Master's Thesis

Embedding customer transactional data for Lifetime Value Prediction

Comparative analysis of discrete and continuous temporal graph neural networks

Ionel Popescu

Student number: 13899430

Date of final version: July 9, 2025

Master's programme: Data Science and Business Analytics

SPecialization: Data Science

Supervisor: Prof. dr. Kevin Pak

Second reader: MSc Ujjwal Sharma

Faculty of Economics and Business

Statement of Originality

This document is written by Student Popescu Ionel who declares to take full responsibility for the contents of this document. I declare that the text and the work presented in this document is original and that no sources other than those mentioned in the text and its references have been used in creating it. I have not used generative AI (such as ChatGPT) to generate or rewrite text. UvA Economics and Business is responsible solely for the supervision of completion of the work and submission, not for the contents.

Contents

Chapter 1

Introduction

Dealing with customer data is probably one of the most obvious use cases of data science tools. One can easily understand why it can be extremely beneficial for a company to predict the behavior of their customers. However, whether it is about customer churn prediction (De Caigny et al. (2021)), click through rate prediction (Alves Gomes et al. (2024)) or, probably the most valuable of them all, predicting customer lifetime value (Fader et al. (2005), Chamberlain et al. (2017)), there are two distinct sources of complexity that companies need to account for. The first is the ever increasing amount of collected data, especially considering the transition towards e-commerce of most of the retail business sector. Secondly, the vast majority of the techniques involved in these prediction problems, address each task individually (Ni et al. (2018)).

The aim of this paper is to investigate the available options for obtaining universal embedding representation of customer transactional data that can subsequently be used in any downstream prediction task. Here, universal embeddings representation refers to vector representation into a general purpose space. This type of representation is usually designed to accurately capture the heterogeneity in the data as opposed to optimally tailoring embeddings towards maximizing performance on a specific downstream task (Ni et al. (2018)). This relatively new idea has been explored using tabular representation of the data and traditional recurrent neural network models (Chamberlain et al. (2017), Bauer and Jannach (2021)). Although successful, these methodologies often include a high level of feature engineering that requires advanced domain knowledge. Additionally, structured relationships like customer-products are harder to capture in the tabular data setup. These particular limitations could potentially be overcome by considering a graph representation of the data, given that the customer-product relationship arises as a natural heterogeneous bipartite graph (Hamilton (2020)). This type of data structures could enable better handling of the sparse nature of customer transactional data by leveraging graph connectivity (Hamilton (2020)). As an example of sparsity in customer transactional data, one could consider that the majority of customers of many retail businesses are one-time only costumers. One other potential advantage is the ability of graph structures to naturally capture the temporal dimension of the transactional data. Temporal graph structures can then be used as input for graph-based learning algorithms such as temporal graph neural networks

(Rossi et al. (2020)).

Temporal graph neural networks (TGNN) proved their utility in diverse applications such as traffic forecasting (Mallick et al. (2019)) or predicting the spread of contagious diseases (Panagopoulos et al. (2020)). Considering them in the context of customer transactional data is more than justified. As mentioned earlier, predicting customer lifetime value is considered to be a difficult task which makes it an ideal candidate for testing the performance of the different embedding techniques. The contribution of the present paper comes from investigating the validity and benefits of temporal graph neural networks in obtaining relevant embeddings of customer transactional data. The experiments are designed to address the following research questions:

RQ1 : *Can embeddings of customer transactional data obtained through temporal graph neural networks improve CLV predictions, when compared with traditional probabilistic models?*

RQ2 *Which type of temporal graph representation - discrete time or continuous time - is more appropriate in the context of customer transactional data representation*?

Chapter 2

Literature Review

Customer lifetime value (CLV) measures the expected total revenue generated by a customer over a specified finite period of time. This metric can be particularly important for marketing purposes, since it can help identify the most valuable customers, or it can provide an overview of the financial health of the company.

The CLV prediction problem is typically defined in a non-contractual environment. In other words, the purchasing behavior of the customer is completely random from the seller's point of view. Thus, given the random nature of this behavior, it is natural that the traditional approach to solving this problem is through probabilistic models. The main idea behind this approach is that the behavior of customers can be modeled using some assumed underlying probability functions (Schmittlein et al. (1987), Fader et al. (2005)). The idea was introduced by Schmittlein et al. (1987), who assume that the customer purchase frequency is modeled by a negative-binomial distribution (NBD), while the customer lifetimes are generated from a Pareto II distribution. What this second assumption implies is that customer dropouts, or customer churn as it is often called, can occur at any point in time after the last purchase. In contrast to this, the Beta Geometric/Negative Binomial Distribution model (BG/NBD) assumes that customer dropouts occur immediately after the last purchase (Fader et al. (2005)).

Although in many retail domains the probabilistic approach proved to be very successful, there are still important limitations of this model. Firstly, only a very limited part of the available customer data is considered. More precisely, they typically use RFM (recency, frequency, and monetary value) data (Fader et al. (2005)). Kumar and Pansari (2016) show how cultural particularities impact the purchasing behavior of the population in different countries. Consequently, if we consider multi-national retail companies, it makes little sense to assume that the RFM data is generated by the same distributions. This simple example is the kind of argument that can justify the need for more complex models that could incorporate as much as possible from the available customer data.

The next best approach to the CLV prediction problem is supervised learning. Here, researchers choose to look into highly flexible models which can model the complex behavior of customers. For example, Vanderveld et al. (2016) use a random forest model in the context of

a very large e-commerce business. They used customer engagement data from both the mobile app and web browser activity to derive more than 40 features that allow them to outperform traditional models. While successful, their application is a perfect example of how these models are usually domain specific. More exactly, the performance of machine learning approaches are highly dependable of specific to sector customer behavior, specific domain features and other similar considerations that are not reproducible for other businesses. One suggestion to further improve this approach was to stack multiple ensemble models like random forest or gradient boosting algorithms (Yan and Resnick (2024)). Usually this is a multistage approach where churn prediction and CLV prediction are considered to be interconnected tasks. Yan and Resnick (2024) separately train three different random forest models, one for predicting probability of churn, one for predicting average order value and one for predicting number of future orders placed by the customer. Stacking multiple models methodology is present in most of the research that reported improvement on probabilistic approach (Gadgil et al. (2023), Yan and Resnick (2024))

A different direction in the research of CLV prediction is given by Chamberlain et al. (2017) that on top of considering an initial set of 132 features in their data set, they generate customer embeddings by considering their view history. They further use these embedding in a stacked supervised learning model. Bauer and Jannach (2021) further improve onto this approach by considering a sequence-to-sequence RNN with temporal convolutions and dynamic features to obtain customer purchase embeddings. The novelty in their approach is given by the attention they pay to the temporal dimension of the data, as opposed to Chamberlain et al. (2017) who only consider a snapshot in time in obtaining the customer embeddings. The validity of considering the temporal dimension is also confirmed by Alves Gomes et al. (2024) who use skip-gram method to obtain embeddings of the customer activity data, which they subsequently use in a recurrent neural network model to predict the click-through rate. They report strong evidences in favor of using self-supervised models to represent customer data.

One thing that all these methods have in common is the tabular representation of the data. However, recent research in graph structures suggest that temporal graph representation are likely to better capture the relationships between a network of entities, especially in a temporal context (Nguyen et al. (2018), Rossi et al. (2020)).

To the best of my knowledge, graph structures were not considered so far in representing customer transactional data, although customers-products networks arise as a natural bipartite graph (Hamilton (2020)). More exactly, customers and products could be represented in a network as two distinct types of nodes, where edges only connect customers nodes with products nodes. Starting from this core observation, the customer transactional data can be modeled using a temporal graph network which is a specific neural network model, designed as an encoder-decoder pair (Rossi et al. (2020). The encoder part of the network maps the nodes of the graph into an embedding space, while the decoder part uses the data embeddings in a predefined task of the model which can be supervised or self-supervised.

In conclusion, temporal graph neural networks could be a promising solution for obtaining helpful embedding representation of customer data. Two distinct self-supervised tasks could be considered: contrastive learning and link prediction/graph completion (Hamilton (2020). The first method trains the network to distinguish between similar and different data points such that, the more different two data points are, the farther apart they will be in the embedding space. On the other hand, the link prediction task trains the network to predict whether there should be an edge between two nodes or not.

Chapter 3

Data description

The dataset considered for the experiments is the well known *Online retail dataset*, made available through the UCI Machine Learning Repository (Daqing Chen (2012)). It contains transactional data for a UK-based non-store, online retail business that primarily sells unique all-occasion giftware. The data span from December 1st 2009 to December 12 2011 and consists of more than one million rows, where each row captures an individual transaction. There are 8 initial features in the dataset, as follows:

- *InvoiceNo*: unique invoice number assigned to each purchase made by a customer. There are no missing values and if the invoice number starts with letter 'C' it denotes a cancellation.

- *StockCode*: A unique identifier for each product available in the business inventory. There are 5305 unique products in the dataset, with no missing values. Nevertheless, I identified several likely non-product like 'M', 'POST', 'DOT'. I assume these entries refer to administrative and shipping costs that can negatively impact the predictive performance of the models. There are a total of 5459 rows associated with such non-product related entries, which I choose to remove from the dataset, given that these entries are presumed independent of customer purchase behavior.

- *CustomerID*: unique identifier for each customer. Approximately 20% of the data in this column are missing values. Upon a closer analysis of these entries one can notice how approximately 70% of them are of 0 value invoices. What is more, they are relatively uniformly distributed in time. I assume that these purchases are most likely the result of promotional campaigns run by the store. Consequently, aside from the fact that these transactions must be removed due to lack of customer identification, I expect that they do not have too big of an impact in the predictive power of the models.

- *Description*: The name and small description of the product. This is a good example of high-cardinality categorical feature, which presents challenges for traditional machine learning models.

- *Quantity*: number of units of product purchased by the customer. Negative values indicate returns or cancellations.
- *InvoiceDate*: date and hour of the purchase. This attribute enables engineering of rich features like frequency and recency, which are essential for the probabilistic approach of CLV, but also of complex purchase time patterns like seasonality.
- *UnitPrice*: price per unit of product.
- *Country*: the country of the customer. There are 43 unique countries in the dataset, but more than 92% of the orders are placed from UK. Nevertheless, this is another good example of high-cardinality categorical feature. Although, one could easily consider restricting the number of dimensions, for example through grouping European countries and non-European countries into two distinct categories.

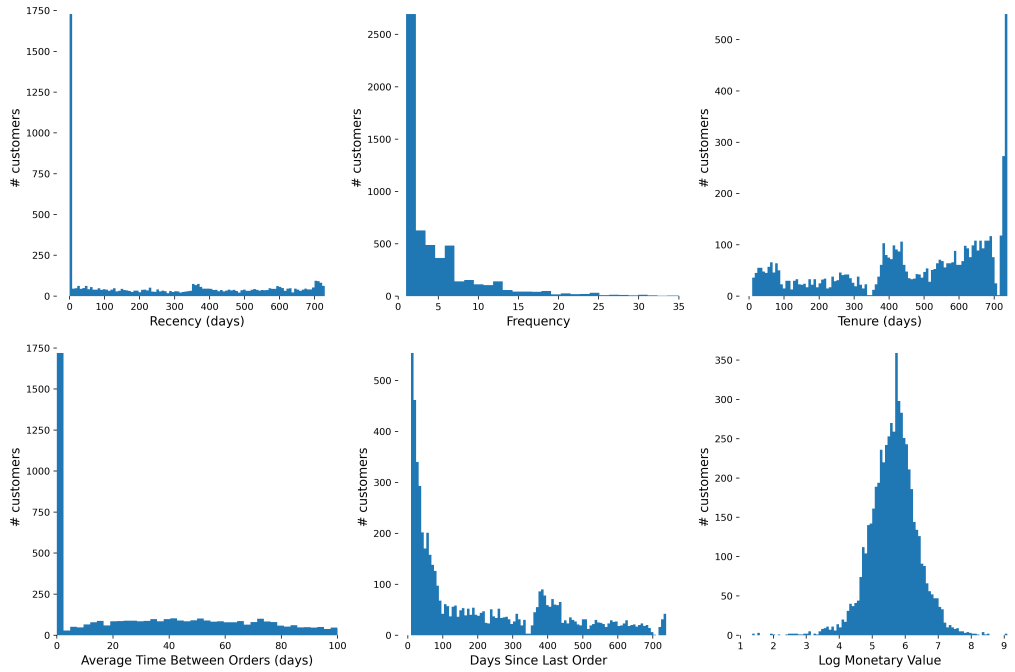

Figure 3.1: Distribution of Key RFM and Temporal Features. In the first column of the graphs we can notice how out of the total of 5790 unique customers, almost 1750 are one time only customers. Additionally, for the repeat customers there is a relatively uniform distribution of the average time between orders. The frequency distribution shows that the vast majority of customers have not place more than 10 orders. The tenure and the number of days since last order describe a clear seasonality in acquiring new customers, but also in their purchasing behavior.

Following the initial exploration, the dataset was cleaned to retain only meaningful customer-product interactions. This involved removing rows with missing customer ID's, zero-revenue transactions, and non-product StockCode's. The resulting cleaned dataset, containing approximately 800,000 observations, forms the basis for constructing recency, frequency, and monetary value (RFM) features, which are central to the CLV estimation methodology described in the

next section. Closer analysis into these features reveals there are 5852 unique customers that cumulatively placed approximately 37000 orders over the two years. The median customer placed 3 orders for a total of 44 different products, for which they spent 300 pounds. The median inter-purchase time is 40 days, while the mean is 53 days, indicating a slightly skewed distribution with some customers exhibiting longer gaps between purchases. This insight provides the starting point in deciding the validation period for the CLV predictive models considered in the experiments.

Figure 3.1 further improves the understanding of the dataset. For example, looking at the distribution of the number of days since the last order, there is a clear pattern of seasonality. It looks like there is an increase in the number of buying customers in the last quarter of the year, during the winter holidays, followed by a very weak month of January. This could justify the need for multiple validation periods that could capture both peak-season customer behavior and off-peak customer behavior.

In addition to the traditional features presented in figure 3.1, I also consider obtaining additional features that could potentially improve the quality of customer representation. For example, I consider the time of the day the orders were placed, which shows that the majority of orders were placed within normal business working hours, before 18:00. Coupled with the observation that the median customer buys 17 unique items per order, I conclude that the majority of customers are small businesses. Lastly, given the times series nature of the customer transactional data, I also consider lags of total monthly purchases.

Chapter 4

Research design

In the present paper, I aim to investigate the validity and benefits of using self-supervised learning in obtaining relevant embedding representations in a vector space of customer transactional data. As mentioned before, the performance of this approach is tested in the context of the subsequent CLV prediction problem. Thus, I begin this section by introducing a more detailed description of the traditional probabilistic approach, which plays the role of the benchmark model. Next, I introduce a more detailed description of the temporal graph structures that I consider in the initial representation of the data. This aspect in particular is one of the novelties introduced in this paper. More exactly, to my knowledge, there has been no previous research in which graph structures are used for this purpose. I continue with describing the architecture of the temporal graph neural network used in learning the embedding space. Finally, I introduce the evaluation methodology for measuring the performance of the methods in the context of the CLV prediction problem.

4.1 Probabilistic approach of the CLV problem

The prediction of CLV within the probabilistic approach starts with distinguishing between two key aspects of the customer behavior: how often will customers place orders and how much they will spend for each purchase. Thus, the expected CLV of a customer for a fixed period of time is defined as:

$$\mathbb{E}[\text{CLV}] = \mathbb{E}[\text{Number of future purchases}] \times \mathbb{E}[\text{Monetary value per purchase}] \tag{4.1}$$

For the present paper I consider modeling the number of future purchases with the BG/NBD model (Fader et al. (2005)), while for the expected monetary value I use the Gamma-Gamma model (Colombo and Jiang (1999)).

The beta-geometric/negative-binomial distribution model has a number of five underlying assumptions:

I. The number of transactions made by an active customer follows a Poisson process with a transaction rate λ. This in turn implies that the time between transactions for an active

customer is exponentially distributed with rate λ, or, in other words:

$$f(t_j|t_{j-1};\lambda) = \lambda \exp\{-\lambda(t_j - t_{j-1})\}, \text{ for } t_j > t_{j-1} \geq 0. \tag{4.2}$$

II. Customers differ in their transaction rates, such that each customer transaction is modeled by a unique Poisson process. The heterogeneity in λ parameter across customers is modeled by a gamma distribution with shape parameter r and scale parameter α,

$$f(\lambda|r,a) = \frac{\alpha^r \lambda^{r-1} e^{-\lambda\alpha}}{\Gamma(r)}, \text{ for } \lambda > 0. \tag{4.3}$$

III. After any transaction, a customer may become inactive with a probability p (churn probability). This is the key distinction with the Pareto/NDB model where a customer can become inactive at any point in time after the last transaction (Fader et al. (2005)). Consequently, in the context of BG/NBD model, the point at which a customer "drops out" is distributed across transactions according to a shifted geometric distribution. Thus, immediately after any transaction,

$$\mathbb{P}[\text{inactive immediately after} j^{th} \text{ transaction}] = p(1-p)^{j-1}, \text{ for } j = 1, 2, 3, \cdots \tag{4.4}$$

IV. The churn probability p varies across customers following a beta distribution with shape parameters a and b. Thus,

$$f(p|a,b) = \frac{p^{a-1}(1-p)^{b-1}}{B(a,b)}, \text{ for } 0 \leq p \leq 1, \text{ where } B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}. \tag{4.5}$$

V. The transaction rate λ and the churn probability p vary independently across customers.

The model requires only two inputs from customer purchase history: recency (t_x - number of unit times since the last transaction) and the frequency (x, the number of transactions during and observation period T). Then, under assumptions I-V, Fader et al. (2005) derive the likelihood function for individual customers and then, the likelihood function for a randomly chosen customer $(X = x, t_x, T)$, given parameters r, α, a, and b (as defined in equations 4.3 - 4.5):

$$\begin{aligned} L(r,\alpha,a,b|X=x,t_x,T) = &\frac{B(a,b+x)}{B(a,b)} \frac{\Gamma(r+x)\alpha^r}{\Gamma(r)(\alpha+T)^{r+x}} \\ &+ \delta_{x>0} \frac{B(a+1,b+x-1)}{B(a,b)} \frac{\Gamma(r+x)\alpha^r}{\Gamma(r)(\alpha+t_x)^{r+x}}, \end{aligned} \tag{4.6}$$

where $\delta_x = 1$ if $x > 0$ and 0 otherwise. The four model parameters are finally estimated using maximum likelihood estimation (MLE). More exactly, by maximizing the sum of the log-likelihoods across all customers,

$$l(r,\alpha,a,b) = \sum_{i=1}^{N} \log[L(r,\alpha,a,b|X_i = x_i, t_{x_i}, T_i)] \tag{4.7}$$

In a similar fashion, Colombo and Jiang (1999) describe different five underlying assumptions regarding the expected monetary value of customers. Here I mention the three most important ones that mention the distributional assumptions of their model:

VI. The monetary value of a customer's transaction is independent of the transaction process itself. In other words, the amount a customer spends on any given purchase is not correlated with the frequency or timing of their orders.

VII. The variability in the amount spent by an individual customer on each transaction is modeled using a gamma distribution with parameters u and θ.

VIII. Customers differ in their average transaction values. This heterogeneity is captured by assuming u is constant across customers, while θ is described by a second gamma distribution (hence the name of the model), with parameters ν and ϕ.

Under these main assumptions, the conditional expectation of a customer's average monetary value is given by the following equation derived by Colombo and Jiang (1999):

$$\mathbb{E}[w_i|u, v, \theta] = \frac{u(x_i, \bar{z}_i + \theta)}{ux_i + v - 1.} \tag{4.8}$$

In other words, we can compute the expected monetary value for any customer i who has made x_i past number of transactions, with an observed average monetary value $\bar{z}_i$. Just like in the case og BG/NBD, the parameters of the Gamma-Gamma model (u, v and θ) are typically estimated using maximum likelihood estimation (MLE). It is important to mention at this point that all these assumptions only hold in the context of a non-contractual relationship with the customers.

I choose to use the BG/NBD model, coupled with the Gamma-Gamma model, as the benchmark results, due to its straightforward use and well-documented theoretical support. Additionally, the model is computationally efficient and requires minimal feature engineering.

4.2 Representing customer data with temporal graphs

The main contribution of this paper comes from considering graph structures to represent customer transactional data. This idea comes from the observation that transactional data is fundamentally about relationships: customer-product relationships, product-brand, store-customer, and even customer-customer relationships. Graphs provide a natural and intuitive way to represent these entities as nodes and their interactions as edges. This could potentially provide a way of capturing a richer representation of this type of data. For example, a customer's likelihood to purchase a given product might be strongly influenced by what similar customers have bought, but this type of relationship might be difficult to capture with traditional tabular methods.

Formally, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}$ is defined by a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$ that capture the relationships between these nodes (Hamilton (2020)). The customer transactional

data arises like a natural graph, which is a special case of graphs that are characterized by two distinct types of nodes, with no edges between nodes of the same type. In the context of the dataset considered in the experiments (Daqing Chen (2012)), there are customer nodes and product nodes, where customers and products are considered independent. However, this can be further generalized to a heterogeneous graph structure where $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_k$, where $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$, for $\forall\, i \neq j$ (ex. customers, products, brand and seller). Furthermore, both nodes and edges can have associated attributes to better describe the nodes themselves and also, the relationships between them. There are also node-level statistics and graph-level statistics that are used to characterize these types of structure. However, for further details in this sense, the reader is referred to Hamilton (2020) who provides a thorough introduction to these elements.

On top of the bipartite graph structure there is the temporal aspect that needs to be captured by the data representation. In that sense, there are two distinct approaches: discrete temporal graphs (DTG) and continuous temporal graphs (CTG).

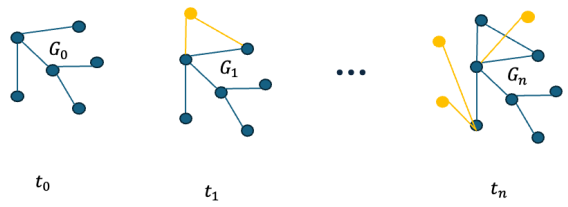

Figure 4.1: Graphical representation of a discrete temporal graph structure

Discrete temporal graph structures are a collection of snapshots taken of the same graph as it evolves in time. Figure 4.1 provides a sketch of the logic behind discrete temporal graph structures. So, DTG is an object $\mathcal{G} = \{\mathcal{G}_t,\ \text{for}\ t = 0, 1, ..., T\}$. To be even more exact, at t_0 $\mathcal{G}$ contains snapshot $\mathcal{G}_0$, at time t_1 snapshots $\mathcal{G}_0$ and $\mathcal{G}_1$ and so on until t_n, hence the discrete nature of the temporal representation of the data. What is not entirely clear from this visual representation is that node attributes can change in time as well. For example, the status of a customer can become inactive with time. In conducting the experiments I choose monthly snapshots of the customer transactional data. This decision is sufficiently justified from multiple points of view. Firstly, monthly aggregation can smooth out random fluctuations in number of purchases that might be present in a more granular representation (daily of weekly). Also, by the nature of the interactions, transactional data can be very sparse, especially on daily or weekly basis. Secondly, from a business point of view, customer purchase behavior is often

times influenced by monthly cycles, such as paydays or billing. This is particularly relevant for the *Online Retail* dataset, given that most likely, a considerable number of customers are small business. Lastly, considering a larger time granularity could come with significant computational complexity. For example, for the two years of data in the dataset, there are 24 monthly snapshots or 104 weekly snapshots.

In terms of information included in the graph, DTG allow for dynamic attributes for both nodes and edges. Consequently, I focus on including as many attributes as possible for both of these graph elements. Thus, for each snapshot, I generate the updated attributes for the customer, such as the RFM data, but also temporal features, such as the most likely time of day when the customer places the order, the average time between orders, or the number of unique products ordered until time t. Similarly, for products, I consider total number sold during the last snapshot of time, the average unit price, the number of unique customer, and the description of the product.

The continuous temporal graph can be represented as a timed list of events which may include edge addition or deletion and node addition or deletion (Rossi et al. (2020)). Using the introduced notation, the continuous temporal graph object is an object of the form $\mathcal{G} = x(t_1), x(t_2), ...\}$, where events $x(t)$ can belong to one of the two categories: *node events* at time t, $v_i(t)$, where i is the index of the node i and v is the vector of attributes (or features) associated with the event. An example of node event in this context is the acquisition of a new customer; *interaction events* $e_{ij}(t)$ between node i (customer) and node j (product), for example, a purchase of a product j by customer i. Consequently, following this logic, at any point in time t, there is a graph representation $\mathcal{G}(t) = \{\mathcal{V}[0,t], \mathcal{E}[0,t]\}$, where $\mathcal{V}[0,t] = \{i : \exists\, v_i(t) \in \mathcal{G},\ \text{for } t \in [0,t]\}$ and $\mathcal{E}[0,t] = \{(i,j) : \exists\, e_{ij}(t) \in \mathcal{G},\ \text{for } t \in [0,T]\}$. Here, T is some fixed final moment in time. Also, it is important to stress that the difference in the context of CTG is subtle and it comes from the fact that t is continuous time. Consequently, for DTG structures, $\mathcal{G}$ captures event by event interactions between nodes, whereas DTG only capture the changes for fixed windows of time. Thus, we can conclude that CTG are a generalized form of DTG that allow for data representation at the smallest level of temporal granularity available. As a consequence to this, flexible machine learning models can uncover complex temporal relationships that are potentially not available at a discrete time level (Rossi et al. (2020)).

In choosing between the two versions of temporal graphs, Skarding et al. (2021) show that both choices have their strengths and weaknesses. In terms of temporal representation, DTG are restricted by a fixed time window that is arbitrarily chosen in the modeling phase. On the other hand, CTG carry the most information due to exact temporal data, theoretically offering higher potential to model dynamic networks. Also, unlike DTG, they are independent of snapshot definitions. In other words, increasing temporal granularity in DTG can lead, in some situations, to snapshots with no network structure. In the context of the dataset considered in this paper, DTG can handle richer information representation of the customer

nodes. In particular, I derive updated RFM features and other engineered features in each new snapshot. This is not permitted in the CTG framework which only allows for static features of the customers. Thus, in a specific context like customer transactional data representation this aspect might lead to a significant loss of information.

4.3 Graph Neural Networks

Graph neural networks (GNN) are a subclass of neural network models that have emerged as a solution to specific real-world data sets that are naturally best represented in graph structures: social networks, protein interaction networks, knowledge graphs, etc.(Hamilton (2020)). Their purpose is to learn to represent the data in a vector space that can subsequently be used as input for a different supervised learning task (e.g., classification, clustering). There are two main sources of information that are used as input for these models: features based information that comes from the attributes of the nodes and edges, and structural based information that comes for relationships between nodes in the graph.

The core idea behind the basic GNN architectures is the message passing (or neighborhood aggregation). In this paradigm a node state is updated by aggregating information from its neighboring nodes. By state, here I refer to its hidden vector representation. Mathematically, the general formulation of a GNN layer is

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}_{\text{self}}^{(k)}\mathbf{h}_u^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)}\sum_{v\in\mathcal{N}(u)}\mathbf{h}_v^{(k-1)} + \mathbf{b}^{(k)}\right). \tag{4.9}$$

In other words, the embedding (or the hidden representation), denoted as $h_u^{(k)}$ of the node u in the iteration k, is a function of the previous state of the node and the message from all nodes v in the neighborhood of u, $\mathcal{N}(u)$. Important to mention here is that $h_u^{(0)}$ represents the initial attributes (or input features) of the node u. In addition to that, σ is a basic activation function (e.g. tanh or ReLU), also called in the context of GNN, update function. $\mathbf{W}_{\text{self}}^{(k)}$ and $\mathbf{W}_{\text{neigh}}^{(k)}$ are trainable parameter matrices and the summation over the neighboring nodes is the message function $\mathbf{m}_{\mathcal{N}_u}$. Thus, equation 4.9 can also be expressed as

$$\text{Update}(h_u, \mathbf{m}_{\mathcal{N}_u}) = \sigma(\mathbf{W}_{\text{self}}h_u + \mathbf{W}_{\text{neigh}}\mathbf{m}_{\mathcal{N}_u}) \tag{4.10}$$

Figure 4.2 illustrates the concept of message passing for GNN. An important aspect to note is how this process captures the intricate relationships between nodes. Also, this approach is particularly developed in the context of a static graph where there are no changes in the structure of the graph or in the natures of its nodes and edges.

The parameters that are estimated during the training of the GNN, as briefly mentioned before, are the weight matrices $\mathbf{W}_{\text{self}}^{(k)}$ and $\mathbf{W}_{\text{neigh}}^{(k)}$. These matrices are the shared components of the message (or aggregation) function $\mathbf{m}_{\mathcal{N}_u}$, and of the Update function. The parameters are usually estimated using well known gradient descent algorithms like stochastic gradient decent

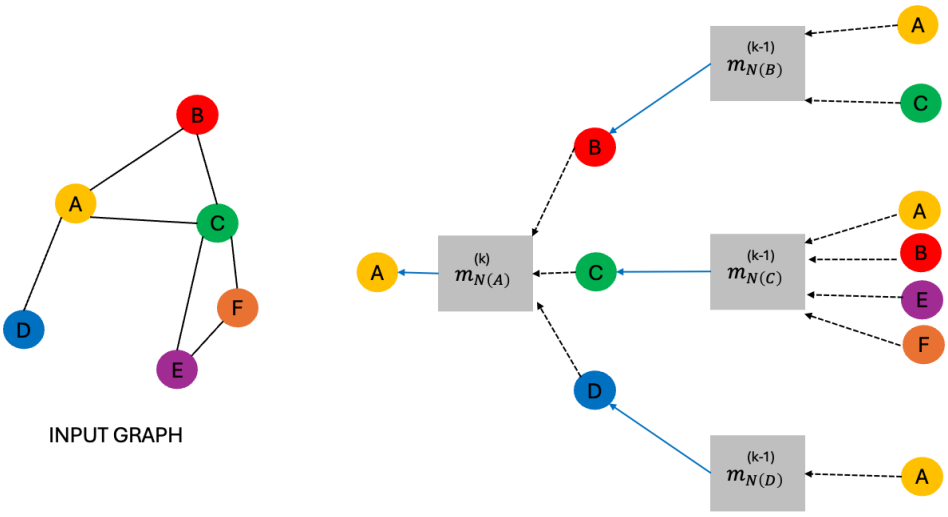

Figure 4.2: Graphical representation of message passing concept in the context of Graph Neural Networks. The graph shows the message passing process to node A from its neighbors B,C and D, in iteration k. In turn, each one of them updated their state in iteration $k-1$ using messages from their own neighbors.(Hamilton (2020)).

(SGD) or Adam. One critical element in here is the loss function, which basically defines the task of the model. I expand on the specific loss function used in the experiments in a later subsection.

4.4 Temporal Graph Neural Networks

There are two main distinct approaches in designing GNN that incorporate temporal dynamics: discrete temporal graph neural networks (DTGNN) and continuous temporal graph neural networks (CTGNN) (Skarding et al. (2021)). Naturally, their differences arise from the way data is represented by each of them. Skarding et al. (2021) emphasize the increased complexity in the implementation of CTGNN and that a general framework for their use is yet to be developed. On the other hand, DTGNN are more present in academic research and have a broader implementation infrastructure. However, given the novelty of graph representation of customer transactional data, I consider experimenting with both of these options.

4.4.1 DTGNN Model

For the discrete temporal model I choose to experiment with an integrated TGNN, which "is an encoder that combines GNNs and recurrent neural networks (RNNs), thus combining modeling of the spatial and temporal domain in one layer" (Skarding et al. (2021)). I consider a Heterogeneous Graph Convolutional Long Short Term Memory (HeteroGCLSTM) model (Rozemberczki et al. (2021)), which is a generalization of the model proposed by Seo et al. (2016). The main distinction in HeteroGCLSTM model is that each node type has its own weights in the LSTM cell. For example, following the formulation proposed by Seo et al. (2016), the customer nodes

will accounted for in the model by

$$
\begin{aligned}
i^c &= \sigma(W^c_{xi} *_{\mathcal{G}} \; x^c_t + W^c_{hi} *_{\mathcal{G}} \; h_{t-1} + w^c_{ci} \circ c^c_{t-1} + b^{ct}_i), \\
f^c &= \sigma(W^c_{xf} *_{\mathcal{G}} \; x^c_t + W^c_{hf} *^c_{\mathcal{G}} \; h_{t-1} + w^c_{cf} \circ c^c_{t-1} + b^c_f), \\
c^c_t &= f^c_t \circ c^c_{t-1} + i^c_t \circ \tanh(W^c_{xc} *_{\mathcal{G}} \; x^c_t + W^c_{hc} *_{\mathcal{G}} \; h_{t-1} + b^c_c), \\
o^c &= \sigma(W^c_{xo} *_{\mathcal{G}} \; x^c_t + W^c_{ho} *_{\mathcal{G}} \; h_{t-1} + w^c_{co} \circ c^c_t + b^c_o), \\
h_t &= o^c \circ \tanh(c^c_t).
\end{aligned}
\tag{4.11}
$$

Here, I use superscript c to denote customer nodes. Next, note the most important elements of this model:

- $x^c_t \in \mathbb{R}^{n_c \times d^c_x}$ is the input feature matrix for n_c customer nodes and d^c_x customer features, at time t.
- $W^c *_{\mathcal{G}} \; x^c_t$ denotes the graph convolution of x^c_t. Intuitively, this is a specialized message function applied to the input features of a customer node, analogous to 2D convolution functions used in computer vision. Further details of these aspect are beyond the scope of this paper.
- c^c_t is the LSTM cell state at time t, which can be described as the memory of the graph at time t.
- $i^c, f^c, o^c \in [0,1]^{d_h}$ are the input, forget and output gates. These are essential elements of the LSTM cell in learning the long-term dependencies in the data. In other words, their weights are updated in the learning process so that the model optimally selects what to "remember" and what to "forget" over time.
- $\circ$ is the element wise product.

Similarly, product nodes will have their own weights in the model. It is important to note here that the convolution operator $*_{\mathcal{G}}$ is using information from both customer nodes and product nodes.

The self supervised aspect of the model is performed through the task defined in the training phase of the model. In this scenario, the model has to learn a meaningful embedding representation of the customer transactional data which is usually approached with the link/relation prediction (Hamilton (2020)). Intuitively speaking, the model has to predict whether there is a link between two nodes or not. I choose to use the binary cross entropy with logits loss function, which is part of a standard class of loss functions for this kind of self-supervised task (Hamilton (2020)). The loss function is defined as

$$
\begin{aligned}
L = &\sum_{(u,v)\in\mathcal{D}_{\text{pos}}} \left[\max(h_u^\top h_v, 0) - h_u^\top h_v + \log(1 + e^{-|h_u^\top h_v|})\right] \\
&+ \sum_{(u',v')\in\mathcal{D}_{\text{neg}}} \left[\max(h_{u'}^\top h_{v'}, 0) + \log(1 + e^{-|h_{u'}^\top h_{v'}|})\right],
\end{aligned}
\tag{4.12}
$$

where h_u, as before, denote the node embeddings and $\mathcal{D}_{\text{pos}}$ and $\mathcal{D}_{\text{neg}}$ are sets of pair of nodes (u, v) that exist in the graph (positive observations) and (u', v') do not exist in the graph (negative observations). So, by minimizing this loss function, the model changes the parameters of the model such that the embeddings of connected nodes are pushed closer to each other in the embedding space and embeddings of unconnected nodes further apart.

In conclusion, HeteroGCLSTM module is a natural choice for building a base model in the context of customer transactional data represented in a discrete temporal graph object. It leverages the established benefits of integrated DTGNN models for spatio-temporal learning (Skarding et al. (2021)), and is specifically designed to handle the bipartite nature of the data.

It is important to note here that the model can be easily modified for a regression task, to directly predict the CLV for a customer. Although not entirely relevant in the context of embedding representation, directly training the DTGNN model in a supervised manner could potentially provide a clearer understanding of the performance of the self-supervised approach. Thus, in training the DTGNN model for the regression task, I consider as an output the 3 months CLV for each customer and the mean squared error (MSE) loss function.

4.4.2 CTGNN model

Probably the main difference in the continuous-time paradigm is the *memory* module. Its purpose is to keep a timestamped vector $s_i(t)$ for each node i the model has seen until some time t. This module enables the model to memorize long term dependencies for each node in the graph (Rossi et al. (2020)). So, in the context of customer transactional data, each customer and each product have their own vector of states that is updated using a *memory updater* function

$$\begin{aligned} s_i(t) &= \text{mem}(\bar{\text{m}}_i(t), s_i(t^-)), \text{ where} \\ \bar{\text{m}}_i(t) &= \text{agg}(\text{m}_i(t_1), \text{m}_i(t_2), ..., \text{m}_i(t_b)), \text{ for } t_1, t_2, ..., t_b \leq t_b. \end{aligned} \tag{4.13}$$

Here, $\bar{\text{m}}(.)$ is the *message aggregator* function. Its role is to aggregate over the generated messages for a node i, in the context of batch processing. Additionally, the message function $\text{m}_i(.)$ is applied twice for every purchasing event, once for the customer nodes and once for the product nodes:

$$\begin{aligned} \text{m}_{cust}(t) &= \text{msg}(s_{cust}(t^-), s_{prod}(t^-), \Delta t, e_{cust,prod}(t)), \\ \text{m}_{prod}(t) &= \text{msg}(s_{prod}(t^-), s_{cust}(t^-), \Delta t, e_{cust,prod}(t)) \end{aligned} \tag{4.14}$$

The second core module of the model is the *embedding module*, used to generate temporal embeddings $z_i(t)$ for node i at time t. Rossi et al. (2020) consider multiple options for implementation of this module. Without going into details, I mention here the three options:

- *Identity* function: $\text{emb}(i, t) = s_i(t)$. It uses the memory state as the node embedding
- *Time projection* function: $\text{emb}(i, t) = (1 + \Delta t \mathbf{w}) \circ s_i(t)$. Here $\mathbf{w}$ are learnable parameters and Δt, as before, it the time since the last event.

- *Temporal graph attention*: This is a more complex implementation of this module, but its core idea is that it uses L graph attention layers to compute node's i embedding by aggregating information from L-hop temporal neighborhood. In other words, the model "looks" in a neighborhood of nodes that it can reach in at most L steps and learns to which neighbors it should pay more "attention" by assigning them more weight.

I experiment with all these three options, although I expect *time projection* function to perform better in the context of customer transactional data. The state of a customer is highly dependent of its temporal characteristic. For example, the longer the time since last order, the most likely the customer is inactive. By considering the time mechanism Δt, the model places a larger emphasis on temporal aspects like this.

Finally, the training procedure is similar to the one described in the DTGNN model. The self-supervised learning task is link prediction and I also use the same loss function, binary cross entropy with logits.

4.5 CLV - Supervised learning model

The final step of this methodological approach is to estimate the $\text{CLV}_i(t)$ for every customer i at time t. I propose using a Gradient Boosting model for the task. They are often the best candidate model in dealing with tabular data, the reason being their ability to model complex, non-linear relationships. Figure 4.3 is a simplified graphical representation of the methodology propose in this paper. In the last step, there can be other supervised learning tasks as well (e.g. in the context of customer transactional data, churn prediction, customer segmentation, etc.). Alternatively, more complex methodological architectures can be adapted, similar to the stacked approach proposed by Chamberlain et al. (2017).

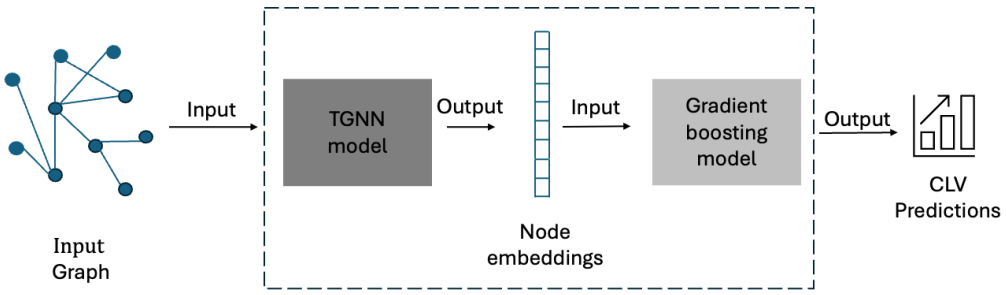

Figure 4.3: Graphical representation CLV prediction model. The graph representation of the customer transactional data is used as input for the TGNN model which outputs the embedding representation for customer nodes. Next, the obtained embeddings are used as features in the Gradient Boosting model to predict the CLV for every customer.

4.6 Evaluation metrics

In this subsection I introduce the evaluation metrics I consider for assessing the performance of the CLV prediction models. As discussed earlier, I consider this prediction problem to be a strong indicator of how well customers are represented in the embedded space. CLV prediction is a regression task aiming to predict a continuous value representing future customer spending. I consider multiple metrics that describe a model performance from different perspectives.

First, I consider Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.15}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{4.16}$$

where y_i is the actual CLV for customer i and $\hat{y}_i$ is the predicted CLV, for n total customers. I choose RMSE because its squaring of errors heavily penalizes large prediction errors, which are particularly undesirable for high-value customers. I include MAE as a complement because it is less sensible to outliers and provides a more straightforward interpretation of the average error magnitude, especially since CVL values are positive only values. Additionally, I consider root mean squared log error (RMSLE) which is also meant to counter the effect of larger prediction errors.

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2} \tag{4.17}$$

The logarithm transformation will scale down the large error before measuring the squared error. To be more exact, due to properties of the logarithms, the inner difference is actually the relative error, such that the scale of the error is less significant. Moreover, since log function is strictly increasing, it implies that underestimation in prediction ($y_i/\hat{y}_i > 1$) will receive a higher penalty than overestimation ($y_i/\hat{y}_i < 1$). This aspect, in particular, makes it a relevant error measure in the context of CLV prediction for the Online Retail data set. I observe that the majority of the customers have zero CLV value, such that it is only natural that we want to penalize the models more for underestimating the values of buying customers (the important customers) and overestimating for the zero values customers.

Given the highly skewed nature of CLV data, I compute the adjusted version of symmetric absolute percentage error (sMAPE). This metric is an alternative to mean absolute percentage error (MAPE) which is extremely sensible to zero true values. The adjusted versions of this metric is designed to handle the situations in which both the true value and the predicted value are zero. This metric aims to provide a scale independent understanding of the relative error, which is very useful considering that CLV distribution covers multiple orders of magnitude.

$$\text{adj sMAPE} = \frac{200}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i| + \varepsilon} \tag{4.18}$$

To assess the explanatory power of the models, I use the R^2 measure,

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{4.19}$$

where $\bar{y}$ is the mean of the actual CLV values. This measures the proportion of the variance in the CLV that is explained by the customer embeddings. It provides a measure of goodness-of-fit that can easily be compared among the models.

Finally, for a better understanding of how the models perform for different segments of customers, I consider quintile analysis. More exactly, I investigate the performance of the models across five different customer segments grouped by their real CLV value.

Chapter 5

Experiments and results

In this section I present the results of the experiments, in line with the methodology described in the previous section. The performance of the DTGNN and CTGNN models is focused on the CLV prediction problem applied to the *Online Retail II* data set. This in turn aims to demonstrate the validity and benefits of using temporal graph neural networks (TGNN) in obtaining embeddings for the unique customers found in the data. I evaluate the performance of this models against the widely used BG/NBD probabilistic model.

Based on the exploratory data analysis section, as a test period, the last 3 full three months of the data set were chosen. The evaluation is structured as follows: first, I present an overall comparison of model performance across the five error measures (RMSE, MAE, RMSLE, R^2 and sMAPE). Second, I provide a visual representation of prediction accuracy to intuitively grasp the differences in model behavior. Next, I conduct a detailed quintile analysis to understand how each model performs across the different segments of customer value. Finally, I conduct a diagnostic analysis of the temporal graph neural network models to better understand their performance differences.

5.1 Overall model performance

Table 5.1 shows the performance of the three main models. The proposed DTGNN model coupled with the LightGBM model, demonstrates a superior performance across almost all the error metrics. It achieves a significantly lower RMSE (767 vs. 932) and MAE (191 vs 295) when compared to the benchmark model. Furthermore, it's sMAPE of 80.23 is also substantially better, indicating a much lower symmetric average percentage error. This result is also strengthened by the lower RMSLE score (2.32 vs 3.58) which suggests that DTGNN mode is more effective at managing predictions across different orders of magnitude. On the other hand it is very clear that the CTGNN model is significantly under performing when compared with any of the other two models, across four out of the five metrics, prompting a more detailed investigation in a later section.

One interesting result is difference in the R-squared (R^2) scores. Here we see that the

Table 5.1: Overall Model Performance Comparison

Model	**RMSE**	**MAE**	**R^2**	**sMAPE**	**RMSLE**
BG/NBD (Benchmark)	932	295	**0.60**	146.99	3.58
DTGNN + LightGBM	**767**	**191**	0.58	**80.23**	**2.32**
CTGNN + LightGBM	1420	398	0.42	141.92	3.75

BG/NBD benchmark model has a higher R^2 than DTGNN model. This indicates that while the benchmark model has a higher average error, it is marginally better at explaining the overall variance in the target data. In other words, on average, the DTGNN prediction is likely to be closer to the real value. On the other hand, the distribution of the BG/NBD predictions are overall more closely distributed to the real values. Figure 5.1 provides a visual representation of this aspect. We can observe how the DTGNN predictions are on average closer to the real value, but overall, the BG/NBD predictions have a larger maximum value, closer to the real CLV.

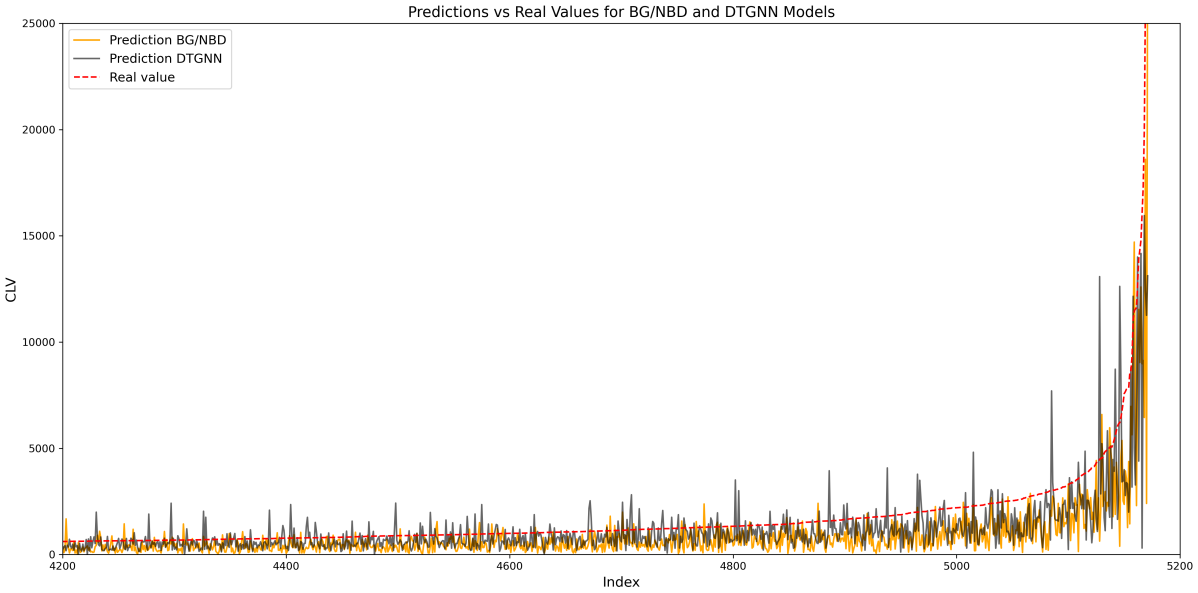

Figure 5.1: Visual representation of the BG/NBD benchmark model and DTGNN model predictions vs. the real values. The red line describes the 1000 most valuable customers ordered by their real CLV. The orange and the black lines are the predictions made by the models. We can observe that on average the black line is closer to the read line, but the overall larger prediction value corresponds to the BG/NBD model (the orange line)

Figure 5.2 complements the visual representation of figure 5.1 by showing how, on average the predictions made by DTGNN model are closer to the real values than the prediction made by BG/NBD model. This is also completely in line with the results reported in table 5.1.

For a more detailed understanding of these overall results, I split the customers into five

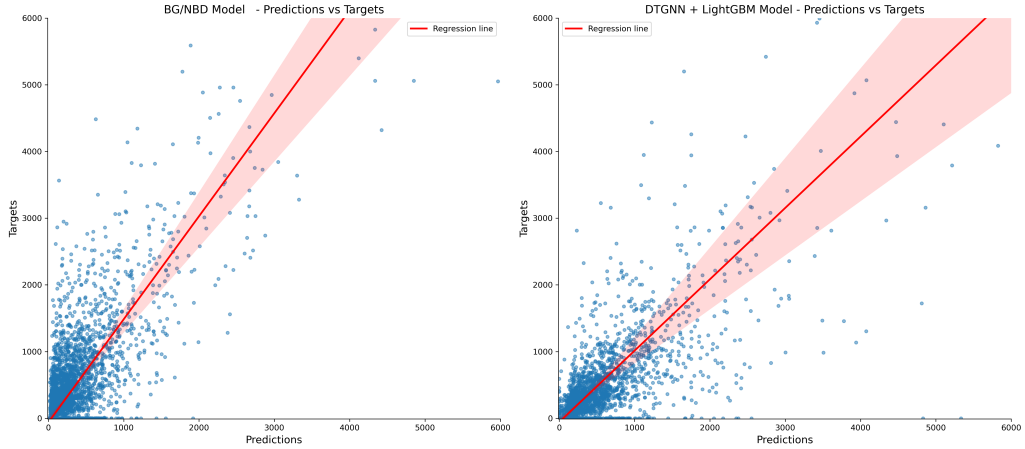

Figure 5.2: These two scatter plots are a visual representation of the predictive performance of the two models. Each point on the two graphs represents a customer with the prediction of the model on the x-axis and the real value on the y-axis. We can observe how the regression line on these values in the case of the DTGNN model (the right plot) is closer to the 45 degrees angle, which indicates that, on average, DTGNN predictions are on average closer to the real values than in the case of BG/NBD predictions (the left plot)

groups based on their true future values. It is important to mention that the groups are not equally numbered, since the vast majority of customers (approximately 3000) have zero real CLV. Thus, with slight abuse of naming, quintile 1 stands for the lowest CLV customer group, while quintile 5 stands for the highest value customer group. Table 5.2 shows the tabular results of such quintile analysis for the BG/NBD benchmark model.

Table 5.2: Quintile Analysis of BG/NBD Model Performance

	Targets	Predictions	Performance Metrics				
Quantile	Mean	Mean	RMSE	MAE	R^2	sMAPE	RMSLE
1	3.68	136.23	193.07	133.54	0.00	194.30	4.53
2	217.89	209.80	183.75	136.00	0.00	66.19	0.90
3	420.97	268.81	265.01	230.46	0.00	76.18	1.10
4	783.48	413.49	478.90	426.08	0.00	80.65	1.18
5	2598.31	1302.51	2850.01	1350.38	0.48	81.08	1.20

Figure 5.3 provides a visual representation of the overall quintile analysis. As expected, absolute errors increase for the higher-value quintiles, both RMSE and MAE. Although overall DTGNN model is the best performer, when looking at the first three quintiles we can notice how BG/NBD performs slightly better in terms of RMSE. This indicates that DTGNN has larger error for this groups of customer. But since MAE is still lower for these segments, it means there are few significant larger error that drive up the RMSE. This aspect can also be

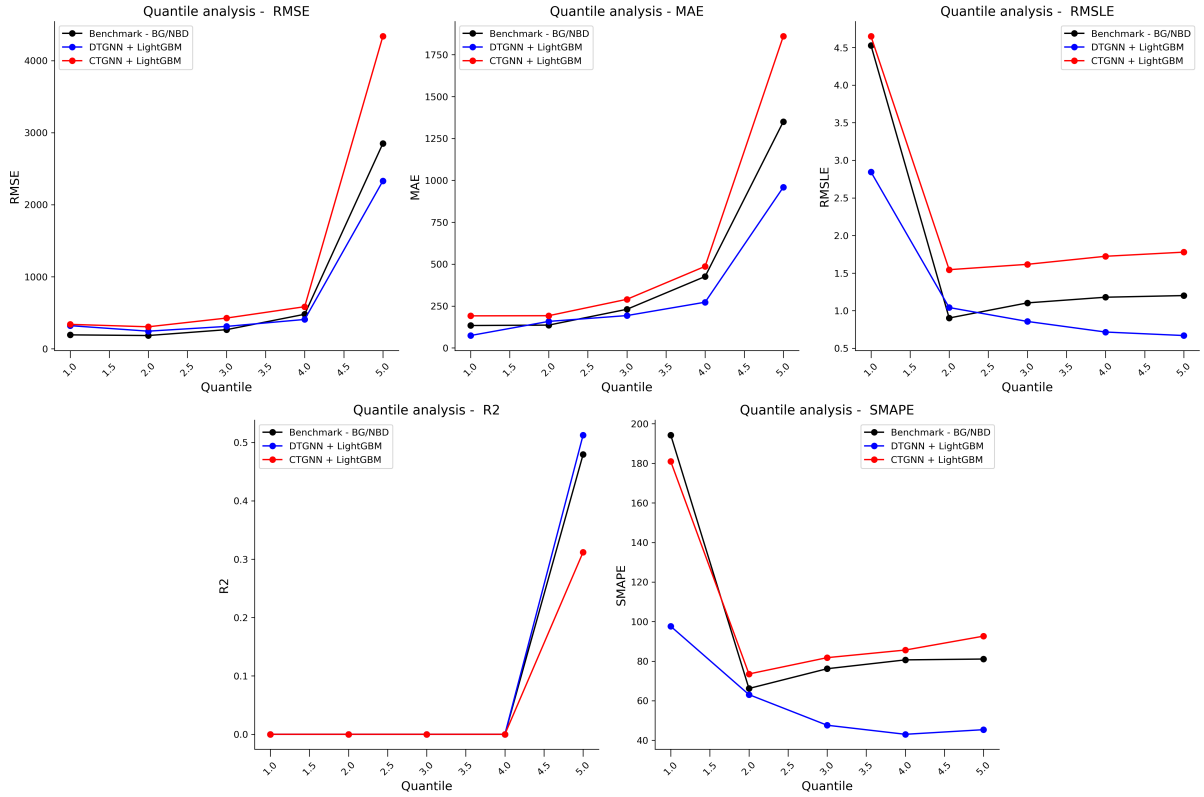

Figure 5.3: Quintile analysis of the model performance. Each of the 5 subplots shows the measurements of a specific error metric for each of the five customer groups.

confirmed by figure 5.2, where we can see there are few observations which have a larger than 2000 prediction for zero real values.

Looking at the sMAPE and RMSLE subplots, DTGNN displays again a clear advantage when compared with the other two models. The decreasing trend in both measures indicate that its predictions become relatively more accurate as customer value increases. The R^2 subplot is probably the most revealing. It shows how for the first 4 subgroups of customers, none of the models demonstrate any significant explanatory power ($R^2 \approx 0$). In the top quintile, all three models show a significant spike in R^2. Notably, the DTGNN model achieves the top R^2 result, surpassing the BG/NBD benchmark model. This observation implies that the BG/NBD marginal lead in the overall R^2 score might be a result of aggregation over the entire data set, whereas the more granular analysis shows that DTGNN is performing better for the most valuable group of customers.

5.2 Investigating CTGNN model performance

So far the results suggest that CTGNN model is significantly under performs when compared with both the DTGNN model and with the benchmark model. One potential explanation might be that, the continuous temporal graph representation is a to granular representation for customer transactional data. More exactly, I intuitively expect that due to the sparsity in

event for a given customer, but also to the considerable number of one time only customer, representing data at an hourly level is to noisy. Thus, I test this intuition by applying CTGNN model to aggregated data on a weekly and monthly level.

Table 5.3: Performance Comparison of CTGNN Aggregation Strategies

Model	RMSE	MAE	R^2	sMAPE	RMSLE
BG/NBD (Benchmark)	932	295	0.60	146.99	3.58
DTGNN + LightGBM	**767**	**191**	0.58	**80.23**	**2.32**
CTGNN Daily	1420	398	0.42	141.92	3.75
CTGNN Weekly	864	329	0.52	138.71	3.66
CTGNN Monthly	1082	345	0.42	146.13	3.55

The results of these diagnostic experiments are displayed in table 5.3 and figure 5.4. Both aggregated variants dramatically outperform the original CTGNN model in terms of RMSE and MSE measures. The weekly aggregation strategy appears to be the most effective across all measures. In particular, the RMSE is reduced from 1420 to 864, which also improves upon the performance of the benchmark model. Looking closer, the quintile analysis in figure 5.4 shows how the CTGNN model applied to weekly aggregated data is on par with the benchmark model in terms of RMSE and MSE scores.v

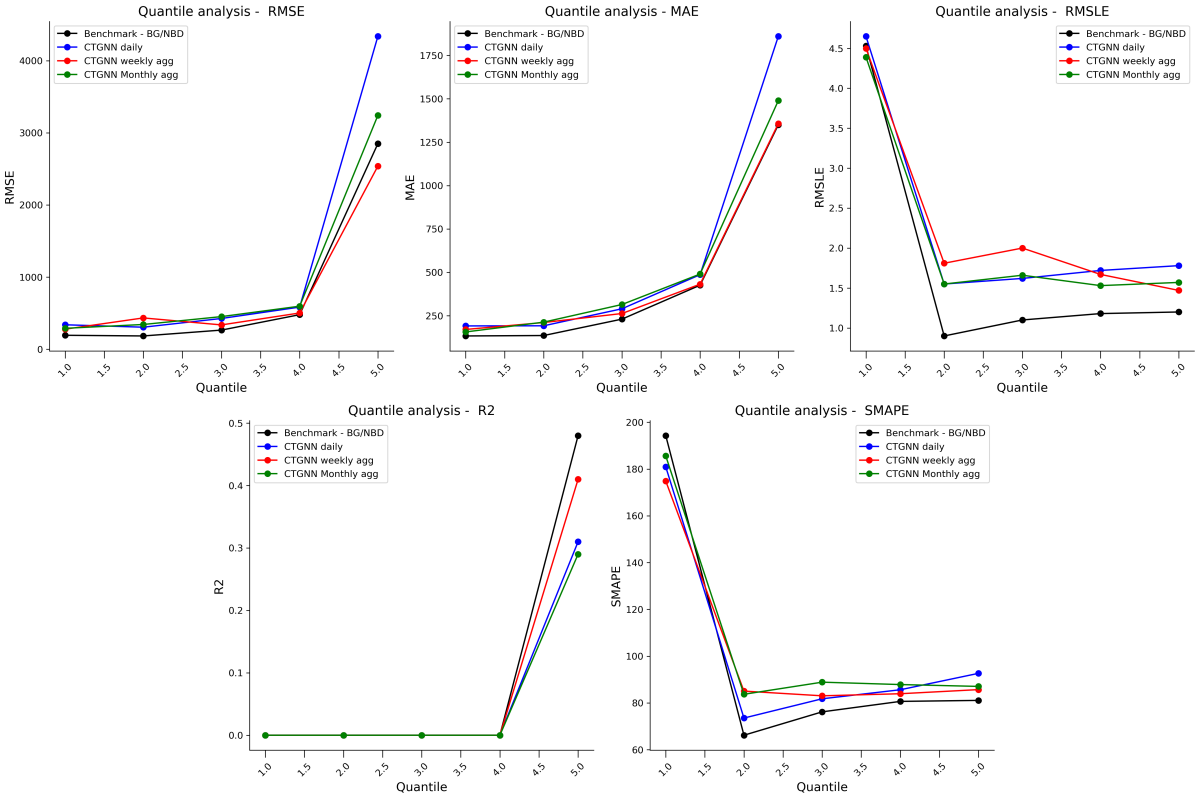

Figure 5.4: Quintile analysis of the model performance. Each of the 5 subplots shows the measurements of a specific error metric for each of the five customer groups.

Moreover, for all other measures is the best performer for quintile 5, the most valuable group of customers.

This finding suggests that the failure of the initial CTGNN models was not due to an inherent flaw in the continuous time temporal graph modeling concept, but rather a domain specific issue. More specifically, the customer transactional nature of the data induces a signal-to-noise ratio issue. The raw, event-by-event data is likely too noisy for the model to learn stable patterns. By aggregating transactions, the model has access to less noisy temporal signal, in a way, similar to the signal that is used in the DTGNN model.

The question remain to how come the DTGNN model is still outperforming the CTGNN model. Most likely the answer lies in the richness of features used in building the discrete temporal graph. Like stated before, the DTGNN model allows for dynamically changing customer attributes (e.g., RMF features, lag revenue etc.), while the CTGNN allows only for static customer attributes (e.g. country of the customer). So, in essence the DTGNN model is build upon hand crafted features, while CTGNN relies entirely on temporal signal from the interactions between customers and products.

Chapter 6

Discussion and Conclusions

The purpose of this research paper is to investigate the validity and benefits of using graph structures and temporal graph neural network models in obtaining vector representation of customer transactional data. To validate the benefits this approach, I consider CLV prediction as a subsequent supervised task for the embeddings to be used as input features. To the best of my knowledge, there are no previous research that investigate the use of graph neural networks for that purpose. I consider two distinct approaches in building the graph structure, discrete temporal graph and continuous temporal graph. Using graph neural network models build on top of these two distinct structures, I experiment on the open source *Online Retail II* data set.

The overall results show a clear superiority of the discrete temporal graph neural network model (DTGNN), coupled with the LightGBM model. This is a stacked CLV prediction model, which yields performances similar to those reported by Chamberlain et al. (2017). These empirical findings show that temporal graph neural networks can be a viable approach in improving CLV predictions, when compared with traditional probabilistic models like BG/NBD. Thus, the first contribution of this paper is to add to potential use cases of temporal graph neural networks and to enrich the CLV literature. In that sense, further research could also be aimed at investigating more complex stacked architecture to improve the prediction capabilities of the models. In the designing the experiments, I focus on exploring the predictive power of the embeddings obtained through the self-supervised learning performed by the temporal graph neural networks. However, following methodologies similar to those proposed by Bauer and Jannach (2021) and Chamberlain et al. (2017), the initial features could also be used alongside the embeddings as input in the prediction models.

The second important contribution is related to the continuous temporal graph neural network (CTGNN) model. The empirical results partially contradict the findings of Rossi et al. (2020). It seams that high temporal granularity is detrimental in the context of customer transactional data, and a discrete time representation helps in better capturing the signal and in filtering out the noise in the data. However, I do acknowledge an important advantage of CTGNN over DTGNN, namely that there is little feature engineering required for the continuous implementation. Furthermore, aggregating data on a weekly bases brought significant

improvements to the model, which suggest that further investigation could help with better understanding the impact of time granularity on this type of model performance. On the other hand, discrete temporal graph implementation allows for dynamically changing the customer node attributes, which require complex feature engineering and domain knowledge. In that sense, the two models are a case of trade off between model complexity and model performance.

Another important empirical finding comes from the quintile analysis. The R-squared plot shows that none of the models can explain any significant portion on the bottom 90% of the customers. The entire predictive power of all models is concentrated on their top 10% customers. This confirms again the difficulty of CLV prediction problem, but it also might be an indication that future research could focus on improving the prediction power of the models in this region of the customer base. This might lead to significant business and economic implications in practice.

One important limitation of the TGNN approach is the black-box nature of the models. This implies an inherent lack of interpretability, which offer traditional models like BG/NBD a big advantage in the face of more complex models like these.

In conclusion, this research successfully demonstrates that temporal graph neural networks represent a strong alternative for CLV prediction. Subsequently, they are a viable approach in obtaining vector representation of customer transactional data that could be used in any supervised learning task.

Bibliography

Alves Gomes, M., Meyes, R., Meisen, P., and Meisen, T. (2024). It's Not Always about Wide and Deep Models: Click-Through Rate Prediction with a Customer Behavior-Embedding Representation. *Journal of Theoretical and Applied Electronic Commerce Research*, 19(1):135–151.

Bauer, J. and Jannach, D. (2021). Improved Customer Lifetime Value Prediction with Sequence-To-Sequence Learning and Feature-Based Models. *ACM Transactions on Knowledge Discovery from Data*, 15(5).

Chamberlain, B. P., Cardoso, , Bryan Liu, C. H., Pagliari, R., and Deisenroth, M. P. (2017). Customer lifetime value prediction using embeddings. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume Part F129685, pages 1753–1762. Association for Computing Machinery.

Colombo, R. and Jiang, W. (1999). A stochastic RFM model. *Journal of Interactive Marketing*, 13(3):2–12.

Daqing Chen (2012). Online Retail II [Dataset]. UCI Machine Learning Repository.

De Caigny, A., Coussement, K., Verbeke, W., Idbenjra, K., and Phan, M. (2021). Uplift modeling and its implications for B2B customer churn prediction: A segmentation-based modeling approach. *Industrial Marketing Management*, 99:28–39.

Fader, P. S., Hardie, B. G., and Lee, K. L. (2005). "Counting your customers" the easy way: An alternative to the Pareto/NBD model. *Marketing Science*, 24(2):275–284.

Gadgil, K., Gill, S. S., and Abdelmoniem, A. M. (2023). A meta-learning based stacked regression approach for customer lifetime value prediction. *Journal of Economy and Technology*, 1:197–207.

Hamilton, W. L. (2020). Graph Representation Learning. Technical Report 3.

Kumar, V. and Pansari, A. (2016). National culture, economy, and customer lifetime value: Assessing the relative impact of the drivers of customer lifetime value for a global retailer. *Journal of International Marketing*, 24(1):1–21.

Mallick, T., Balaprakash, P., Rask, E., and Macfarlane, J. (2019). Graph-Partitioning-Based Diffusion Convolutional Recurrent Neural Network for Large-Scale Traffic Forecasting.

Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Koh, E., and Kim, S. (2018). Continuous-Time Dynamic Network Embeddings. In *The Web Conference 2018 - Companion of the World Wide Web Conference, WWW 2018*, pages 969–976. Association for Computing Machinery, Inc.

Ni, Y., Ou, D., Liu, S., Li, X., Ou, W., Zeng, A., and Si, L. (2018). Perceive your users in depth: Learning universal user representations from multiple E-commerce tasks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 596–605. Association for Computing Machinery.

Panagopoulos, G., Nikolentzos, G., and Vazirgiannis, M. (2020). Transfer Graph Neural Networks for Pandemic Forecasting.

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., and Bronstein, M. (2020). Temporal Graph Networks for Deep Learning on Dynamic Graphs.

Rozemberczki, B., Scherer, P., He, Y., Panagopoulos, G., Riedel, A., Astefanoaei, M., Kiss, O., and Beres, F. (2021). PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, pages 4564–4573.

Schmittlein, D. C., Morrison, D. G., and Colombo, R. (1987). Counting Your Customers: Who Are They and What Will They Do Next? Technical Report 1.

Seo, Y., Defferrard, M., Vandergheynst, P., and Bresson, X. (2016). Structured Sequence Modeling with Graph Convolutional Recurrent Networks.

Skarding, J., Gabrys, B., and Musial, K. (2021). Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey. *IEEE Access*, 9:79143–79168.

Vanderveld, A., Pandey, A., Han, A., and Parekh, R. (2016). An engagement-based customer lifetime value system for E-commerce. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August-2016, pages 293–302. Association for Computing Machinery.

Yan, Y. and Resnick, N. (2024). A high-performance turnkey system for customer lifetime value prediction in retail brands: Forthcoming in quantitative marketing and economics. *Quantitative Marketing and Economics*, 22(2):169–192.