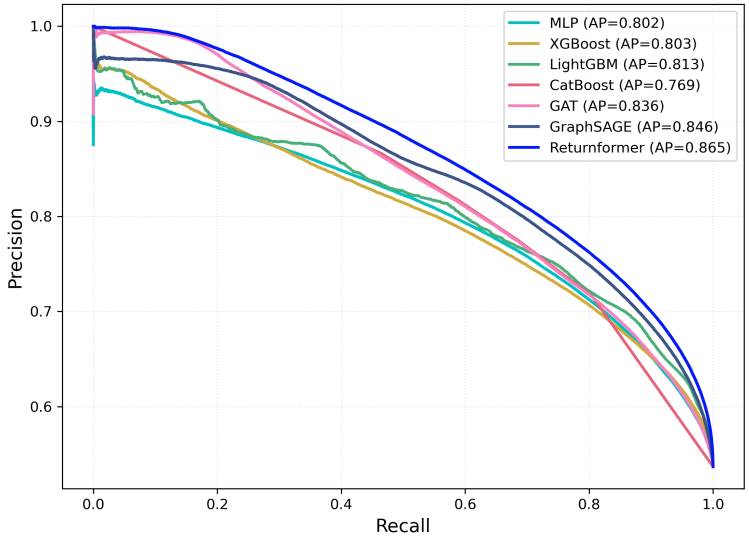


Figure 12. Precision–Recall(PR) Curves on the Test Set.

As shown in Figure 12, compared with other baseline models, the proposed Returnformer achieved higher precision at the same recall level. The Average Precision (AP) of our model reach 0.865, indicating that the model prioritizes recall under the default threshold and aims to capture users with potential return risk as comprehensively as possible. In practical e-commerce scenarios, return prediction models are typically used as decision-support tools rather than directly triggering punitive actions for all high-risk users. Therefore, retailers can flexibly adjust the decision threshold according to their tolerance for misclassification. By selecting an appropriate decision threshold, retailers can introduce preventive measures for users with a high predicted return risk, such as displaying reminder pop-ups or providing enhanced size recommendation services. At the same time, the implementation of these preventive strategies can also help retailers identify and filter users who tend to abuse return policies.

5. Conclusions and Future Works

This paper proposed a Graph Transformer-based return prediction model, Returnformer, which forecasts customers' return tendencies before their payment. Based on the ASOS clothing dataset, our model achieved a recall of 86.75% and an AUC of 84.42%. In comparison with previous research methods, the Returnformer leverages the structural

information of the user–item interaction graph and utilizes a graph-level attention mechanism to capture and disseminate global return patterns. As a result, the model exhibits robust performance in predicting return risk at the level of individual product variants, providing retailers with a tool for early warning of return risks.

Retailers can adopt proactive strategies based on forecasting models, such as personalized information reminders and adjusting freight. Retailers can also provide personalized informational prompts to users with a high predicted return risk, such as size recommendations and more detailed product information. Adjusting the freight will also prompt customers with a high return rate to evaluate the necessity of their purchase behavior more carefully. Meanwhile, the model also helps optimize the supply chain and improve profitability.

Although this study offers the advantages mentioned above, it still presents limitations. The model primarily relies on the keep–return events of users and products, as well as basic attributes of users and products. However, other factors may also influence return behaviors, such as product reviews, seasonal factors, and promotions. In addition, model predictions are usually based on learning from static historical data, yet user preferences and product popularity are dynamic. Applying real-time data to return prediction models remains a challenge. Future research will focus on developing a return prediction model that incorporates dynamic data learning and enhances the model's interpretability.

Author Contributions: Conceptualization, Q.C. and N.Z.; methodology, Q.C.; software, N.Z.; validation, Q.C. and N.Z.; investigation, H.L.; resources, Q.C.; writing—original draft preparation, Q.C. and N.Z.; writing—review and editing, Q.C. and H.L.; project administration, Q.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original data used in this study are openly available in the Open Science Framework (OSF) repository at https://osf.io/c793h/overview (accessed on 17 March 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

GNN	Graph Neural Network
GCN	Graph Convolutional Network
GAT	Graph Attention Network
GraphSAGE	Graph Sample and Aggregate
MLP	Multi-Layer Perceptron
XGBoost	Extreme Gradient Boosting
KAN	Kolmogorov–Arnold Networks
GEA	Graph External Attention
AUC	Area Under the Receiver Operating Characteristic Curve
ROC	Receiver Operating Characteristic

References

1. Statista. Global Retail e-Commerce Sales 2014–2027. Available online: https://www.statista.com/statistics/worldwide-retail-e-commerce-sales (accessed on 20 September 2025).
2. Röllecke, F.J.; Huchzermeier, A.; Schröder, D. Returning customers: The hidden strategic opportunity of returns management. *Calif. Manag. Rev.* **2018**, *60*, 176–203.
3. de Araújo, A.C.; Matsuoka, E.M.; Ung, J.E.; Massote, A.; Sampaio, M. An exploratory study on the returns management process in an online retailer. *Int. J. Logist. Res. Appl.* **2018**, *21*, 345–362.
4. Zhao, S.; Wu, F.; Jia, T.; Shu, L. The impact of product returns on price and delivery time competition in online retailing. *Comput. Ind. Eng.* **2018**, *125*, 658–667. [CrossRef]
5. Dzyabura, D.; El Kihal, S.; Hauser, J.R.; Ibragimov, M. Leveraging the power of images in managing product return rates. *Mark. Sci.* **2023**, *42*, 1125–1142. [CrossRef]

6. Roichman, R.; Sprecher, B.; Blass, V.; Meshulam, T.; Makov, T. The convenience economy: Product flows and GHG emissions of returned apparel in the EU. *Resour. Conserv. Recycl.* **2024**, *210*, 107811. [CrossRef]
7. Mishra, A.; Dutta, P. Return management in e-commerce firms: A machine learning approach to predict product returns and examine variables influencing returns. *J. Clean. Prod.* **2024**, *477*, 143802. [CrossRef]
8. Al Imran, A.; Amin, M.N. Predicting the return of orders in the e-tail industry accompanying with model interpretation. *Procedia Comput. Sci.* **2020**, *176*, 1170–1179. [CrossRef]
9. Bhattacharjee, B.; Unni, K.; Pratap, M. Bayesian-optimized extreme gradient boosting models for classification problems: An experimental analysis of product return case. *J. Syst. Inf. Technol.* **2024**, *26*, 495–527. [CrossRef]
10. Li, J.; He, J.; Zhu, Y. E-tail product return prediction via hypergraph-based local graph cut. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London, UK, 19–23 August 2018; pp. 519–527.
11. Zhu, Y.; Li, J.; He, J.; Quanz, B.L.; Deshpande, A.A. A local algorithm for product return prediction in e-commerce. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden, 13–19 July 2018; pp. 3718–3724.
12. Li, X.; Zhuang, Y.; Fu, Y.; He, X. A trust-aware random walk model for return propensity estimation and consumer anomaly scoring in online shopping. *Sci. China Inf. Sci.* **2019**, *62*, 052101. [CrossRef]
13. McGowan, J.; Guest, E.; Yan, Z.; Zheng, C.; Patel, N.; Cusack, M.; Donaldson, C.; de Cnudde, S.; Facini, G.; Dzogang, F. A dataset for learning graph representations to predict customer returns in fashion retail. In Proceedings of the Workshop on Recommender Systems in Fashion and Retail, Seattle, WA, USA, 18–22 September 2022; pp. 99–108.
14. Lv, J.; Liu, X. The impact of information overload of e-commerce platform on consumer return intention: Considering the moderating role of perceived environmental effectiveness. *Int. J. Environ. Res. Public Health* **2022**, *19*, 8060. [PubMed]
15. Fernandez-Lores, S.; Crespo-Tejero, N.; Fernández-Hernández, R.; García-Muiña, F.E. Online product returns: The role of perceived environmental efficacy and post-purchase entrepreneurial cognitive dissonance. *J. Bus. Res.* **2024**, *174*, 114462. [CrossRef]
16. Ambilkar, P.; Dohale, V.; Gunasekaran, A.; Bilolikar, V. Product returns management: A comprehensive review and future research agenda. *Int. J. Prod. Res.* **2022**, *60*, 3920–3944. [CrossRef]
17. Prayogo, D.H.; Domanski, R.; Golinska-Dawson, P. The key factors for improving returns management in e-commerce in Indonesia from customers' perspectives—An analytic hierarchy process approach. *Sustainability* **2024**, *16*, 7303.
18. Li, Y.; Xu, L.; Li, D. Examining relationships between the return policy, product quality, and pricing strategy in online direct selling. *Int. J. Prod. Econ.* **2013**, *144*, 451–460. [CrossRef]
19. Yoga, I.A.M.; Budiarsa, M.; Ardika, I.W.; Wiranatha, A.A.P.; Suryawan, A. Improving the marketing strategy into 7PS+ for sustainability: A study of Bali tourism, Indonesia. *Russ. J. Agric. Socio-Econ. Sci.* **2020**, *104*, 15–23.
20. Walsh, G.; Albrecht, A.K.; Kunz, W.; Hofacker, C.F. Relationship between online retailers' reputation and product returns. *Br. J. Manag.* **2016**, *27*, 3–20.
21. Saarijärvi, H.; Sutinen, U.-M.; Harris, L.C. Uncovering consumers' returning behaviour: A study of fashion e-commerce. *Int. Rev. Retail Distrib. Consum. Res.* **2017**, *27*, 284–299. [CrossRef]
22. Masuch, S.; Landwehr, J.R.; Flath, C.M.; Thiesse, F. The faster, the better? The impact of short delivery times on product returns in online retailing. *J. Retail.* **2024**, *100*, 475–485. [CrossRef]
23. Yan, S.; Xu, X.; Bian, Y. Pricing and return strategy: Whether to adopt a cross-channel return option? *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 5058–5073.
24. Rajasekaran, V.; Priyadarshini, R. An e-commerce prototype for predicting the product return phenomenon using optimization and regression techniques. In Proceedings of the International Conference on Advances in Computing and Data Sciences, Dehradun, India, 20–21 April 2021; pp. 230–240.
25. Cui, H.; Rajagopalan, S.; Ward, A.R. Predicting product return volume using machine learning methods. *Eur. J. Oper. Res.* **2020**, *281*, 612–627. [CrossRef]
26. Tripathy, S.; Kumar, A.; Mahanty, B. Short-lived product returns forecasting when customers are unwilling to return the product: A grey-graphical evaluation and review technique. *Technol. Forecast. Soc. Change* **2023**, *195*, 122755. [CrossRef]
27. Ma, J.; Kim, H.M. Predictive model selection for forecasting product returns. *J. Mech. Des.* **2016**, *138*, 054501. [CrossRef]
28. Chou, M.C.; Sim, C.-K.; Yuan, X.-M. Policies for inventory models with product returns forecast from past demands and past sales. *Ann. Oper. Res.* **2020**, *288*, 137–180. [CrossRef]
29. Fu, Y.; Liu, G.; Papadimitriou, S.; Xiong, H.; Li, X.; Chen, G. Fused latent models for assessing product return propensity in online commerce. *Decis. Support Syst.* **2016**, *91*, 77–88. [CrossRef]
30. Urbanke, P.; Kranz, J.; Kolbe, L. Predicting product returns in e-commerce: The contribution of Mahalanobis feature extraction. *Electron. Mark.* **2015**, *25*, 353–368.
31. Duong, Q.H.; Zhou, L.; Van Nguyen, T.; Meng, M. Understanding and predicting online product return behavior: An interpretable machine learning approach. *Int. J. Prod. Econ.* **2025**, *280*, 109499. [CrossRef]

32. Hofmann, A.; Gwinner, F.; Fuchs, K.; Winkelmann, A. An industry-agnostic approach for the prediction of return shipments. In Proceedings of the Americas Conference on Information Systems (AMCIS), Salt Lake City, UT, USA, 15–17 August 2020.
33. Ma, S.; Wang, W. Proactive return prediction in online fashion retail using heterogeneous graph neural networks. *Electronics* **2024**, *13*, 1398. [CrossRef]
34. Joshi, T.; Mukherjee, A.; Ippadi, G. One size does not fit all: Predicting product returns in e-commerce platforms. In Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Barcelona, Spain, 28–31 August 2018; pp. 926–927.
35. Kedia, S.; Madan, M.; Borar, S. Early bird catches the worm: Predicting returns even before purchase in fashion e-commerce. *arXiv* **2019**, arXiv:1906.12128. [CrossRef]
36. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In Proceedings of the 30th Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016.
37. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
38. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph attention networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
39. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. In Proceedings of the 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
40. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How powerful are graph neural networks? In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
41. Dwivedi, V.P.; Bresson, X. A generalization of transformer networks to graphs. *arXiv* **2020**, arXiv:2012.09699.
42. Liang, J.; Chen, M.; Liang, J. Graph external attention enhanced transformer. In Proceedings of the International Conference on Machine Learning, Vienna, Austria, 21–27 July 2024; pp. 29560–29574.
43. Liu, Z.; Wang, Y.; Vaidya, S.; Ruehle, F.; Halverson, J.; Soljačić, M.; Hou, T.Y.; Tegmark, M. KAN: Kolmogorov–Arnold networks. In Proceedings of the International Conference on Learning Representations, Singapore, 22–26 April 2025.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.