ColdGuess: A General and Effective Relational Graph Convolutional Network to Tackle Cold Start Cases

Bo He
bhe@amazon.com
Amazon Inc.
US

Xiang Song
xiangsx@amazon.com
AWS AI Education and Research
US

Vincent Gao
vincegao@amazon.com
Amazon Inc.
US

Christos Faloutsos*
faloutso@amazon.com
AWS AI Education and Research
US

ABSTRACT

Low-quality listings and bad actor behavior in online retail websites threatens e-commerce business as these result in sub-optimal buying experience and erode customer trust. When a new listing is created, how to tell it has good quality? Is the method effective, fast, and scalable? Previous approaches often face three limitations/challenges: (1) unable to handle cold start problems where new sellers/listings lack sufficient selling histories. (2) inability of scoring hundreds of millions of listings at scale, or compromise performance for scalability. (3) has space challenge from large-scale graph built on giant business size. To overcome these limitations, we proposed ColdGuess, an inductive graph-based risk detector built upon a heterogeneous seller-product graph, which effectively identifies risky seller/product /listings at scale. ColdGuess tackles the large-scale graph by *consolidated nodes*, and addresses the cold start problem using *homogeneous influence*[1]. The evaluation on real data demonstrates that ColdGuess has stable performance as the number of unknown features increases. It outperforms the lightgbm[2], a commonly used risk detection model in production, by *up to 34 pcp* ROC AUC in the cold start case when a new seller sells a new product . The resulting system, ColdGuess, is effective, adaptable to changing bad actor behavior, and is already *in production*. This paper belongs to "Application and analysis – Large-scale graph and modeling", and in the "Novel research paper" category.

KEYWORDS

Graph Neural Networks, Cold Start Problem, Network Inference

ACM Reference Format:
Bo He, Xiang Song, Vincent Gao, and Christos Faloutsos. 2022. ColdGuess: A General and Effective Relational Graph Convolutional Network to Tackle Cold Start Cases . In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Detecting low-quality listings and bad actor behavior is an important task for e-commerce websites [3, 9]. When a new product arrives, how to tell it is good-quality? When a bad actor creates a new account, how to detect it before any fraudulent behavior endangers customers shopping experience. Is the detection mechanism fast and scalable? Can it score hundreds of millions of listings at scale? When a new seller is associated with multiple existing seller accounts through either weak or strong relations, can this method tell which relation to emphasize? These are the research problems we focus on in this work.

Traditional machine learning models like logistic regression, support vector machine[12, 16], and boosting trees [11] are widely used in risk detection. However, they often face following limitation/challenges:

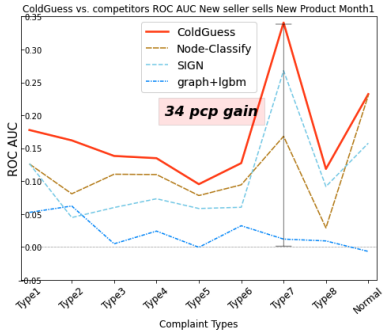
- unable to model new sellers/offers (*the Cold Start Problem*). New sellers/offers lack sufficient histories for traditional models to identify as risky. As large e-commerce companies continuously expand to new countries, the cold start problem becomes more common and critical.
- inability of scoring hundreds of millions of listings at scale, or compromise performance for scalability.
- non-graph based methods (e.g. logistic regression, boosting tree, etc.) are not able to leverage seller-seller and seller-product [3]linkage information[4]. Linkage information has proven to be valuable as risky sellers/offers are often found clustered.[5] Graph-based method face the challenges of memory space when applied to large-scale graph with giant e-commerce business size.

To overcome these limitations, we proposed ColdGuess, a graph neural network based risk detector which leverages the seller-seller and seller-product linkage information to identify risky seller/product /listings at scale. It handles the large-scale graph

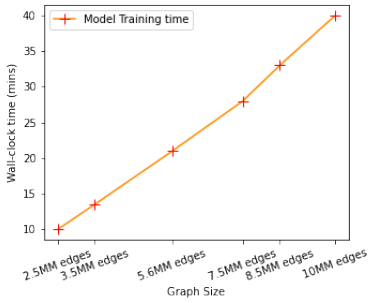
*On leave from CMU
[1]See definition of *consolidated nodes* and *homogeneous influence* in Section 4.3
[2]lightgbm is abbreviated as lgbm in the paper.

[3]For example, Seller Smith sells Nike shoes. Here Smith is a seller, Nike shoes is a product. The Nike shoes sold by Smith is an offer listing. In this paper, we use offer, listings, and offer listings interchangeably.
[4]e.g., sellers are associated if they share the same information or similar characteristics.
[5]For example, majority of sellers who share the same information with a risky seller are also found risky.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456



(a) ColdGuess wins all benchmark models in the cold start case when new sellers sell new products .



(b) ColdGuess scales linearly: model training time vs graph size.

Figure 1: ColdGuess's performance and scalability.

by *consolidated nodes*[6], and addresses the cold start problem by *homogeneous influence*[7] and message passing from existing sellers/products to new sellers/products . As shown in Table 1, there are many competitors, but none of them has all the features that ColdGuess offers.

We evaluated ColdGuess and compared it with four competitors[8] on a full spectrum of sample data and three simulated cold start scenarios (i.e. new offers, new sellers, and new sellers who sell new product). We found that ColdGuess wins with increasing margins as the severity of cold start cases increases. As shown in Figure 1, ColdGuess outperforms lgbm by up to **34 percentage points (pcp) in AUC** in the extreme cold start scenario when a new seller sells a new product .

The main contributions and advantages of ColdGuess are:

- **General:** The inductive ColdGuess framework can handle (a) *Dynamic heterogeneous graphs built on highly relational data* with tens of millions of nodes, hundreds of millions of edges, and massive node and edge features. (b) *Missing values in edge feature.* In previous approaches to graph modeling, missing edge features cannot be inferred from their neighbors since propagation only happens on the nodes. To deal with it, ColdGuess augmented edge features by concatenating it with its neighboring edges.

[6]See definition in Section 4.3
[7]See definition in Section 4.3
[8]MetaHIN is not used as it is designed for cold start recommendation. EGNN and CensNet are not used as they do not work with heterogeneous graphs.

Table 1: ColdGuess matches all specs, while competitors miss one or more of the features.

Property \ Method	lgbm [11]	SIGN[4]	RGCN[19]	GAT[21]	MetaHIN[15]	EGNN[5]	CensNet[10]	ColdGuess-Naive	ColdGuess
Node connectivity		√	√	√	√	√	√	√	√
Heterogeneous info			√		√				√
Edge features						√	√		√
Cold start					√		√		√

- **Effective**, especially for *Cold Start Problem*: ColdGuess achieves stable performance with increasing number of missing features in cold start scenarios. It outperforms lgbm by up to *34 pcp* AUC in the extreme cold start case where a new seller sells a new product .
- **Fast and Scalable:** ColdGuess can scale to a business size graph with hundreds of millions of edges in inference. It takes only 2 minutes to make 1MM+ predictions. The training on a graph with 10MM+ edges takes 40 minutes on a p3.8xlarge EC2 instance equipped with 4 V100 GPU and 224GB memory. ColdGuess scales linearly with the input size in both training and inference, as shown in Figure 6.

2 PRELIMINARIES

Notations. The notations used in the following sections are listed in Table 2.

Graph. A graph is composed of nodes and edges with relations defining the heterogeneity $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, $\mathcal{R}$ is the set of relations. Each edge $e \in \mathcal{E}$ belongs to only one relation $r \in \mathcal{R}$. Let $\mathcal{X}$ be the node feature space such that $\mathcal{X} = (x_1, \ldots, x_N)^T$ where x_i represents the node feature of v_i. Let $\mathcal{Y}$ be the edge feature space such that $\mathcal{Y} = (y_1, \ldots, y_M)^T$ where y_i represents the edge feature of e_i. Let $\mathcal{Z}$ be the label space.

Graph Convolutional Network. GNNs [18, 24] are a series of multi-layer feed-forward neural networks that propagate and transform layer-wise features. Among these models, relational graph convolutional network (RGCN) [19] was designed to operate on large-scale heterogeneous graphs.

3 RELATED WORK

Cold Start. Cold start problem gains increasing interest in last few years, and was widely researched in recommendation systems [2, 13, 14, 25]. Most recent works [8, 15, 22] leverage heterogeneous graph to capture richer semantics via higher-order graph structures and meta information. Hu, Binbin et al. [8] showed that side information such as meta-path based context is useful to alleviate the cold start problem. Lu, Yuanfu et al. [15] leveraged meta-learning to address cold-start recommendation. However, unlike user-item recommendation scenarios where users do not directly connect with other users, there are rich linkage information between sellers in a seller-product graph in e-commerce. Furthermore,

Table 2: Notation and descriptions.

$\mathcal{G}$	A graph.
$\mathcal{V}$	The set of nodes.
$\mathcal{E}$	The set of edges.
$\mathcal{R}$	The set of relations.
$\mathcal{X}$	The node feature space where x_i represents the node feature of $v_i \in \mathcal{V}$.
$\mathcal{Y}$	The edge feature space where y_i represents the edge feature of $e_i \in \mathcal{E}$.
$\mathcal{H}$	The node embedding matrix where h_v represents the embedding vector of v.
$\mathcal{Z}$	The defect type label space where z_i represents the i-th label vector.
$\mathcal{S}$	The set of nodes $v \in \mathcal{V}$ with the node type of Seller.
$\mathcal{O}$	The set of edges $e \in \mathcal{E}$ with the edge type of (Seller, Offer, product).
$\mathcal{P}$	The set of nodes $v \in \mathcal{V}$ with the node type of product .
l	An offer listing $\langle s, o, p \rangle$ with $s \in \mathcal{S}$, $o \in \mathcal{O}$ and $p \in \mathcal{P}$.
L	A set of offer listings.
$\mathbf{s}, \mathbf{p}, \mathbf{o}$	The feature vectors of seller node s, product node p and offer edge o.

unlike edges between users and items which contain little information [1, 7], the listing edges in a seller-product graph usually contain rich information on its selling histories. Our proposed ColdGuess leverages such linkage information to address cold start problem. Other than only using GNN, ColdGuess uses *homogeneous influence* to further boost the performance and *consolidated nodes* to handle the scalability issue.

4 METHOD

4.1 Problem Definition

We use customer complaint types as labels for low-quality listings. There are eight types of complaints. We formulate this risky listing detection as a multi-label classification problem because one listing could have multiple types of complaints.

Problem 1. *Given (1) labelled listings $\mathcal{L}$ in 1 out of k categories, (2) the numerical and categorical features for sellers, products , and offers denoted as* $\mathbf{s}, \mathbf{p}, \mathbf{o}$ *respectively, and (3) Seller-Product graph $\mathcal{G}$, find a classifier f which maximizes the likelihood of offer listing l having complaint type z:*

$$\hat{z} = \text{argmax}\, f(z|s, p, o, \mathcal{G}) \tag{1}$$

In addition to Problem 1, the classifier should be efficient to handle *large-scale graphs* and work effectively with *cold start problems*. We summarize the cold start problems into three cases, and illustrate them in Figure 2.

- **New offer case**: An existing seller creates a new offer under an existing product . Both seller and product nodes of that new offer exist in the Seller-product graph. The offer edge is newly added to the graph, and has limited history information.
- **New seller case**: An new seller creates a new offer under an existing product . Only the product node of the new offer exists in the Seller-Product graph. The seller node and offer edge are newly added, and have limited seller and offer information.

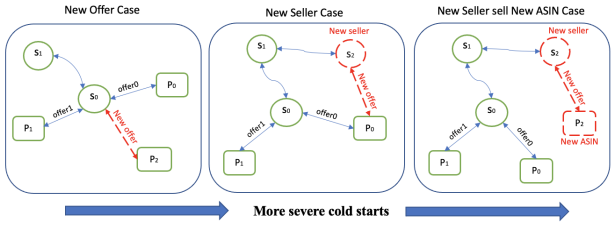


Figure 2: Three cold start scenarios: new offer case (left), new seller case (middle), new sellers sell new products case (right).

- **New sellers sell new products case**: A new seller creates a new product which was never sold on Amazon. Neither of the seller nor product nodes are related to any existing listings on the Seller-product graph. But the seller can be related to an existing seller[9]. Therefore, limited information is known about the seller, the product , and the offer.

In the last two cases, new sellers can be connected with existing sellers if they possess associated information. The severity of cold start increases from new offer case to new seller case, and to new seller sells a new product case.

4.2 Seller-Product Graph

We constructed Seller-Product graph as a heterogeneous graph with seller and product nodes. Within the graph, sellers are connected if they possess associated information. A seller is connected with an product if the seller sells that product . The resulting graph $\mathcal{G}$ has two node types and nine relations (i.e. eight seller-seller relations plus a seller-product relation (i.e. offer edge)), as well as massive features on seller nodes, product nodes, and offer edges. The product and seller level features are encoded as the node feature vectors. The offer level features are encoded as edges feature vectors. Product node features include product type, product 's selling history, etc. Seller node features include seller's orders, selling

[9]For example, a seller used to sell on European market start her business on US market. Then her US seller account will be associated with her European seller account.

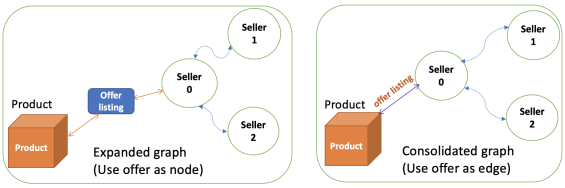


Figure 3: A toy example of an expanded graph with offers as nodes (left) vs. a consolidated graph with offers as edges (right)

history, etc. Offer edge features include list price, orders, offer's selling history, etc.

We call this graph design as *consolidated nodes* because we treat offers as edges instead of nodes. It reduces the number of edges by half as compared to using offers as nodes.

Why the 'consolidated node' proposal? Alternatively, we can construct the graph by using offers as nodes (Figure 3 right). An offer node connects a seller and a product nodes. To predict the complaint type of an offer can leverage the RGCN node classification framework. We evaluated this alternative set up on 4 test sets over 4 months' dataset, one for each month, and found: 1) ColdGuess consistently outperforms in at least 7 out of 9 classes when being evaluated on the full spectrum of data and cold start cases. It is superior to RGCN benchmark with increasing margins as the severity of cold-start cases grows.2) ColdGuess saves GPU memory space by 65% and computation time by 15% when both methods are evaluated on a 4-GPU P3.8xlarge EC2. The *consolidated nodes* design reduces the graph size without losing offer information as ColdGuess extracted offer information via an edge embedder[10].

4.3 Proposed Method: ColdGuess

ColdGuess contains three modules: 1) A node embedder built on a 3-layer RGCN which encodes raw seller and product node features into two embedding vectors emb_s and emb_p. 2) An edge embedder built on a 2 layer multi-linear perceptron (MLP) which encodes offer edge features $\mathbf{o}$ into an embedding vector emb_o 3) A final MLP classifier which concatenates the seller, product , and offer embedding vectors ($emb_s \| emb_p \| emb_o$) and predicts the product complaint type. The overall model architecture is shown in Figure 4.

Module 1: Node Embedder. It creates the seller and product embedding (emb_s and emb_p) using RGCN [19]. Since seller and product node features have different dimensions and meanings, we projected them into the same embedding space before feeding them into the graph convolution.

Due to the large scale of Seller-Product graph, we take a graph-based mini-batch approach [6] for propagation. We sampled a batch of 1024 offer edges and extracted the corresponding ego network which covers the neighbors 3-hops away from the source edges, and fed this mini-batch into the 3-layer RGCN module.

Module 2: Edge Embedder. It creates offer edge embedding (emb_o) via a MLP model. It takes an offer's edge features and its neighboring edge features as input. Offer edges usually have missing values,

[10]See Section 4.3 Module 2: Edge Embedder

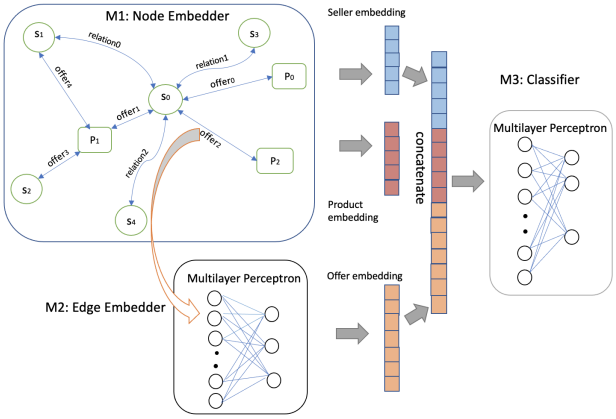


Figure 4: The overall architecture of ColdGuess

especially in cold start cases. We enrich the offer feature vector $\mathbf{o}_o$ by concatenating it and its neighbors. The neighboring edge is defined as edges connected to the offer's head and tail node (i.e. product and seller nodes). We create an offer feature summary for both the product node and the seller node as:

$$\begin{aligned} \mathbf{o}_p &= \psi(\mathbf{o}_1, \ldots, \mathbf{o}_n), \text{where } o_i \in \mathcal{N}_p \\ \mathbf{o}_s &= \psi(\mathbf{o}_1, \ldots, \mathbf{o}_n), \text{where } o_i \in \mathcal{N}_s \end{aligned} \tag{2}$$

where ψ is a aggregation function such as mean, max-pool, etc. [11], o_i are offer features of edge i, $\mathcal{N}_p$ and $\mathcal{N}_s$ are the sets of an offer's neighboring edges connected to its product and seller nodes respectively, $\mathbf{o}_p$ and $\mathbf{o}_s$ are the summary of feature vectors of offers connected to product and seller nodes. The final offer feature vector is defined as $\mathbf{o}' = \mathbf{o}_o \| \mathbf{o}_p \| \mathbf{o}_s$, where $\|$ denotes the concatenation operation. Eventually, $\mathbf{o}'$ is fed into the MLP classifier to generate the offer embedding emb_o.

The concatenation of neighboring offer information with target offer is a direct offer to offer influence. We call it *homogeneous influence*. It is in contrast to the non-homogeneous influence from offer to offer when using offers as nodes, where information from an offer node will first pass to a seller/product node and then to another offer node.

Why the 'homogeneous influence'? homogeneous influence more effectively keeps offer information as compared to the 2-hop propagation (offer -> seller/product -> offer) when offers are used as nodes. Some offer features contain strong signals for low-quality listings. However, they will be averaged out by its neighbors during propagation in non-homogeneous influence setting. The reduced signal becomes more difficult for model to detect. Moreover, the message passing process will blend intermediate seller/product node information and added noises signals. In the *homogeneous influence* method, offer features do not go through the propagation process and are largely kept. Concatenating neighboring offer features works for the cold start problem because when there are no features on a new offer we can borrow the features from its neighbors.

[11]We use mean in implementation.

Module 3: Final Classifier. This classifier first concatenates emb_s, emb_p and emb_o as inputs and predict the probability of the offer listing l that will have a defect type z via a multi-layer perceptron.

$$emb_l = f(emb_s \| emb_p \| emb_o) \tag{3}$$

where $\|$ denotes the concatenation operation and f denotes the MLP layer. Finally it uses sigmoid to get the prediction result.

Train Objective. We train ColdGuessin an end to end fashion using mini-batches of offer listings L. We define the cross-entropy loss as the classification error:

$$\mathcal{L} = -\frac{1}{|L|} \sum_{l \in L} z_l \log p_l + (1 - z_l) \log (1 - p_l) \tag{4}$$

where $z_l \in \{0, 1\}$ is the ground-truth for the offer listing l and p_l is the output of the ColdGuess. For a fair comparison with lgbm , which trained 9 binary models one for each class, in this paper we trained ColdGuess in the same fashion. In production, we used the sum of 9 cross entropy loss as the objective function and trained ColdGuess as one model to predict all classes. It reduces the inference time without much performance loss except one class lowered by less than 2% absolute AUC.

We optimize the whole model using stochastic gradient descent. The loss is back-propagated over the entire framework to update all the parameters.

4.4 Handling Dynamic Graphs

ColdGuess is designed to be an inductive model which can handle the dynamic changes of a graph with newly added nodes and edges. It means no need to re-train the model if there is graph updates in the inference stage. In the inductive model the same relation types share the same weights. Therefore, for the newly added nodes in model inference, given their relations to other nodes, they can participate into the propagation with the known weights on relations we obtained from training stage.

5 EVALUATION

5.1 Evaluation Setup

We conducted extensive evaluation on ColdGuess using real data, and compared it with three baseline models. For a fair comparison, all models are trained using the same feature set.

- **lgbm** We chose lgbm [11] as benchmark because (1) it is a commonly used production model in risk detection (2) it can show the value of using graph information.
- **SIGN** [4] is a graph deep learning architecture which sidesteps the need for graph sampling by using graph convolutional filters of different sizes that are amenable to efficient precomputation. It allows extremely fast training and inference. We trained SIGN using the same graph as training ColdGuess. We chose SIGN as benchmark to evaluate the value of using learnable weights in ColdGuess.
- **RGCN** [19] is a graph neural network designed for modeling relational graph data. We chose RGCN as benchmark to compare the performance between using offers as edges vs. as nodes.

We also developed a simplified version of ColdGuess:

- **ColdGuess-Naive** ColdGuess-Naive combines one-hop graph propagation with lgbm . It first fills the missing seller features using one-hop graph propagation, and then feeds the seller features along with product and offer features to a lgbm model. A seller's missing features is filled by simply averaging her connected sellers, with all seller-seller relation types equally weighted. We do not fill offer/product missing features using neighboring offers/products because two products could be totally different things, and test results shows it under-performs filling missing seller features only.

We used the implementation of lgbm from LightGBM project [12]. We used the implementations of SIGN and RGCN from DGL [23] and implemented ColdGuess using DGL and PyTorch [17]. We used Adam optimizer to train our model. For SIGN and ColdGuess, we chose a 3-layer structure. For RGCN, we chose a 6-layer structure to collect comparable neighborhood information as ColdGuess. The embedding dimension of the hidden layers is set to 64.

We chose ROC AUC as the major evaluation metric because (1) rank-based metrics are robust; (2) it shows the FPR/TPR for all possible threshold values and (3) unlike PR AUC which varies as the label distribution of underlying data changes, ROC AUC is easier for comparison across data sets with different label distributions. All numbers reported in the following sections are presented using percentage point (pcp) in AUC, notated as %.

5.2 Data Sets

We collected 5 seller-product data sets, including 1 train set and 4 test sets, by taking snapshots of the seller-product database at the beginning of each month from over 4 consecutive months. Due to confidential limitations, we do not present the detailed statistics of these graphs. We present test result of Month 1 in Table 3. The test results from Month 2 to 4 are similar to Month 1 and shown in the supplementary material. We constructed a heterogeneous graph for each month as described in Section 4.2.

We used customer complaint types as labels. There are eight complaint types based on their root causes, such as expired, defective, damaged, etc. We numbered those complaint types from Type1 to Type8 for illustration purpose. By adding a Normal type into the classification, we ended up with nine types.

We compared ColdGuess with lgbm , SIGN, RGCN and ColdGuess-Naive in four scenarios. Below we explained how each scenario is generated to reflect the real world problem. In all scenarios, a complete Seller-Product graph is used for model training.

- **Full spectrum of sample data**: The graph built on the full spectrum data is denoted as G_o.
- **New offers case**: We created the new offer set by randomly sampling 25% of offer listings for each of the 4 minority complaint classes: Type2 , Type6 , Type7 , and Type8 , and 1% of offer listings from the rest of the classes including the Normal one. In this case, new offer edge features are all set as missing except the ones known once an offer is created. The resulting graph is denoted as G_{no}.
- **New sellers case**: We created the new seller set by first sampling new offers in the same way as described above. Secondly, we took the sellers associated with the sampled

[12] https://github.com/microsoft/LightGBM

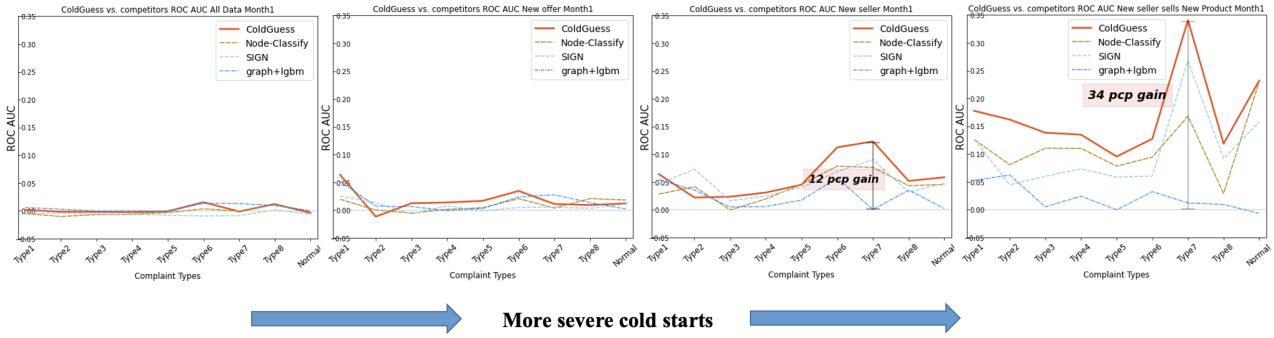


Figure 5: Performance of ColdGuess, lgbm , ColdGuess-Naive, SIGN and RGCN on a full spectrum of sample data, new offer case, new seller case, and new seller sell new product case using data of Month1.

new offers as new sellers. Finally, we extracted all the offers listed under the new sellers as the final new offer set for evaluation. We did not sampled new sellers directly because we want to ensure there is enough data in the minority classes. In this case, the new seller node features and new offer edge features are all set as missing except the ones known once a seller registers or an offer is created. The resulting graph is denoted as G_{ns}.

- **New sellers sell new products case**: We create the data set for this case by sampling the new offers and sellers in the same way as described in the *New seller case*. Given the sampled new offers, we took their associated products as new products . Then we extracted all the offers listed under either the new sellers or new products as the final new offer set for evaluation. In this case, new product and new seller node features, as well as new offer edge features are all set as missing except the ones known once a seller is registered or an offer or a product is created. The resulting graph is denoted as G_{nsnp}

5.3 Performance Evaluation

In this section we investigated the following two questions:

- **RQ1**: Is ColdGuess effective to handle cold start problems?
- **RQ2**: Can ColdGuess scale to large graphs with hundreds of millions of edges?

5.3.1 *Effectiveness of ColdGuess.*

We compared the performance of ColdGuess with lgbm , SIGN, RGCN and ColdGuess-Naive on four use cases using G_o, G_{no}, G_{ns}, G_{nsnp} respectively. Figure 5 shows that ColdGuess starts winning with increasing margins as the severity of cold start cases increases (from the left most figure to the right most figure). ColdGuess outperforms lgbm , SIGN, RGCN and ColdGuess-Naive by up to **34 %**, **11.4 %**, **17.3 %**, **32.9 %** respectively. Table 3 gives the detailed performance numbers of Month 1, and the numbers of Month 2 to 4 are listed in the supplementary materials.

Full spectrum of data. The G_o column in Table 3 shows that ColdGuess and ColdGuess-Naive are on par with lgbm when being evaluated on the entire data set. SIGN and RGCN slightly under-perform lgbm . This is because when a node feature is a strong signal, its value will be averaged out by its neighbors. The reduced signal makes it more difficult for the model to detect it as risky. ColdGuess alleviates this issue by concatenating offer features with offer's neighboring features through homogeneous influence. Offer edge features will not be diluted by its neighbors because aggregation and updates only happen on nodes. However, ColdGuess still slightly under-performs lgbm when node features play an important role in some cases. This finding is line with Sergei et al. [20] that gradient boosted decision trees often outperform other machine learning methods on tabular data.

New offer case. In this case, new offers' offer features are set as missing values except the list price. We filled those missing values with zeros. The G_{no} column in Table 3 shows that ColdGuess outperforms lgbm in 8 out of 9 classes by up to 6.5%. Meanwhile, homogeneous influence brings extra gain to ColdGuess. As shown in Table 3, ColdGuess outperforms ColdGuess-Naive in 7 out of 9 classes, outperforms SIGN in 8 out of 9 classes, and outperforms RGCN in 6 out of 9 classes. ColdGuess under-performs lgbm in Type2 because Type2 's top important features are on product nodes, such as product age, glance view, etc. As mentioned in the full spectrum of data case, those strong signals are averaged out by their neighbors and thus makes it hard to learn by GNN.

New seller case. In this case, we set the features of new sellers and their created offers as missing except the list price. We filled those missing values with zeros. The G_{ns} column in Table 3 show that ColdGuess outperforms lgbm in all classes by up to **12.3%**. It outperforms ColdGuess-Naive, SIGN and RGCN in almost all classes by up to **12.3%**, 4.3% and 4.7% respectively. The fact that SIGN and RGCN outperform ColdGuess-Naive indicates that GNN can improve the performance of cold start cases. In addition to the benefit of GNN, *homogeneous influence* brings extra performance lift to ColdGuess.

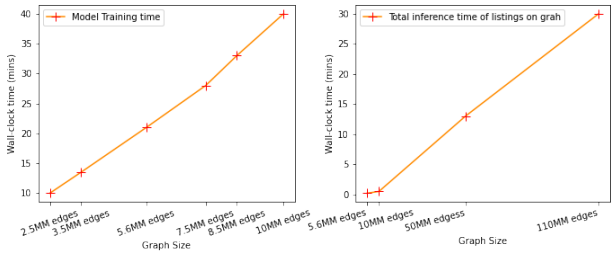
New seller sells a new product case. In this case, features of new sellers, products , and offers are all set to missing values except list price and product category. The G_{nsnp} column in Table 3 shows that ColdGuess outperforms lgbm in all classes by **9.5-34.1%**. It outperforms ColdGuess-Naive, SIGN and RGCN in all classes by

Table 3: ColdGuess wins with increasing margins over ColdGuess-Naive, SIGN, and RGCN as the severity of cold starts increases. G_o, G_{no}, G_{ns}, G_{nsnp} represent full data, new offer case, new seller case, and new seller sells new product case in Month1. We take lgbm as the baseline and present the performance gains of different methods in ROC-AUC.

Defect Type	G_o	G_{no}	G_{ns}	G_{nsnp}
gains of ColdGuess vs. lgbm				
Type1	0.2%	6.4%	6.4%	**17.8%**
Type2	-0.2%	-1.2%	2.2%	**16.2%**
Type3	-0.2%	1.2%	2.3%	**13.8%**
Type4	-0.3%	1.4%	3.1%	**13.5%**
Type5	-0.1%	1.7%	4.5%	9.5%
Type6	1.5%	3.5%	**11.2%**	**12.7%**
Type7	-0.1%	1.1%	**12.3%**	**34.1%**
Type8	1.2%	0.9%	5.2%	**11.9%**
Normal	-0.4%	1.2%	5.9%	**23.2%**
gains of ColdGuess-Naive vs. lgbm				
Type1	0.6%	5.2%	5.4%	5.2%
Type2	0.3%	0.7%	3.5%	6.2%
Type3	0.0%	0.6%	0.5%	0.5%
Type4	0.0%	-0.1%	0.6%	2.4%
Type5	-0.1%	0.3%	1.7%	-0.1%
Type6	1.3%	2.4%	5.7%	3.2%
Type7	1.3%	2.8%	0.0%	1.2%
Type8	0.9%	1.4%	3.5%	0.9%
Normal	-0.1%	0.2%	0.2%	-0.7%
gains of SIGN vs. lgbm				
Type1	-0.5%	2.5%	4.6%	**12.7%**
Type2	-0.4%	1.3%	7.3%	4.8%
Type3	-0.7%	-0.7%	1.6%	6.0%
Type4	-0.7%	0.8%	2.5%	7.3%
Type5	-0.8%	-0.2%	4.0%	5.8%
Type6	-0.9%	0.5%	6.9%	6.0%
Type7	-0.9%	0.5%	9.0%	**26.8%**
Type8	0.2%	0.3%	3.2%	9.2%
Normal	-0.6%	1.1%	4.8%	**15.8%**
gains of RGCN vs. lgbm				
Type1	-0.5%	2.0%	2.8%	**12.6%**
Type2	-1.0%	0.0%	4.1%	8.1%
Type3	-0.7%	-0.5%	-0.1%	**11.1%**
Type4	-0.6%	0.1%	1.9%	**11.0%**
Type5	-0.3%	0.4%	4.6%	7.8%
Type6	0.4%	2.1%	7.9%	9.4%
Type7	-0.1%	0.4%	7.6%	**16.8%**
Type8	1.1%	2.1%	4.3%	2.9%
Normal	-0.1%	1.8%	4.6%	**23.0%**

[a] The AUC gain over lgbm greater than 10 pcp is black bolded and underlined, greater than 5 pcp is underlined.

up to **32.9%**, **11.4%** and **17.3%** respectively. Using node connectivity information and GNN improves the model performance of ColdGuess, *homogeneous influence* further promotes the model performance by a considerable margin.



(a) Training time **(b) Inference time**

Figure 6: ColdGuess scales linearly: training time vs. graph size (left), and inference time vs. graph size (right)

In summary, though ColdGuess, lgbm , ColdGuess-Naive, SIGN, RGCN are tied when being evaluated on the full spectrum of sample data, ColdGuess outperforms all competitors in three cold start cases on 4 different datasets. The stable superb performance of ColdGuess, especially in the *New seller* and *New seller sells a new product* cases, comes from its advantage of leveraging node connectivity information, graph neural network and *homogeneous influence*. ColdGuess outperforming SIGN and RGCN shows that *homogeneous influence* can further boost the performance after node connectivity information and graph neural network are already used.

5.3.2 *Scalability of ColdGuess.*

We studied the training and inference scalability of ColdGuess using graphs of different sizes. All experiments are conducted on p3.8xlarge EC2 instances equipped with 4 V100 GPU. In model training, ColdGuess run on labelled offer listings in each epoch.[13] In model inference, ColdGuess scores all the offer listings as we do in the production system. Figure 6a shows that the training time of ColdGuess on graphs scales linearly from 2.5M edges to 10M edges. Figure 6b presents the inference time of ColdGuess on graphs scales linearly from 5.6M edges to 110M edges.

5.4 Ablation Study

Table 4 demonstrates the efficacy of each component of ColdGuess. For easy comparison, we summarize each model's performance by aggregating the ROCAUC across nine classes using geometric mean[14]. Table 4 compares model performance relative to lgbm in four evaluation cases.

Node connectivity plus lgbm (i.e. ColdGuess-Naive) outperforms lgbm by +0.5% to +2.4% AUC. Node connectivity plus relation type information (i.e. RGCN) improves the performance of ColdGuess-Naive in the last two severe cold start cases by +2.5% to 8.1% AUC[15]. Consolidating nodes in RGCN plus homogeneous influence (i.e. ColdGuess) further improves performance of RGCN by +0.4% to 5.6% AUC. The design decision of consolidating nodes and using homogeneous influence has the highest impact on the prediction accuracy(up to 5.6% AUC gain).

[13] The number of labelled data increases linearly with the number of edges in a graph.
[14] We tried both geometric mean and harmonic mean for aggregation, and found they gave similar results
[15] In non-cold start cases, we observe lgbm works better than graph-based models as later will average out significant signals in neighbor aggregation. See 5.3.1 for details

Table 4: Ablation study shows the efficacy of each component of ColdGuess. We report model performance relative to lgbm using ROCAUC.

Method	G_o	G_{no}	G_{ns}	G_{nsnp}
lgbm	0.0%	0.0%	0.0%	0.0%
ColdGuess-Naive	0.5%	1.5%	2.4%	2.3%
RGCN	-0.2%	0.9%	4.3%	11.6%
ColdGuess	0.2%	1.8%	5.9%	17.2%

6 CONCLUSION AND FUTURE WORK

In this paper, we proposed ColdGuess, a product risk detector which deals with highly multi-relational large-scale graph and addresses the cold start problems via *consolidated nodes* and *homogeneous influence*. ColdGuess enjoys the following advantages over traditional models:

- **General:** This inductive framework can handle dynamic heterogeneous graphs with tens of millions of nodes, hundreds of millions of edges, as well as massive node and edge features.
- **Effective**, especially for *Cold Start Problems*. It outperforms lgbm in all three cold start scenarios by *up to 34 pcp* AUC, and demonstrates stable performance with increased number of missing values.
- **Fast and Scalable:** ColdGuess scales linearly with the input size, as shown in Figure 6. It makes 1MM+ predictions in less than 2 mins on a 4-GPU P3.8xlarge EC2 instance.

ColdGuess has superior performance in cold start problems. It is already in production on 4-GPU P3.8xlarge EC2 instance using AWS Batch, and scores tens of millions of listings on daily basis. It can reach its maximum capacity of scoring hundreds of millions of listings in sales/holiday seasons. Our future work includes adding temporal information so that we can combine multiple months of data to augment the data size of rare classes in training.

REFERENCES

[1] Nabiha Asghar. 2016. Yelp dataset challenge: Review rating prediction. *arXiv preprint arXiv:1605.05362* (2016).

[2] Ye Bi, Liqiang Song, Mengqiu Yao, Zhenyu Wu, Jianming Wang, and Jing Xiao. 2020. A Heterogeneous Information Network based Cross Domain Insurance Recommendation System for Cold Start Users. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2211–2220.

[3] Jidong Chen, Ye Tao, Haoran Wang, and Tao Chen. 2015. Big data based fraud risk management at Alibaba. *The Journal of Finance and Data Science* 1, 1 (2015), 1–10.

[4] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. 2020. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198* (2020).

[5] Liyu Gong and Qiang Cheng. 2019. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9211–9219.

[6] William L. Hamilton, Ren Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. *Advances in Neural Information Processing Systems* (2017).

[7] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[8] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1531–1540.

[9] Xiang Hui, Maryam Saeedi, Zeqian Shen, and Neel Sundaresan. 2016. Reputation and regulations: Evidence from eBay. *Management Science* 62, 12 (2016), 3604–3616.

[10] Xiaodong Jiang, Pengsheng Ji, and Sheng Li. 2019. CensNet: Convolution with Edge-Node Switching in Graph Neural Networks.. In *IJCAI*. 2656–2662.

[11] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems 30 (NIPS 2017)* (2017).

[12] Hyun-Chul Kim, Shaoning Pang, Hong-Mo Je, Daijin Kim, and Sung Yang Bang. 2003. Constructing support vector machine ensemble. *Pattern recognition* 36, 12 (2003), 2757–2767.

[13] Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. 2008. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*. 208–211.

[14] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. 2014. Facing the cold start problem in recommender systems. *Expert Systems with Applications* 41, 4 (2014), 2065–2073.

[15] Yuanfu Lu, Yuan Fang, and Chuan Shi. 2020. Meta-learning on heterogeneous information networks for cold-start recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1563–1573.

[16] V Mareeswari and G Gunasekaran. 2016. Prevention of credit card fraud detection based on HSVM. In *2016 International Conference on Information Communication and Embedded Systems (ICICES)*. IEEE, 1–4.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703* (2019).

[18] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

[19] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 593–607.

[20] Ivanov Sergei and Prokhorenkova Liudmila. 2021. Boost then convolve: gradient boosting meets graph neural networks. *arXiv preprint arXiv:2101.08543* (2021).

[21] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[22] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 968–977.

[23] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. 2019. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315* (2019).

[24] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[25] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 315–324.

7 SUPPLEMENTARY MATERIAL

Table 5, 6 and 7 show that ColdGuess wins with increasing margins over lgbm, ColdGuess-Naive, SIGN, and RGCN as the severity of cold starts increases using test sets from Month 2 to Month 4.

Table 5: ColdGuess wins with increasing margins over ColdGuess-Naive, SIGN, and RGCN as the severity of cold starts increases. G_o, G_{no}, G_{ns}, G_{nsnp} represent full data, new offer case, new seller case, and new seller sells new product case in Month 2. We take lgbm as the baseline and present the performance gains of each model in ROC-AUC.

Defect Type	G_o	G_{no}	G_{ns}	G_{nsnp}
gains of ColdGuess vs. lgbm				
Type1	0.1%	5.2%	5.9%	**17.6%**
Type2	0.1%	-0.5%	2.7%	**15.9%**
Type3	-0.2%	1.3%	2.8%	**14.6%**
Type4	-0.3%	1.1%	2.8%	**12.9%**
Type5	-0.1%	1.7%	4.8%	**10.1%**
Type6	1.7%	2.7%	9.5%	**12.9%**
Type7	0.4%	1.7%	**14.6%**	**33.7%**
Type8	1.9%	2.9%	7.0%	**12.7%**
Normal	-0.4%	1.3%	6.3%	**23.8%**
gains of ColdGuess-Naive vs. lgbm				
Type1	0.4%	3.9%	4.9%	4.7%
Type2	0.3%	0.7%	4.0%	6.4%
Type3	0.0%	0.9%	0.9%	0.7%
Type4	0.0%	-0.3%	0.4%	2.2%
Type5	0.0%	0.2%	1.7%	0.0%
Type6	1.5%	2.2%	4.0%	2.8%
Type7	1.4%	2.6%	3.6%	2.1%
Type8	0.9%	2.0%	3.7%	0.3%
Normal	-0.1%	0.2%	0.3%	-0.9%
gains of SIGN vs. lgbm				
Type1	-0.6%	0.8%	4.2%	**12.1%**
Type2	-0.3%	1.6%	7.8%	5.0%
Type3	-0.5%	-0.5%	2.0%	8.0%
Type4	-0.7%	0.9%	2.2%	7.4%
Type5	-0.7%	0.9%	4.7%	6.8%
Type6	-0.7%	0.9%	5.4%	5.2%
Type7	-0.9%	0.7%	**11.7%**	**25.7%**
Type8	0.5%	2.3%	4.87%	9.6%
Normal	-0.7%	1.5%	5.6%	**17.0%**
gains of RGCN vs. lgbm				
Type1	-0.5%	0.9%	3.0%	**13.2%**
Type2	-0.7%	0.2%	4.5%	7.4%
Type3	-0.6%	-1.1%	0.3%	**11.9%**
Type4	-0.6%	0.9%	1.9%	**10.7%**
Type5	-0.3%	1.6%	5.2%	8.6%
Type6	0.7%	1.8%	6.8%	7.8%
Type7	0.0%	6.2%	**10.9%**	**16.0%**
Type8	2.1%	2.7%	6.2%	4.7%
Normal	-0.1%	2.0%	5.5%	**23.98%**

[a] The AUC gain over lgbm greater than 10 pcp is black bolded and underlined, greater than 5 pcp is underlined.

Table 6: ColdGuess wins with increasing margins over ColdGuess-Naive, SIGN, and RGCN as the severity of cold starts increases. G_o, G_{no}, G_{ns}, G_{nsnp} represent full data, new offer case, new seller case, and new seller sells new product case in Month 3. We take lgbm as the baseline and present the performance gains of different methods in ROC-AUC.

Defect Type	G_o	G_{no}	G_{ns}	G_{nsnp}
gains of ColdGuess vs. lgbm				
Type1	0.2%	5.5%	5.2%	**17.3%**
Type2	0.3%	-0.4%	3.1%	**16.6%**
Type3	-0.2%	1.1%	2.7%	**14.6%**
Type4	-0.2%	1.1%	2.6%	**12.8%**
Type5	-0.1%	1.6%	4.9%	9.8%
Type6	1.0%	1.7%	8.0%	**14.5%**
Type7	-0.6%	1.9%	**12.4%**	**33.0%**
Type8	3.1%	4.3%	**12.4%**	**15.0%**
Normal	-0.2%	1.5%	6.3%	**22.9%**
gains of ColdGuess-Naive vs. lgbm				
Type1	0.7%	4.4%	4.5%	5.1%
Type2	0.4%	0.8%	4.0%	6.4%
Type3	0.1%	0.9%	0.8%	0.6%
Type4	0.1%	-0.4%	0.4%	2.6%
Type5	0.0%	0.4%	2.0%	-0.1%
Type6	1.4%	1.7%	2.5%	3.8%
Type7	2.1%	4.1%	4.6%	1.4%
Type8	2.5%	3.4%	8.3%	0.4%
Normal	0.1%	0.4%	-0.2%	-0.9%
gains of SIGN vs. lgbm				
Type1	-0.2%	1.8%	3.6%	**11.3%**
Type2	-0.2%	1.9%	7.9%	5.7%
Type3	-0.6%	-0.8%	1.9%	7.1%
Type4	-0.8%	0.5%	1.4%	7.7%
Type5	-0.6%	1.4%	5.0%	7.6%
Type6	-3.5%	-3.0%	2.9%	8.0%
Type7	-0.7%	1.3%	9.4%	**25.7%**
Type8	0.9%	2.2%	7.2%	**11.0%**
Normal	-0.4%	1.5%	6.4%	**16.1%**
gains of RGCN vs. lgbm				
Type1	-0.3%	1.8%	2.1%	**12.1%**
Type2	-0.5%	0.9%	4.5%	7.7%
Type3	-0.6%	-1.4%	0.1%	**11.9%**
Type4	-0.5%	0.4%	1.7%	**10.6%**
Type5	-0.4%	1.7%	5.3%	8.1%
Type6	0.4%	1.4%	3.1%	7.5%
Type7	0.0%	1.1%	8.7%	**15.9%**
Type8	3.0%	4.7%	**11.5%**	6.9%
Normal	0.0%	2.0%	6.4%	**24.1%**

[a] The AUC gain over lgbm greater than 10 pcp is black bolded and underlined, greater than 5 pcp is underlined.

Table 7: ColdGuess wins with increasing margins over ColdGuess-Naive, SIGN, and RGCN as the severity of cold starts increases. G_o, G_{no}, G_{ns}, G_{nsnp} represent full data, new offer case, new seller case, and new seller sells new product case in Month 4. We take lgbm as the baseline and present the performance gains of different methods in ROC-AUC.

Defect Type	G_o	G_{no}	G_{ns}	G_{nsnp}
gains of ColdGuess vs. lgbm				
Type1	0.2%	5.8%	6.2%	**17.5%**
Type2	0.1%	0.1%	2.9%	**16.5%**
Type3	-0.2%	1.8%	3.1%	**15.5%**
Type4	-0.1%	1.7%	3.5%	**12.5%**
Type5	-0.1%	2.0%	5.5%	**10.0%**
Type6	0.9%	2.2%	7.1%	**15.1%**
Type7	0.5%	2.0%	**15.1%**	**36.0%**
Type8	3.8%	5.1%	**12.9%**	**16.7%**
Normal	-0.4%	1.6%	7.4%	**25.4%**
gains of ColdGuess-Naive vs. lgbm				
Type1	0.6%	4.4%	5.0%	4.5%
Type2	0.5%	1.1%	4.1%	6.2%
Type3	0.0%	1.3%	1.0%	0.6%
Type4	0.1%	0.0%	0.8%	2.2%
Type5	0.0%	0.5%	2.3%	0.1%
Type6	1.4%	2.2%	2.4%	3.4%
Type7	2.8%	3.7%	7.3%	3.2%
Type8	2.7%	3.4%	6.3%	1.4%
Normal	0.1%	0.4%	0.2%	-0.8%
gains of SIGN vs. lgbm				
Type1	-0.3%	2.1%	4.7%	**11.4%**
Type2	0.0%	3.1%	8.3%	6.1%
Type3	-0.6%	-0.4%	2.5%	8.6%
Type4	-0.8%	1.6%	2.4%	7.9%
Type5	-0.5%	1.7%	5.7%	7.4%
Type6	-4.3%	-3.3%	2.2%	**10.0%**
Type7	0.1%	1.1%	**10.9%**	**28.1%**
Type8	1.3%	3.2%	7.4%	**12.6%**
Normal	-0.6%	1.6%	7.1%	**17.9%**
gains of RGCN vs. lgbm				
Type1	-0.3%	2.4%	3.4%	**12.9%**
Type2	-0.5%	1.6%	4.6%	8.2%
Type3	-0.5%	-0.1%	1.0%	**13.2%**
Type4	-0.3%	1.7%	3.0%	**10.8%**
Type5	-0.3%	1.8%	6.0%	8.7%
Type6	-0.3%	5.7%	4.8%	9.5%
Type7	0.9%	1.5%	**10.6%**	**17.8%**
Type8	3.5%	4.5%	**11.0%**	7.8%
Normal	0.0%	1.6%	6.9%	**25.9%**

[a] The AUC gain over lgbm greater than 10 pcp is black bolded and underlined, greater than 5 pcp is underlined.