Improved Customer Lifetime Value Prediction with Sequence-To-Sequence Learning and Feature-based Models

JOSEF BAUER, University of Klagenfurt, Austria
DIETMAR JANNACH, University of Klagenfurt, Austria

The prediction of the Customer Lifetime Value (CLV) is an important asset for tool-supported marketing by customer relationship managers. Since standard methods based on purchase recency, frequency and past profit and revenue statistics often have limited predictive power, advanced machine learning techniques were applied to this task in recent years. However, existing approaches are often not fully capable of modeling certain temporal patterns that can be commonly found in practice, such as periodic purchasing behavior of customers. To address these shortcomings, we propose a novel method for CLV prediction based on a combination of several machine learning techniques. At its core, our method consists of a tailored deep learning approach based on encoder-decoder sequence-to-sequence recurrent neural networks (RNNs) with augmented temporal convolutions. This model is then combined with gradient boosting machines (GBMs) and a set of novel features in a hybrid framework. Empirical evaluations based on real-world data from a larger e-commerce company and a public dataset from the domain of online retail show that already the sequence-based model leads to competitive performance results. Stacking it with the GBM model is synergistic and further improves accuracy, indicating that the two models capture different patterns in the data.

CCS Concepts: • **Information systems** → *Decision support systems*; *Business intelligence*; • **Computing methodologies** → *Neural networks*;

Additional Key Words and Phrases: Customer Lifetime Value, Machine Learning, Neural Networks

ACM Reference Format:
Josef Bauer and Dietmar Jannach. 2021. Improved Customer Lifetime Value Prediction with Sequence-To-Sequence Learning and Feature-based Models. *ACM Trans. Knowl. Discov. Data.* 1, 1 (May 2021), 38 pages. https://doi.org/10.1145/3441444

1 INTRODUCTION

The Customer Lifetime Value (CLV) is a common business metric used in Marketing and Customer Relationship Management [25, 28]. With the CLV metric, the goal is to assess the future, long-term monetary value of existing customers, most commonly in terms of the predicted net profit induced by customers during their entire lifetime.[1] The resulting CLV predictions can then be used as a basis for a number of managerial decisions, in particular for ones related to investments for customer acquisition, customer retention, and churn prevention [22]. As a side effect, considering

[1] According to [5], the CLV represents the "*net present value of the profits linked to a specific customer once the customer has been acquired, after subtracting incremental costs associated with marketing, selling, production and servicing over the customer's lifetime.*"

Authors' addresses: Josef Bauer, University of Klagenfurt, Universitätsstraße 65-67, Klagenfurt am Wörthersee, 9020, Austria, josef.b.bauer@gmail.com; Dietmar Jannach, University of Klagenfurt, Universitätsstraße 65-67, Klagenfurt am Wörthersee, 9020, Austria, dietmar.jannach@aau.at.

1556-4681/2021/5-ART $15.00
https://doi.org/10.1145/3441444

the customer lifetime value can also help businesses focus more on long-term strategic aspects than solely on short-term profitability.

A variety of approaches to estimate the CLV have been proposed in the literature during the last decades. The typical factors that are considered in CLV calculations are based on past purchase statistics of individual customers. The most widely used models for example rely on recency, frequency and monetary value (RFM) statistics. Such comparably simple models are easy to apply in practice. However, they are often based on specific distributional assumptions [6, 15, 16, 21, 26], sometimes leading to limited prediction accuracy when the assumptions are not satisfied [24].

In recent years, the success of different machine learning (ML) techniques for various real-world applications has triggered an increased interest in applying such techniques for the CLV prediction problem and the related, but simpler churn prediction problem.[2]

There are different ways in which CLV prediction can be modeled as a machine learning problem. First, it can be considered as a regression problem. In such an approach, the prediction is based on a set of engineered features, which, for example, encode aggregate statistics about the profit that was made by an individual customer so far, the point in time of their most recent transactions, demographic characteristics of the customers, as well as domain-specific factors. The advantage of such models is that they are very general in the sense that a variety of different aspects can be encoded within such a general prediction framework, and that various ML techniques can be directly applied for learning.

An alternative way of looking at CLV prediction is to consider it as a *time series forecasting* problem. Specifically, we can try to use past transaction data to make predictions for the immediate next time slots in the future (e.g., for a few weeks), and then aggregate these predictions to obtain the overall CLV for a given time horizon in the future. Compared to approaches that directly model the CLV, e.g., as a sum of predicted profits per time-step, the advantage of time series modeling is that the sequential nature of the data is preserved. Therefore, certain patterns in the data like periodic repurchases can be more easily captured by using time series models than by trying to model these aspects with manually engineered features. Moreover, multivariate time series models have the advantage that they can potentially exploit fine-grained information about the customer's preferences, which are often collected over longer periods of time on online retail sites.

In this work, we propose a hybrid CLV prediction framework that leverages the advantages of both modeling approaches and demonstrate its effectiveness in two application scenarios from the e-commerce domain. The CLV predictions in our extensible framework are obtained by combining the predictions of a more traditional feature engineering approach and a time series modeling approach based on a novel deep learning architecture using Recurrent Neural Networks (RNNs).

The contributions of our approach can be summarized as follows:

- The presented work is the first that relies on RNNs for the CLV prediction problem.
- The proposed RNN-based model consists of a novel architecture with multiple encoder-decoder sequence-to-sequence (S2S) gated recurrent unit layers, where major ingredients for the effectiveness of the model are the use of both an extensive set of features as well as stacked temporal convolutions.
- Both prediction models consider a number of novel features, which have not been explored in the literature before.
- We demonstrate the effectiveness both of the novel RNN-based prediction model and in particular the combined model with two real-world datasets from the domain of online retail.

[2]We consider churn prediction as a simpler problem, because we are only interested in a binary outcome instead of a numerical estimate.

The results show that both models are important for the overall effectiveness of the method, as they are able to leverage different types of information.

- Even though some of the evaluated models incorporate domain-specific factors, our modeling approach and computation pipeline is more general and can be used to leverage various types of additional knowledge, including, for example, clickstream data. To ensure reproducibility of our work, we share all code used in the experiments online. The links are provided in the experimental evaluation section.

The paper is organized as follows. Next, in Section 2, we review existing formalizations of the CLV prediction problem and summarize common technical approaches. In Section 3, we provide a motivating example and give an overview of our approach, while Section 4 discusses its technical details. The outcomes of the empirical evaluations are provided in Section 5. The paper ends with a discussion of the findings and an outlook on future works.

2 PREVIOUS WORKS

2.1 Methods Based On RFM Statistics

Classical RFM models make use of historical recency, frequency and monetary value data of customers [9]. They are based on the following basic features:[3]

(1) Recency (R): The time since the last purchase.
(2) Frequency (F): The number of purchases made by the customer.
(3) Monetary Value (M): The total profit or revenue resulting from purchases by the respective customer.

One basic RFM prediction approach is to categorize customers into groups based on the quantiles of the R, F, and M attributes, and to then predict the future profits (or: CLV) for an individual customer by using the past average profits of the other customers in the same group [4].

The advantage of many RFM approaches is that they are easy to understand and implement. However, there are several disadvantages. First, pure RFM models are restricted to exactly these three predictors (R, F and M) and additional types of information are usually not considered. Second, the resulting models do not account for changes over time, like a rise in mean profit across all customers. And third, the quantile-based thresholds are fixed and selected with respect to each attribute individually, rather than being optimized based on the joint frequency distribution. As a result, we can obtain imbalanced groups with some R, F and M value combinations having only few data points, which can easily lead to overfitting.

2.2 Models based On Markov Chains

Some of the limitations of plain RFM segmentation approaches can be addressed by the use of Markov chain (MC) models. Such models do not try to directly predict CLV values, but in a first step rather aim to predict the customers' future behavior, i.e., their probable next states [35], leading to a sequential model. Specifically, the main questions often are to predict whether or not a repurchase is going to happen and how much the customer is going to spend.

Such MC models are often based on recency, frequency, and monetary value attributes like the described RFM approaches. Compared to traditional RFM models, however, MC approaches allow us to model customer groups at a more granular level and to consider transitions from one state into another in a probabilistic manner by constructing transition matrices. A potential disadvantage of plain MC approaches, however, is that they do not consider additional types of information that could be relevant to the problem, including, for example, which kinds of products a customer

[3] A comprehensive survey of traditional methods can be found in [21].

has bought. MC models also do not retain temporal information prior to the last state, due to the Markov property. This implies that potentially relevant information about states prior to the last state is not taken into account. Moreover, MC-based methods can be prone to overfitting since the natural ordering of values is not considered.

In terms of application scenarios, an MC-based method has for example been applied in [35] to design more effective marketing strategies. Its performance has however only been evaluated using simulated examples, whereas the evaluations in our present work are based on real-world data. In our experiments, see Section 5, we use the MC-based method from [35] as one of the baselines.

2.3 Negative Binomial Distribution (NBD) Models

In the research literature, a family of more sophisticated probabilistic approaches were proposed, which are—like some of the above-described MC-based approaches—driven by the intuition that the CLV prediction process can be split into two parts. The first problem is to predict whether or not a customer will buy again. The second prediction problem relates to the question how many orders and how much profit will be made.

In contrast to MC methods, the specific idea for this type of models is that each step in this process is based on a different distributional assumption, i.e., the purchasing process of every customer is treated as a realization of a certain probability distribution. One distribution models the probability that a customer is still active at a given time. Other distributions model the number of future transactions during the prescribed future time frame as well as the profit made by each customer. Based on these distributions, point estimators like the expected value can be calculated and the resulting numbers can then be used as an estimate for the CLV.

The most common approaches in this area are based on the negative binomial distribution (NBD). Examples of such approaches are the Pareto/NBD and the BG/NBD model [15, 16, 36]. These models also attempt to deal with the so called *censorship issue* [3], which is the problem that only incomplete data is available for customers that are still active at the end of the data collection period. Note that this is a general issue in application scenarios where the date of the (future) customer churn is unknown, especially in cases without long-term historical data.

NBD-based approaches have the advantage of being principled and intuitive. They are particularly accurate when the specific distributional assumptions are correct or approximately satisfied and when the CLV is not influenced by other hidden variables to a large extent. However, these assumptions are not always met in practice, which lowers the predictive performance of these models [24]. Moreover, additional predictor variables and the time series nature of the data are not considered in these models.

Performance comparisons of the methods described in [15] and [36] were originally conducted for an online shopping application scenario involving the sale of music, using the CDNOW dataset. This dataset is publicly available but too small for our purposes. These methods were later used in practice and were further investigated by other authors, e.g., in [24]. In our experiments reported in Section 5, we include the BG/NBD model—which can be seen as an advanced RFM model—as one of the baselines in our performance comparison.

2.4 CLV Prediction as a Supervised Learning Task

More recent CLV prediction methods extend the previously described approaches in different ways. First, they often use more advanced ML techniques, in particular ensemble tree methods. Second, to improve the performance of the predictions, additional features (in the sense of predictor variables) are used besides the classic recency, frequency and monetary value features.

The work presented in [38], for example, is based on a random forest model that uses a set of additional, “handcrafted” features in the context of an online bargain (deals) website. Their model

includes features based on engagement (conversion rates, e.g., via clicks in emails), location-based features (distance to location of the deals), demographic features, user behavior statistics like vouchers used and times between purchase and redemption, as well as traditional RFM features of varying timescales. Moreover, a quarterly adjustment process is applied and customer segmentation based on purchasing frequencies is performed.

A very recent approach—also based on a random forest model—is presented in [10], with an application in the fashion industry. Compared to [38], the model from [10], like ours uses more features and it in particular relies on features that are derived from user interaction logs (session logs) and the customers' return histories. Furthermore, one specific novel characteristic of [10] in that context is that it generates a *user embedding* based on the customer's product view history in the form of sequential clickstream data. The general idea is to use these representations of the users to assess the similarities of their interests and behavior. In our method, we will also use such embeddings, although in a slightly different form, based on purchase instead of view logs (see Section 4.2).

However, our method is also different from [10] in different ways. We, for example, directly predict the CLV and not churn, we use more features, and furthermore include an RNN in our architecture to capture the time series nature of the problem. Generally, the prediction of churn (rates) can represent a valuable input to the CLV estimation process. In our e-commerce application scenario, customer churn is usually not observed. Customers of e-commerce sites typically do not close their accounts and decide to not buy again on the site. In the datasets that we use for evaluation, information about churn is also not available, which is why churn is modeled implicitly in our approach when the future CLV is predicted to be zero at some point in time.

Overall, the advantages and disadvantages of machine learning-based approaches are as follows. First, one typical key advantage of such models is their often high prediction performance. This is in particular the case when various other predictor variables besides R, F and M can be used. Second, many machine learning-based methods do not rely on specific distributional assumptions, which have to be met when NBD-based models are applied. A potential disadvantage is that these models have higher computational demands and require a certain amount of additional data collection and feature engineering. Moreover, depending on the used learning approach, the resulting models might be difficult to interpret.

2.5 Time Series Forecasting Approaches

The sequential, RNN-based component of our model makes predictions about the development of multiple time series over time. The prediction of (multiple) time series has been investigated in depth in the literature in the past decades [8]. The technical approaches range from statistical techniques, e.g., based on an autoregressive integrated moving average (ARIMA) model, over various types of traditional machine learning techniques, to the most recent deep learning approaches, see [33] for a recent comparison. Several multivariate time series approaches [29] have been proposed for different applications in the literature as well, e.g., cash-flow prediction [31] and macroeconomics [27].

The recent interesting work of [20], addressing market basket prediction, shares some similarities with our method with respect to the used the concepts, although using a different approach and application setting. In particular, it addresses co-occurrence, sequentiality, periodicity, and recurrence for next basket prediction with the purpose to speed up shopping sessions.

From the various time series forecasting approaches from the literature, we include a traditional ARMA model (as well as ARIMA/SARIMA variants) as an additional baseline in our experiments. One of the reasons for their use is that a recent comparison in [33] revealed that such traditional models can lead to performance results that are competitive with more recent and more complex

models in certain domains. However, as our results will show, these traditional time series models have their limitations in our problem setting, since they are only able to consider each time series individually. We will discuss this aspect in more detail in Section 5.3.3.

2.6 Alternative Ensembling Approaches

A number of alternative ensembling models exist that could be used instead of those used in our approach. For example, in [40], *deep forests* ("gcForest") are introduced. This method aims to be a substitute to deep artificial neural networks, incorporating several ideas of the latter while being easier to train. Specifically, it is based on a cascade structure to enable representation learning and uses a multi-grained scanning approach. It could in principle be both used for sequential modeling as well as ensembling. In particular, this method appears promising as an alternative to our GBM-based stacking method. The approach of [40] has however not been applied to similar problems yet and the majority of datasets that were used in existing experiments contain images, audio or text. Furthermore, the method is specifically tailored to classification problems. Therefore, it has to be adapted and extended so that it can be applied to our time series based regression problem. We leave such extensions as future work.

3 OVERVIEW OF THE PROPOSED METHOD

The main novelty of our learning approach is that we make use of time series information of past customer behavior by means of a deep sequence-to-sequence RNN with temporal convolutions and dynamic features, which we combine with GBMs to achieve high prediction performance. In this section, we first provide examples that illustrate how considering multiple features over time can improve the predictions (Section 3.1). Afterwards, we provide a high-level overview of our method (Section 3.2).

3.1 Motivating Examples for Sequential Modeling

Figure 2 sketches parts of the information that one might have recorded for some Customer *A* and Customer *B* over a certain period of time. Every row of each table represents a feature and the numbers in each column represent the corresponding values of this feature for a certain period of time, in this case a week. Some of the features in the example are common in many application domains (e.g., the profit in $), whereas others are particularly relevant for our guiding application, an online shop offering goods for children (e.g., the number or the age of the children).

The columns, i.e., the weeks, are organized in ascending order. Likewise, the features that refer to the time since the first and the last purchase, as well as the child age, are also recorded in weeks. The number of orders and the number of items represent how many orders were made and how many different items were bought in the respective week. The Customer ID is also listed in this example, but it is treated in a different way. It is passed to an embedding method as described in Section 4.2, which returns a vector of values resembling a learned latent customer behavior vector.

The question marks represent unknown values in the future. In this example, the CLV should be computed for the limited time horizon of the next four weeks and calculated as the sum of the predicted values for these four weeks.

Looking at the profit numbers for *Customer A*, we can identify two characteristics that are typical for time series. First, *periodicity* seems to play a role. One possible explanation for this pattern could be that the customer replenishes the stock of consumables. Second, we can also observe a *trend*, resulting in an increasing profit over time. This second aspect of increased profit also seems to be related to the number of unique items bought per week. In practice, this might indicate that the customer purchases more and more supplemental products in addition to the consumables that are purchased every four weeks.

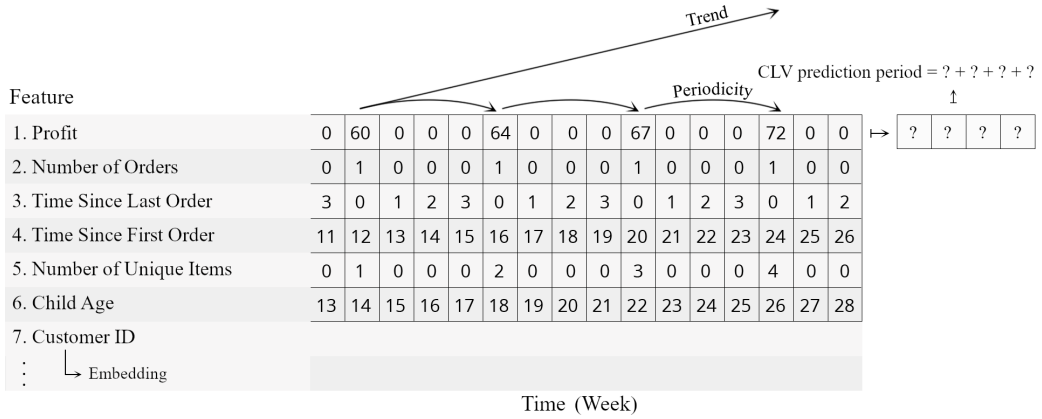

Fig. 1. Example for sequential features of Customer A.

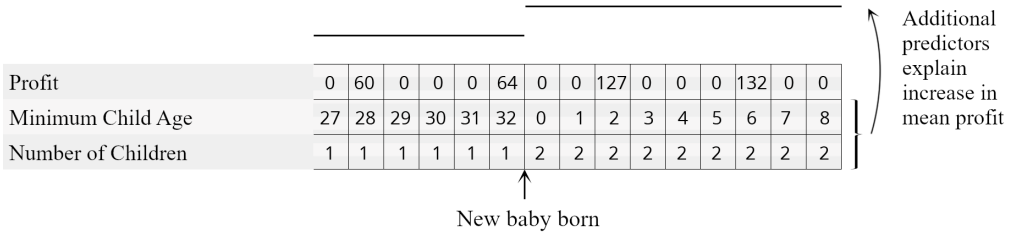

Fig. 2. Example for sequential features of Customer B.

Considering the periodicity and the trend signals, we would expect the profit in the four weeks to be approximately equal to $(0, 78, 0, 0)$. The zero entries are based on the periodicity pattern, and the predicted value for the second week is based on the trend which implies a slight increase in profit every month (here a slope value of 4 is added to the previous value of 74). Generally, one main goal of our method is to capture such time-dependent patterns—and more complex ones—automatically, without the need of manual adjustments or additional feature engineering.

Considering the data collected for *Customer B*, we can observe a sudden increase in profit from \$64 to \$127 starting from week 9. If we only consider the development of the profit feature over time, it is difficult to impossible to predict such an increase. However, if we consider additional time series information like the number and the minimum age of the children—which can be derived with high confidence from the viewed or purchased products or are voluntarily entered by the customer—we can observe that the birth of a new baby is the probable reason for this increase.

Such additional sources of information can be powerful predictor variables for the future interests and needs of a customer. Therefore, considering multiple time series in parallel, as done in our method, helps us predict the increase in profit before it actually happens. Note that some of the observed feature values might be caused by exceptional events, e.g., when the customer purchases

something valuable as a present for someone else. To account for such situations, our method includes a technique to deal with outliers and other types of noise in the data.

3.2 Main Ingredients and High-Level Architecture

3.2.1 Main Ingredients. From a technical perspective, our modeling choice can be summarized at a high level as follows. More details will be provided subsequently in Section 4.

- We use a tailored encoder-decoder sequence-to-sequence RNN, cf. [13] for basics on encoder-decoder RNNs, because it can learn such temporal patterns (like trends and seasonality) automatically, without the need of manual feature engineering. This is a major advantage over previous approaches, as it can be time-consuming and difficult to express all possible temporal dependencies through handcrafted features.
- According to the intuition of the aforementioned examples, we do not only consider profit sequences, but a richer set of sequential features that are relevant to the problem. Our training data therefore is not a matrix consisting of profit sequences, but a tensor consisting of sequences of a number of various features, some of which are shown in Figure 2.
- In practice, the resulting time series are often noisy and therefore it is beneficial to apply smoothing. We therefore use stacked temporal convolutions as one ingredient of our sequence-to-sequence RNN, as these can be seen as automated smoothing methods, like weighted moving averages.
- To further improve the prediction performance, we use a GBM model and combine it with our sequence-to-sequence RNN model by stacking. GBMs are in general well-performing for tabular data and are furthermore able to detect additional patterns in the data which are not captured by the RNN. Finally, GBMs are computationally less expensive than RNNs, which allows us to include more features given the same computing resources compared to our RNN model.
- Both for the GBM model and the RNN, we learn embeddings of the recorded purchase logs. These embeddings in particular help us to reliably perform clustering of similar customers also in data-sparse situations. Furthermore, to consider temporal aspects also in the input to the GBM model, we do not generate one single feature vector per user, but multiple vectors per user, one for each considered point in time (e.g., each point in time consists of the aggregate over a time frame of one day or one week).

3.2.2 Training Data Generation. One additional novelty of our approach compared to existing CLV estimation approaches lies in the way we process the available time series data in order to make sequence-aware predictions with our models. Figure 3 gives an overview of the principles of how we generate the training data for learning.

Defining Input and Output Sequences for the S2S RNN Model. The sequence-to-sequence model takes a series of values with defined length as an input and returns predictions for a subsequent sequence of values as an output. The length of the output corresponds to the estimated customer lifetime and has to be chosen depending on the domain and the available data. In the example, the output sequence length is set to four weeks for illustration purposes. The sum of these four predicted values is used as an approximation of the CLV. Note that such an estimation and limitation of the customer lifetime is usually necessary, since the actual lifetimes are often unobserved in practice.

The input sequence consists of the values observed immediately before the output sequence. The length of the input sequence, which is 12 in the example, can be considered as a tuning parameter. Depending on the domain, it can for example be meaningful to restrict the input length to focus

only on the more recent—and thus more relevant—input data. Note, however, that even though we use fixed-sized input sequences, information prior to the input sequence can still be considered as features in the model, as will be described below.

Sliding Window Approach. In order to make the model more robust, we do not only consider one input sequence and one output sequence, but use a sliding window approach. If we would only use one time-based split—i.e., if we only use the first block in Figure 3 for training—we run the risk of overfitting to a small part of the data. Therefore, we repeatedly go back by one time step and create input and output sequences of the defined lengths for earlier parts of the time series, leading to a number of additional training cases per customer. In each step back, we therefore consider an additional point in time in the past. The procedure is repeated until the defined beginning of the time series is reached.

Generally, using a sliding window approach for time series forecasting is a well established technique in the research literature. However, to our knowledge, it has not been used in this given form for the problem of CLV prediction.

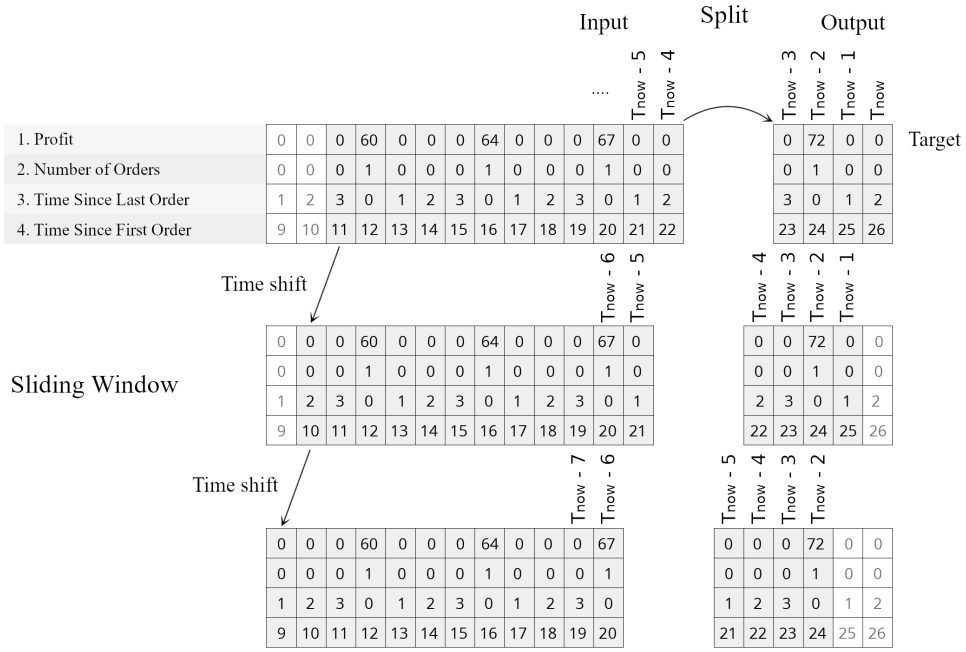

Fig. 3. Sliding window approach for training data generation for the sequence-to-sequence RNNs, shown for Customer A. T_{now} represents the current time. The training data is generated by successively shifting the window backwards in time. For every shift, the data points taken into consideration are shown in dark shaded color. The last four time steps are always the prediction target in this case.

Temporal Aggregation and Data Format for GBMs. Unlike RNNs, GBMs cannot process training data in time series form. Instead, "temporal aggregations" are needed, which are calculated for each point in time. In our approach, we consider temporal information in GBMs by applying various

aggregation functions on time-based features, e.g., the sum of past profit, but also the minimum, maximum and mean profit per order, statistics of times between orders etc.

Note that we do not only use these temporal aggregation features in our GBM model, but also in our sequence-to-sequence RNN model, which resulted in an additional performance increase. Remember however, that a main difference of the two models is that in the GBM model we directly optimize for the sum of the future profits, whereas we predict the elements of a time series with the RNN to better capture sequentiality in the data.

Figure 4 visualizes how we generate multiple training data points for each customer in the GBM model. In the example, we assume that the sum of the profit for a customer up to $T_{\text{now}}-6$ is 306. Note that this number cannot be derived from Figure 3, as Figure 3 only shows profits happening 9 points in time later than the first order of the customer (see the row "Time Since First Order"). We then observe a profit of 67 at $T_{\text{now}}-6$, as shown in Figure 3, leading to a sum value of 373 in the temporal aggregation Figure 4.

Time Step \ Feature	Profit	Profit Lag 1	Profit Lag 2	Profit Lag 3	Profit Lag 4	Sum of Profit	Number of Orders	Time Since First Order	Time Since Last Order	....
Tnow - 4	0	0	67	0	0	373	5	22	2	
Tnow - 5	0	67	0	0	0	373	5	21	1	
Tnow - 6	67	0	0	0	64	306	5	20	0	

Fig. 4. Temporal aggregation with time-dependent features shown for Customer A, which is done for all customers and stacked together row-wise. This is suitable to learn a GBM model and is also used in time series form in our S2S model.

3.2.3 High-Level Architecture. The high-level architecture of our approach and the process and information flow are illustrated in Figure 5. The input to the process are the different types of data ①, including the history of purchase transactions, metadata about the items (e.g., category information), customer data, etc. In a preprocessing phase ②, we compute the feature values that are later fed into the machine learning models and also train the embedding models from the sequential transaction data. The resulting feature values and embeddings are then used by the GBM model ③ and the RNN-based model ④ in the main training phase. The outputs of these models are afterwards combined in a GBM stacking model ⑤, which computes the final CLV predictions for every customer ⑥.

4 TECHNICAL DETAILS

4.1 Feature Engineering Approach

A main challenge in practical machine learning applications is the identification of a suitable set of predictor variables (features) for the given problem. Typical features for the CLV prediction problem, as mentioned above, are based on purchase statistics, e.g., past purchase amounts, the recency

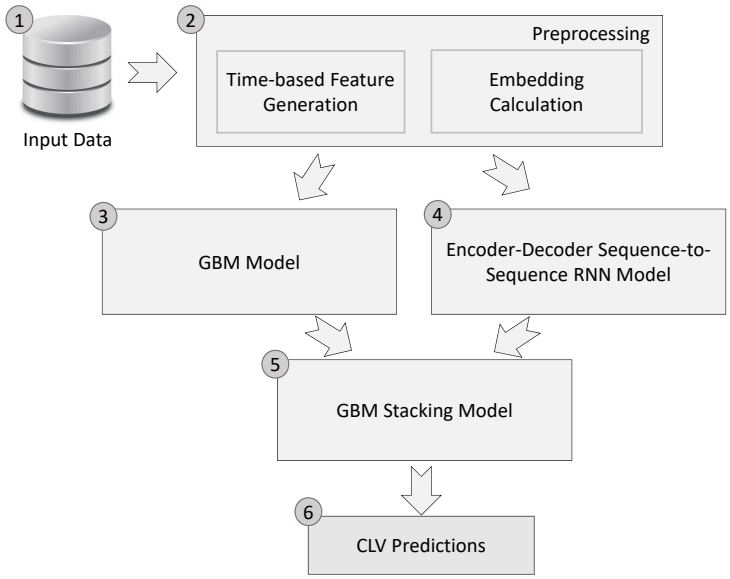

Fig. 5. High-Level Overview.

of the past purchase etc. Besides such more general features, there can also be domain-specific features, e.g., related to certain characteristics of the customer like the number of children in our main application scenario. Furthermore, whether or not certain features are used in a practical application depends on the relevance, availability and quality of the underlying data.

4.1.1 General Model. We formulate our task as a supervised learning problem which supports the integration of an arbitrary number of features, which is a common approach in machine learning. The generic problem formulation is to learn a function $y = f(\mathbf{x})$ that returns the predicted CLV value, where $\mathbf{x}$ is a vector of feature values. These features represent all quantifiable information about a customer that is relevant to the problem. Given such a generic problem formulation, various types of machine learning algorithms can in principle be applied to train a model that is able to generate predictions $\widehat{y} = f(\mathbf{x})$.

4.1.2 Used Features. We designed our hybrid CLV prediction approach in the context of a specific application scenario as described above. Therefore, the set of our predictor variables comprises not only more general features (for example, profit could be equal to usage time in an online service scenario, outside traditional e-commerce), but also a smaller set of features that are specific to the given problem. Many of the features, which we summarize next, have never been used in the literature for CLV prediction before, even though some of them are not limited to a particular application domain.

As explained in Section 3.2.2, a further unique aspect of our feature engineering approach is that we consider all features to be time-dependent. Most of the features we use are in fact dynamic, i.e., change over time. As shown in the example above, we do not consider the number of children of a customer as a *static* feature that is computed only once, but we derive the feature values for every point in time considered in the past, e.g., each day or week. The groups of features we present in the following should therefore be thought of being dependent on time. They are computed in the same way as shown in Figure 4.

The main features of our prediction model can be organized in three groups:

(1) Features based on customer attributes
(2) Features based on orders (item purchases)
(3) Features based on other item interactions

Note that most features of our models—in particular those that are related to orders and other types of customer behavior—represent *aggregated* values, e.g., the number of purchases or the total profit that was made. In principle, our general model also supports the usage of features at a finer *product* level. In such a model, each item on sale could represent a feature and the feature value indicates whether or not the customer purchased this product. In real-world cases, however, this would lead to very large feature vectors and the problem of data sparsity. For this reason, we will generate features only on the level of product categories, which also helps us address sparsity issues.

Features based on customer attributes: We consider both static features of customers as well as attributes that change over time. In our specific application, the estimated age and the number of children at every point in time are important domain-specific factors that influence the predicted profit in the future. The list of customer-related features is provided Table 1.

Table 1. Features based on Customer Attributes

Static and Demographic Features
• The country, residence and sex of the customer. • The customer's estimated income. • The channel through which a customer was acquired.
Dynamic Customer Attributes
• The minimum and maximum child age as well as the mean and the median child age (in days). • Aggregate statistics about the child ages when orders were placed, since these factors determine the range of relevant products. • The number of children, because a higher number of children is expected to correlate with a higher purchase probability. • In other applications, the ages of customers are more relevant than the ages of their children.

Features based on orders: This set of features is similar to the ones used in classical RFM models. In contrast to traditional RFM models, however, our feature set is not only richer but the features are again determined for different points in time. The specific set of features that was used in our experiments is shown in Table 2.[4] These features are not specific to the given e-commerce application domain (e.g., in a telecommunication provider scenario, one might have usage times instead of orders).

[4]Note that some of the features were not available for the online retail dataset, as we will describe in Section 5.1.

Table 2. Order-related Features

Features related to Recency and Frequency
• Total number of orders by the customer. • Number of orders within the last time period (for several time periods, e.g., the whole history, the previous year etc.), indicating how frequently the customer buys in recent times. • Number of days since the last order, which reflects recency. • Number of days since the first order, which informs us how long the customer has been in business with a company. • Lags of the number of orders (i.e., the previous number of orders for varying time steps in the past) and lags of the days since the last order (meaning how long it took in the past for the customer to order again). • Median, mean and variance of the number of days since order for all past orders (cf. the reasoning below). • Mean and standard deviation between order dates (as proposed in [10]). • Time difference statistics (gaps between orders), such as: – average (mean, median) and maximum number of days between orders as well as their variance; – number of days between the last order and the second last order; – number of days between the second last order and the third last order (and optionally further time differences).
Features related to Monetary Value
• Total profit and revenue made by the customer. • Profit made in the last time period (for several time windows of varying length, corresponding to running sums/means). • Revenue made in the last time period. • Lag variables of the profit and nonzero profit, where the lag of nonzero profit represents previous amounts of profit by the customer (e.g., the first lag of nonzero profit is the profit made in the last order of the customer). • Mean profit and mean revenue per order. • Minimum profit and revenue made in an order by the customer. • Maximum profit and revenue made in an order. • The variance of profit per order (assessing uncertainty). • The mean difference of profit per order (capturing a trend).

Features based on other item interactions: This final set of features serves two purposes. First, by considering various types of interactions with items on the shop, our goal is to (i) predict the likelihood that a customer will try new items at certain price levels, and to (ii) acquire general item popularity levels and incorporate repurchasing patterns in our model, because it is plausible that the repurchase probability is an important predictor for future profit.

Second, another part of this third set of features is used to deal with sparse data situations. Here, we do not consider item interactions at the individual level but aggregate past user behavior at the level of item categories. A detailed list of the used features is shown in Table 3.

Table 3. Interaction-related Features

Features related to Item Interactions
• Number of unique items a customer has purchased and the sum of all quantities over every item. • Number of unique subcategories from which the customer has bought a product. • Number of unique brands from which there was a purchase by the customer. • Mean, minimum and maximum price of the items that were purchased (capturing the value of the items typically ordered, like the cheapest and most expensive product a customer has bought). • Mean cost (for the vendor) of the items that were purchased. • Statistics of the number of orders an item appears in (across all customers), which aims to capture popularity (e.g., does the customer prefer to buy rare items?). • Statistics of the number of customers who have bought an item (with the same intuition). • Statistics of the absolute as well as the relative number of repurchases (e.g., the maximum mean number of repurchases is an important feature). • Statistics of both the absolute and the relative number of customers who reordered the respective item. • The maximum price difference of items the customer has bought. • Variances of the item prices and the relative number of repurchases.
Features to Deal with Data Sparsity
• For every item category, we model a feature that represents the number of orders the customer has placed that contained an item from the respective category. • Analogously, for every brand, we record the number of orders which contained an item of this brand. • Customer and product IDs, which are not used directly, but are converted into embeddings, as will be described in Section 4.2.

Novelty and already used features in the literature: From the various features presented in Tables 1–3, to the best of our knowledge only the following were used in previous work for CLV prediction problems:

- Traditional recency (time since last order), frequency (number of orders) and monetary value (net sales) features, but also their restriction to more current time windows instead of considering the whole history (like within the last quarter).
- The number of items bought by a customer.
- The age and country of a customer.
- The time between the first and the last order.
- The average order date as well as the standard deviation of order dates.

All other features presented above can be considered novel for our application domain. Lag variables have not been used for a CLV prediction problem in the literature. Their usage in other applications is, however, common. Likewise, for GBM models it is common to include categorical features which

are often passed in the form of sparse one hot features (i.e., encoding if a given attribute is "active" for a particular training case).

4.2 Learning Embeddings

Instead of directly using raw sequential data, researchers often rely on the use of *embedding* representations, in particular in sparse data situations. In the context of CLV prediction, Chamberlain et al. [10] learn embeddings from logged item view events to identify customers with similar interests. We rely on this method in our framework as well. However, instead of clickstream data, we learn embeddings from purchase data. In case clickstream logs are available, these can be used in our framework as well.

The intuition of this approach is illustrated in Figure 6 (compare Figure 7 in [10]). For each product, we determine the sequence of its purchasers, ordered by purchase time. Using this data representation, our goal is now to identify customers who have ordered similar products at a comparable point of their lifecycle. Technically, we use an adapted SkipGram model with negative sampling [34], known as Word2Vec in natural language processing. In our case, the customers correspond to words and the products can be seen as sentences in terms of the original NLP approach.

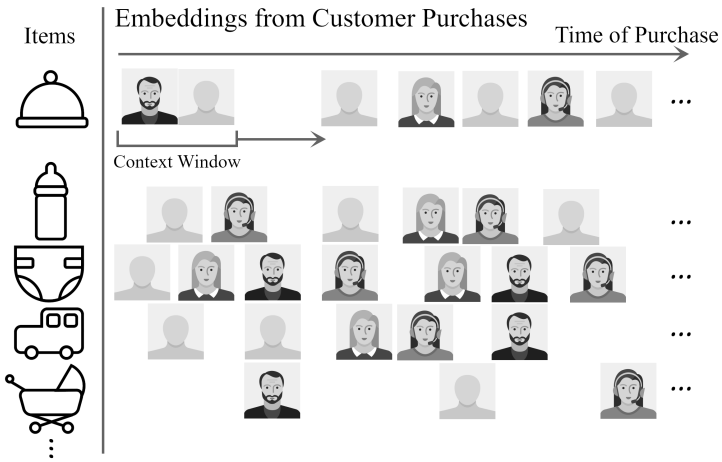

Fig. 6. Visualization of the SkipGram based customer embedding method with the usual context window adapted from [10].

We use the learned embeddings in both main models of our approach: (1) In the GBM model, we use the learned customer embeddings as features. As a result, there are as many additional features as embedding dimensions and the feature values correspond to those of the embedding vector for the customer. (2) In the sequence-to-sequence RNN model, we use an additional embedding layer with frozen weights (i.e., they are static during the main S2S training). The values of these frozen weights are set to the pretrained customer embedding values.

4.3 CLV Prediction With GBMs

The core of our hybrid approach, as sketched in Figure 5, consists of (i) a more traditional supervised learning problem formulation of the CLV prediction problem and (ii) an additional formulation based on a time series interpretation and a deep learning model. For the more traditional problem

encoding, we rely on Gradient Boosting Machines. GBMs, like Random Forests (RF), have shown to lead to high prediction accuracy in many domains. In addition, they are known to lead to generalizable models and are less prone to overfitting than other methods.

Random Forests were used for CLV prediction in [10]. They rely on training a larger number of trees—each on a resampled version of the original data obtained by bagging—as well as on selecting a random subset of features at each split. The goal of this procedure is to reduce the variance of the predictions from a bias–variance tradeoff point of view.

In our work, we use GBMs instead.[5] GBMs are also additive trees, but their main goal is to reduce the bias of the predictions. At each step of the training process, a new tree is added such that it minimizes the loss of the residuals by means of gradient boosting. It therefore focuses on improving the cases with the highest errors. As our experiments will show, the use of GBMs instead of RFs is advantageous in our problem settings with respect to accuracy.

Regarding the format of the input data, note that only the sequence-to-sequence RNN model is able to learn from the data in form of ordered sequences that are stacked together as tensors. The GBM always requires a single target value and a row of feature values for every training data point. For that reason, we have extended the training data as described above in Section 3.2.2 to enable the GBM to learn from the entire history. As highlighted in Figure 4, we do not only have one training data row for each customer, but one training data row for each point in history for that customer.

4.4 CLV Prediction With RNNs and Encoder-Decoder Sequence-to-Sequence Modeling

In the following, we describe our novel deep learning architecture based on RNNs to make predictions using the generated time series data. As sketched in Section 3, we follow an approach where we make profit predictions for individual time steps in the future (e.g., for every week), and compute the CLV as the sum of these predictions.

Our final neural network architecture is shown as a flow diagram in Figure 8. In the following subsections, we will first discuss its individual components.

4.4.1 RNNs and GRUs as Building Blocks. RNNs are a natural choice for creating deep learning models based on sequential data.[6] In our application scenario, models of this type help us to capture long-term trends in purchasing behavior, e.g., a steady increase in order volume over time, as well as periodic patterns. RNNs are neural networks whose weights are shared across time steps with the purpose to generalize with respect to time. Inputs to RNNs are the current time series value as well as the hidden layer values of the previous time step, which are in turn updated. Note that these hidden states are used because they are able to dynamically store information from arbitrary prior times. This would not be possible if we would instead use a fixed number of explicit previous time series values. The hidden states can be seen as a compressed summary of the time series, capturing its nature.

Technically, we use networks of gated recurrent units (GRUs) [13] as building blocks. GRU networks can be considered as a simplification of long-short term memory networks (LSTMs) [23], which retain most of the modeling capabilities of LSTMs while being computationally more efficient and being less prone to overfitting for smaller datasets [14]. Internally, GRU-based models have a recurrent structure with gates that control the flow of information and states that represent memories, allowing them to capture long-term dependencies that are problematic to be stored for usual RNNs, since gradients tend to either vanish or explode. By using GRU blocks, we are thus able to mitigate the vanishing and exploding gradient problem. Other ways of dealing with this

[5] We used the xgboost implementation [12] of GBMs in our experiments.
[6] See [17] for an introduction of the basic concepts of RNNs.

problem include our use of additional features based on temporal statistics, which helps keeping information when backpropagating over a longer time period. Likewise, aggregating on a less granular basis when meaningful (for the *Children* dataset in our applications) can be helpful, since the time sequences will be shorter and are therefore less prone to these problems. The general structure of GRUs is shown in Figure 7. Formally, the model can be expressed as follows. Suppose that (x_t, h_{t-1}) is the concatenation of the current time series input and the previous hidden state vector, which are passed to neural networks, leading to the following equations:

$$z_t = \sigma\left(W_z \cdot (x_t, h_{t-1}) + b_z\right), \tag{1}$$

$$r_t = \sigma\left(W_r \cdot (x_t, h_{t-1}) + b_r\right), \tag{2}$$

where W denote the weight matrices and b the biases in the neural network operations with corresponding subscripts. Note that z_t corresponds to the update and r_t to the reset gate neural network (NN). Their role and interaction is given as follows: First, a proposed new hidden state candidate $\widetilde{h}_t$ is computed by an NN and for its input, the previous states are partially reset by the NN that outputs r_t:

$$\widetilde{h}_t = \tanh\left(W_h \cdot (x_t, r_t \odot h_{t-1}) + b_h\right), \tag{3}$$

where $\odot$ denotes the Hadamard (element-wise) product. Finally, the new state h_t is an average of the previous state and the proposed new hidden state candidate, with weights computed by the "update-NN" (with output z_t):

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t. \tag{4}$$

Note that the function operations (sigmoid and tanh) are applied to every component. The individual networks act as gates and control which information is updated and what persists in memory.

The idea can therefore be summarized as follows: The "reset-NN" r_t determines the amount of the previous hidden state (from the last time step) that should be taken into account when calculating a new hidden state candidate $\widetilde{h}_t$ for the current time step. The edge cases are either to remember everything from each element of the hidden state (having a value of 1 for the component of r_t) or nothing (for a value of 0) when calculating the new hidden state candidate. However, this is not final. The second "update-NN" z_t is used to determine whether the newly calculated hidden state is actually meaningful, or whether it is better to partially adjust the final values of h_t to the previous hidden state vector h_{t-1} by using a weighted average. Again, for every component of z_t, a value of 1 corresponds to using entirely the newly calculated value of $\widetilde{h}_t$ without taking into account the past state, while 0 corresponds to entirely using the previous hidden state instead.

4.4.2 A Sequence-to-Sequence Modeling Approach. When using standard GRU or LSTM layers as implemented in libraries such as Keras[7] and TensorFlow[8], we can either make a prediction for a single time step or generate an output sequence that has the same size as the input sequence. This represents a major limitation in our problem setting. Often, the future period of time for which we want to make predictions is much shorter than the history of past sales of a customer, which may span several years. For instance, if we want to make CLV predictions for the *upcoming* six months, we could only take the *last* six months of transactions into account when using the default implementation.

To deal with this issue, we use a sequence-to-sequence model architecture, which has been successfully used in several fields such as machine translation in the past [13, 37]. Following such an approach, we first use an *encoder model* consisting of a GRU layer that learns a summary of the

[7] https://keras.io
[8] https://www.tensorflow.org

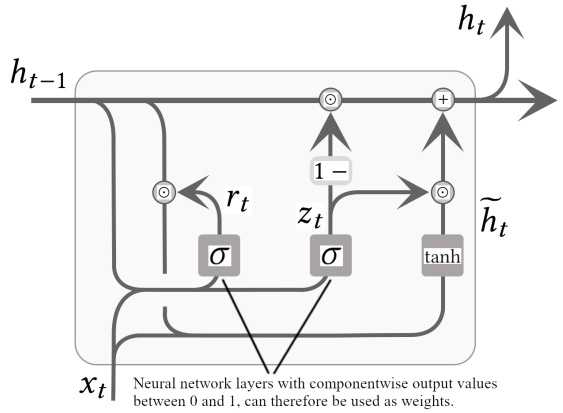

Fig. 7. Network flow diagram for GRUs with annotations. Dark grey boxes represent separate NN layers. The remaining blocks stand for operations (as stated below). The arrows show the directions of the information flow and processing steps. The layout of this diagram has been adapted from the graphics in http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

dependencies within the input time series which is saved in the hidden states. These hidden states are then passed as initial states to another layer of GRUs, the *decoder model*. The purpose of the decoder model is the generation of the output sequence, which represents the future profit for each time step (e.g., every week in the next six months).

Since we use the learned weights of the encoder GRUs as input, the information of a longer period in the past is implicitly included in the model. As a result, the model is no longer restricted to the most recent history (like the past six months). To make the model more robust and to be able to take into account all available training data, we furthermore employ a sliding window approach when generating the training data, as discussed in Section 3.2.2 and Section 4.6.1.

4.4.3 Adding Multiple Encoder Layers. Convolutional neural networks have been successfully applied in different domains, in particular for image recognition. One of the underlying ideas is that in such networks, each convolutional layer captures a different level of abstraction [17]. In our approach, we transfer this idea to the problem of learning patterns from time series data. We therefore extend our sequence-to-sequence approach with multiple layers of encoder GRU networks. The outputs of each GRU encoder layer are passed through a dense layer of corresponding input and output sizes to learn the optimal way of passing information.

In our experiments, we used two such encoder layers. The idea is that the first layer learns more fine-grained time series characteristics, while the second one is able to learn more general rules based on the summaries generated by the first encoder. The second encoder therefore has a regularizing effect. The number of hidden units in each of the layers are hyperparameters that have to be determined in the training phase using the validation data. Technically, the mapping to the output sequence by the decoder layer is achieved with a tensor slicing operation to match the sequence length as well as a time-distributed dense layer.

4.4.4 Incorporating Temporal Convolutions for Smoothing. Another important ingredient of our model is the use of multiple one-dimensional temporal convolutions, in particular causal convolutions. These can be seen as automated smoothing methods aiming to increase the generalization

ability of the model. They are particularly helpful in the presence of noisy observations. An illustration of this concept by means of an example profit sequence of a customer is shown in Figure 9. Given a defined window size, a set of kernels (filters) are learned that perform a specific form of averaging given the neighboring points. Differently from existing static or weighted moving average methods, the approach has the advantage that the averaging method is learned from the data and is not based on certain assumptions. For example, an exponential decay weight (like $(1, 2, 4, 8) \cdot 1/15$) would be a special case.

In our experiments, we found that using two chained convolutions to generate monthly and quarterly statistics led to an increase in performance, due to the improved robustness against overfitting when dealing with largely fluctuating data points. Generally, using multiple temporal convolutions corresponds to the idea that multiple levels of aggregations are learned.

4.4.5 Overall RNN architecture. Our overall architecture for the proposed sequence-to-sequence approach is summarized in Figure 8. We start by applying the two temporal convolutions to smooth the data with the previously engineered features, and combine the outcomes with the pretrained embeddings in a concatenated tensor. Then we have the two GRU encoder layers and the decoder layer. The time-distributed dense layer means that a dense layer is applied to every time step of the decoder output sequence. Using a tensor slicing operation, we finally create the output data, i.e., the sequence of the predicted CLVs. Note that the usual dropout and recurrent dropout are options that can be beneficial for the generalization ability of the model, similar to bagging in random forests. They are based on randomly switching off units.

In Appendix A, we provide further mathematical details of our model.

4.5 GBM-based Stacking

As a final step, we use a *stacking* technique to further improve the prediction accuracy of the CLV predictions. Model stacking can be considered as a meta-modeling technique where the predictions generated by the individual models are used as features in a higher-level model. In our approach, we use GBMs also for the stacking process. The motivation for using GBMs is that they are designed to make predictions based on rules. In our setup, a GBM could for example learn that RNN-based predictions are more accurate than GBM-based ones for customers with a higher number of past purchases.

Technically, our stacking procedure can be summarized as follows:

(1) Train both the GBM model (Section 4.3) and the RNN model (Section 4.4) on the training set of a given time-based training-validation-test split of the data.
(2) Generate predictions for each customer using both methods for the time frame that is covered by the validation set.
(3) Train a second GBM that uses the original set of features as an input but also the predictions of the first GBM and the RNN model to generate predictions for the time frame of the test set. The resulting model makes it possible to compare the values predicted by both (GBM and RNN) model types with the true values on the validation set. Thus, we can decide when it is better to use one type or the other.

Note that the second level GBM corresponds to the first level GBM with an augmented number of features. At the same time, the predictions of the first level GBM and the S2S RNN model are the most essential features. The remaining features are helpful to enable the model learn in what cases which model or combination works best. In fact, the GBM stacking model does not only distinguish between taking either the prediction of the GBM or the S2S model, but it can also learn a combination like a weighted average of the predictions. Further note that the GBM stacking

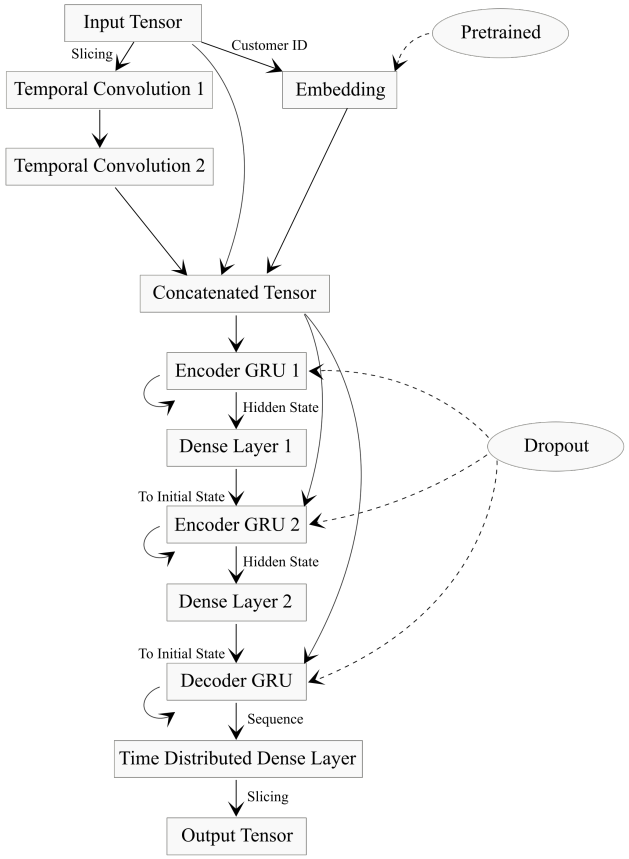

Fig. 8. Overview flow diagram of our sequence-to-sequence RNN architecture.

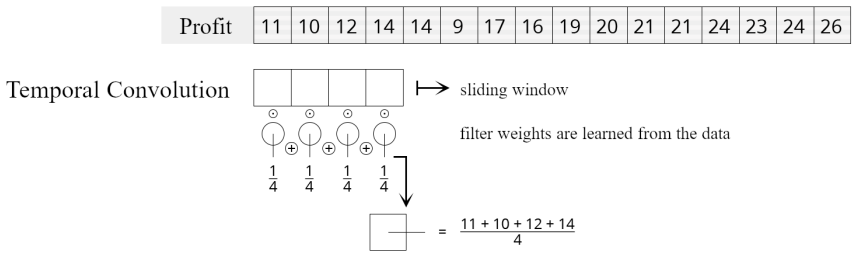

Fig. 9. Illustration of temporal convolutions on an example profit sequence. Note that temporal convolutions are not only applied to profit data, but a much richer set of features described in Section 4.1 to improve performance.

model is trained with the most recent data only, since data before the respective period is needed for training in order to create the predictions of the first GBM and S2S model.

The intuition behind this approach is the fact that both model types learn different patterns in the data and that model ensembling can improve performance by reducing variance. GBMs are good at finding rules in a huge set of features, leading to high accuracy for tabular data. But they are not tailored to sequential modeling, where the S2S model has advantages, as it can better learn sequence patterns. This combination aims to get the best out of both approaches.

4.6 Additional Implementation Aspects

In this section, we discuss two more implementation-related aspects of our method.

4.6.1 On-the-fly Generation of Tensors. The generated training data for our combined model using the sliding windows procedure can become large. We have therefore implemented an approach where we do not necessarily generate the entire training data—i.e., the various tensors for each customer—in advance. Instead, we dynamically generate tensors of training sequences on the fly to make sure that the data fits into working memory also for larger datasets. This is done by randomly sampling a time shift value, e.g., the number of weeks to go back in time, instead of computing the data for all possible shifts at once.

4.6.2 Dealing with Large Training Datasets. Our approach, which leverages the sequential nature of the data, is computationally more demanding than if we would use only one feature vector per customer and compute one CLV prediction. We not only make predictions for each time step in the future, but also consider multiple time series and use a sliding window approach to create multiple subsequences from the data.

The computational complexity in our problem setting in general is multiplicative in the input parameters, which are the number of customers, the number of features and the sequence length. When relying on a naive implementation, in particular the memory demands can be a limiting factor for the sequence-to-sequence-based model. For example, with one million customers, an input sequence size of 52 weeks, an output sequence size of 26 weeks, 10 sequential features and a total set of 128 training weeks, the memory demand would be over 100 GB RAM. In fact, memory requirements can already be problematic during preprocessing, since for building the feature matrix a cross join with all time points has to be performed in order to make zeros explicit.

To deal with this challenge, we take advantage of the fact that training data can be generated for every customer independently and does not need to be stored when this customer is not used for training in the current batch.[9] We therefore first process the training data for a subset of the customers, as we will describe below. We then train the neural network with the given subset of the data and store its weights. These weights can then be reloaded when we continue with the next customer subset. This can all be done in an exhaustive manner, such that all disjoint subsets are successively iterated through, to ensure that all customers are considered during training. Overall, this helps us to avoid to start from scratch every time. The approach is summarized in Algorithm 1. Note that the underlying optimization method is a modified version of mini-batch stochastic gradient descent, which allows for a better use of memory but retains its convergence properties due to the randomization steps [7]. Also note that in Algorithm 1, a number of rounds n_{tol} with a higher error is accepted (which is treated as a hyperparameter), due to the stochastic nature of the optimization method.

[9]By a batch we mean the data that is used for a mini-batch gradient descent update used in the learning of the sequence-to-sequence model; for stochastic gradient descent this would be just one sample, but in order to make use of GPU parallelization it is beneficial to increase this size.

Algorithm 1: Stepwise S2S-Algorithm

Load the customer embeddings trained according to Section 4.2 and randomly initialize the remaining weights of the S2S RNN network described in Section 4.4.
do
  Create K disjoint subsets of customers C_k and load the current weights of the S2S RNN network.
  for *Every* $k \in \{1, ..., K\}$ **do**
    (i) Generate the sequential training data for all customers in C_k by the approach outlined in Section 3.2.2,
    (ii) train the model with this subset of data according to the steps given in Section 4.4 and
    (iii) store the corresponding updated weights.
  end
while *Minimum validation error improved over the past* n_{tol} *steps*;

5 EXPERIMENTAL EVALUATION

This section describes the details of the experimental setup that was used to validate our approach as well as the obtained results. We developed a customizable and highly configurable software framework for the purpose of the evaluation. We share the code online to make our work reproducible.[10]

5.1 Data

5.1.1 Datasets. We evaluated our approach on two datasets, where one is publicly available and the other one is proprietary. Unfortunately, public datasets that include profit or revenue information, which are essential for CLV prediction, are generally scarce.

The larger dataset, termed *Children*, was shared with us by a European e-commerce company selling products for children and families, having over one million customers. The corresponding purchase data was recorded in a period of over three years. In addition, we validated our findings on a public[11], but smaller dataset, which we term *UKRetail*. The dataset consists of 541,909 purchase records by more than 4,000 customers of a UK-based company, which were recorded during a period of about one year [11]. For the smaller *UKRetail* dataset, we build our models based on daily data. For the large *Children* dataset, we used weekly aggregates instead to make the computational processes more efficient. For the same reason, only selected features of high importance were used in the RNN model for the *Children* dataset.

We created training, validation and test data splits as described above in a way that the validation and the test splits span equal periods of time. For the *Children* dataset, we used the last six months of the available data for testing and all other data for training and validation. For the smaller *UKRetail* dataset, we use time frames of four weeks.

5.1.2 Denoising. An analysis of the datasets revealed that both of them contain purchase records from wholesale clients, i.e., customers who are companies by themselves and resell the products they buy. Compared to the large majority of regular customers, the wholesale clients generate very large revenues. In order not to bias our results, we have focused on regular customers and excluded what we considered wholesale customers in our experiments. We applied empirically-determined

[10] The source code used in the experiments can be found at https://bitbucket.org/Josef-Bauer/clv_pred/src/master/.
[11] http://archive.ics.uci.edu/ml/datasets/online+retail

thresholds for both datasets. For the *UKRetail* dataset, we excluded 20 % of the customers with the highest sales values based on an analysis of the quantiles of the distribution. We also excluded 1 % of the customers with very low (negative) sales values. Negative revenue values are not natural and could indicate either fraud or artificial transactions by virtual debitors (e.g., stock movements in the logistics process). The *Children* dataset also includes purchases from wholesale customers, but only to a small extent. We therefore excluded only the 0.1 % of the customers with the highest and lowest sales numbers.

For the public *UKRetail* dataset, we additionally found unusually high and low values at the beginning and at the end of the data collection period. We can only speculate about the reasons for these anomalies, which were not discussed in previous literature. They could be the result of external factors like initial promotions when the site was launched, or the results of other data-related issues (e.g., data imports from existing systems) when the data collection began. The high values at the end of the collection period are also most probably the result of an external effect, like the consideration of additional sales channels. To account for these anomalies we did not include the five first and last sixteen weeks from the *UKRetail* dataset, based again on an analysis of the distribution. Note that we repeated the experiments mentioned below with the original data as well. Keeping the data leads to similar outcomes, with comparable trends but different values. Nonetheless, we believe that not including the questionable data leads to more reliable insights.

In Table 4, we summarize the statistics for both datasets before and after denoising. It is noticeable that the overall dataset size of the UKRetail dataset is significantly reduced by denoising. This is partially due to the smaller dataset size after filtering out anomalies as well as the high amount of transactions generated by wholesale clients. This shows the huge effect wholesalers have on this dataset. We also tested a weaker dataset filtering, for which the absolute errors of all methods increase substantially, but the relative performances remain comparable. Nevertheless, the denoising is justified and yields more stable results.

	Full Dataset	After Denoising
UKRetail dataset		
Number of customers	4,373	3,284
Number of records	541,909	144,863
Children dataset		
Number of customers	1,081,443	1,073,872
Number of records	10,596,431	10,196,814

Table 4. UKRetail and Children dataset statistics before and after denoising.

5.2 Experiment Configurations

5.2.1 Compared Approaches. We compared our approach with four alternative methods.

- *Adjusted Customer Mean Baseline (ACMB) and ARMA*: We use the *ACMB* as a simple baseline. It predicts the CLV for a given future time frame based on the past profit of the customer. The prediction is computed by adjusting the total profit made by a customer in the past to the length of the prediction time frame. We furthermore extended it to *ARMA* and related time series models, for which the *ACMB* can be considered a special case.
- *BG/NBD*: The model from [15], as described in Section 2.3. This can be considered as a traditional RFM model, since it is based on RFM attributes, but uses a more sophisticated distribution fitting approach. It allows for an easy computation and is the most popular RFM method in the literature.

- *Markov Chain (MC)*: As another traditional CLV prediction model, we include the sequential Markov Chain-based approach from [35] as another baseline, see also Section 2.2.
- *Embeddings-RF*: This is a recent method described in [10] which is based on embeddings and random forests. Since this method was evaluated in [10] on a dataset which is not publicly available, we will report the results using the information that was available in our datasets.

For all investigated methods, we determined optimal hyperparameter values for each dataset based on the validation data. For our own approach, which we term *GBM-S2S*, we will not only report the results for the combined model, but also the performance numbers of its components—the GBM-based model (termed *GBM)* and the sequence-to-sequence RNN model (termed *S2S)*—when used in isolation.

5.2.2 Evaluation Measures. We use the root mean square error (RMSE) and mean absolute error (MAE) of the predictions as our evaluation measures. Both summarize the difference between the CLVs predicted by a model and the actual, hidden values in the test set. The difference is that the RMSE measure penalizes larger deviations more strongly. This might be desirable in application scenarios where larger errors lead to a disproportionately higher risk for business or if it is especially important to detect high value customers. The advantage of the MAE is that can be directly interpreted as the average error in terms of money.

Note that in the research literature on time series forecasting, other measures than the MAE or RMSE are commonly used, such as the mean absolute value percentage error (MAPE). In our case, where we deal with a specific application domain, such percentage-based measures are less informative than those that work on the absolute monetary values, in $ in our case. Consider a customer whose CLV we estimate to be $5, while it is in reality $10, and another customer whose CLV is estimated to be $50, while it is in fact $100. The relative error would be the same, but the absolute numbers are much higher for the second case, and a wrong estimate for the second case represent a higher risk or loss for the company.

According to the definition by [5], the CLV is based on profit data. For the *Children* dataset, we computed the profit of each purchase by subtracting the costs known for each product (e.g., purchasing price for the vendor, shipping costs) from the revenues. For the *UKRetail* dataset, no information about costs is provided, which is why we used the revenue as an approximation for the profit.

The CLV definition of [5] also takes into account the discount rate of capital, i.e., the effect of the interest rate over time. In our evaluations, we used an interest rate of zero for two reasons. First, during the data collection period of the *Children* dataset, the risk-free interest rate in Europe was actually close to zero. Second, for the *UKRetail* dataset, the period of data collection only spans one single year, and the prediction time frame is therefore very short.

5.3 Results

In this section, we provide the outcomes of our experiments and discuss their implications.

5.3.1 Outcomes. The main results are summarized in Table 5 for both the *UKRetail* dataset and the *Children* dataset. Overall, our method(s) outperform all baselines in this comparison on both measures and for both datasets, and the lowest (best) RMSE and MAE values are in both cases obtained with the stacked combination of the GBM and sequence-to-sequence RNN model (*GBM-S2S*). All performance differences between our methods and the compared baselines are statistically significant based on a bootstrap test, with all p-values lower than 0.01 (most of them substantially lower).

For the smaller *UKRetail* dataset, the differences between our best-performing combined model (GBM-S2S) and its individual components (GBM and S2S) are not statistically significant, but showed a clear tendency ($p < 0.2$). For the *Children* dataset, however, the combined model is clearly advantageous and the observed differences are statistically significant with very low p-values ($p < 10^{-5}$). This confirms our assumption that GMBs and RNNs are able to consider different patterns in the data.

Method	*UKRetail* RMSE	*UKRetail* MAE	*Children* RMSE	*Children* MAE
Adjusted Customer Mean Baseline (*ACMB*) and *ARMA*[12]	72.6580	49.7365	21.0767	11.9316
BG/NBD [15]	69.1706	41.5405	20.1382	8.0887
Markov Chain (*MC*) [35]	68.8441	45.1282	18.0898	7.6918
Embeddings-RF [10]	68.8211	39.4804	17.4837	6.8601
Our GBM method (*GBM*)	66.2567	33.7816	12.5034	3.4230
Our Sequence-To-Sequence RNN (*S2S*)	65.3565	34.5502	12.1621	3.9071
Stacking of our GBM and Sequence-To-Sequence RNN (*GBM-S2S*)	64.9491	30.7953	11.6473	3.1366

Table 5. Experimental results for the *UKRetail* and the *Children* dataset

5.3.2 Discussion of the Experimental Results. Generally, we can observe that the Adjusted Customer Mean Baseline performs already quite well for both datasets. While a substantial gain in performance can be obtained with more advanced methods, the results for this baseline show that the simple heuristic of using past profit already yields useful predictions. While only a single feature is used in this baseline, it is directly related to the target. It is also intuitively plausible that the past profit is a key predictor for future profit.

The established *BG/NBD* model is advantageous when compared to the simplest baseline. This was expected given that this model (i) is more principled in the sense that it considers the underlying distributions and (ii) also includes recency and frequency information as features.

The Markov Chain (*MC*) model also makes use of recency and frequency information, and for the *UKRetail* dataset, it performs slightly better than the *BG/NBD* model with respect to the RMSE. For this dataset and metric its performance is almost identical to the more advanced *Embeddings-RF* method, while for the MAE the difference is bigger. For the *Children* dataset, it performs better than the *BG/NBD* method for both error measures. Generally, the advantage of the MC method over the *BG/NBD* model can be explained by the facts that (i) the distributional assumption of the *BG/NBD* model might not be perfectly satisfied, and (ii) that the *MC* method is sequential in nature.

The recent *Embeddings-RF* model of [10] achieves yet a higher performance. The difference is bigger for the *Children* dataset compared to the *UKRetail* dataset. The probable reason for this difference is the rather small dataset size of the *UKRetail* dataset. Embedding methods are known to yield higher benefits for larger datasets; therefore the gain obtained by using embeddings is rather limited for this dataset.

Finally, our proposed method leads to the best results, as mentioned above. Looking at our *GBM* and our *S2S* model individually, we observe that they exhibit a different performance depending on the error metric. While the *GBM* model obtains better values with respect to the MAE, the *S2S*

[12]Note that the Adjusted Customer Mean Baseline can be considered a special case of an ARMA time series model, see the discussion in Section 5.3.3.

model achieves the best RMSE. This difference can be explained by the fact that different patterns are detected in the data by both models, therefore yielding different types of predictions.

Stacking both models combines the best of both worlds and helps to further increase the accuracy of the predictions. In addition to the use of two types of models, the inclusion of a rich set of features—among them a small number of domain-specific ones in the case of the *Children* dataset—explain the success of the method.

5.3.3 Experiments with Traditional Time Series Models. Our experiments revealed that traditional time series models such as *ARMA* and its *ARIMA* and *SARIMA* extensions, mentioned in Section 2.5, did *not* improve the performance over the Adjusted Customer Mean Baseline *(ACMB)*. Note that the *ACMB* can be considered as an *ARMA(0, 0)* model, i.e., without autoregressive and moving average terms. This model actually led to the best results of all tested ARMA configurations. An exception only occurs in the case of a moving window evaluation (see the next section) for the *UKRetail* dataset, where overall a moving average *ARMA(0, 1)* model has a marginally lower error (about 0.05 lower RMSE than the *ACMB*).

Several factors contribute to the limited performance of these models in our application domain.

- First, *ARMA/ARIMA/SARIMA* models only consider *individual* time series, and each time series model is unable to take into account information from the other time series, e.g., from similar customers.
- Second, these models are usually univariate, and as such, they cannot take into account multiple features. We also tried multiple regression with *ARMA* errors, but this also did not help to improve the performance. This phenomenon can be explained as follows:

(a) One problem with these external regressors is that their values have to be known or estimated for the future, which limits their use. In our experiments, we tried to either carry the last observation forward, or used lags of the corresponding features, so that we have actual values available for the future period. Both approaches did not lead to improvements.

(b) Furthermore, these additional features are again only from the same customer belonging to the respective time series. In contrast, models like RNNs and GBMs consider the data of all customers and can find general rules.

(c) Finally, only linear effects are modeled, while nonlinearities can appear frequently as well. At the same time, since *ARMA* and its extensions only consider individual time series, they fit parameters separately to each of them, thus being more prone to overfitting.

5.3.4 Results for a Moving Window Evaluation. So far, our measurements were based on one single training-validation-test split of the data. In order to ensure that these results are not the outcome of this particular split, we made additional measurements in which we applied a "moving window" approach. In such a time series cross-validation approach, multiple training-validation-test splits are created, which span different time periods of the available data. This is done by successively going back in time the output sequence length (equal to the size of the validation set and the test set) multiple times. The measurements are then made for each time-based split and the average results are subsequently reported.

In our experiments, we created three time folds in order to have a minimum amount of training data for each data split. Remember that we need more than one training example per customer and that the input sequence length is twice as long as the output sequence in our experimental configuration. The results of these measurements are summarized in Table 6.

The results show that our main conclusions still hold, i.e., our proposed approach leads to the lowest RMSE and MAE values. There are, however, differences in the relative ranking of the baseline methods. In particular, the *MC* method performs slightly better than the more advanced

Method	*UKRetail* RMSE	*UKRetail* MAE	*Children* RMSE	*Children* MAE
Adjusted Customer Mean Baseline (*ACMB*) and *ARMA*	79.0620	54.4851	23.5596	12.2247
BG/NBD [15]	72.8156	53.9270	22.3143	11.3459
Markov Chain (*MC*) [35]	73.4645	45.9742	19.8856	8.2331
Embeddings-RF [10]	73.8194	49.6098	20.2490	7.6505
Our GBM method (*GBM*)	71.0029	35.6562	15.3135	4.4218
Our Sequence-To-Sequence RNN (*S2S*)	69.8316	36.4251	14.7842	4.8876
Stacking of our GBM and Sequence-To-Sequence RNN (*GBM-S2S*)	69.3053	31.8793	13.9548	3.9471

Table 6. Moving time window validation results for the *UKRetail* and the *Children* dataset

Embeddings-RF method in the experiments for the *UKRetail* dataset. Also, the *BG/NBD* works notably better than the other baselines in terms of the RMSE for this dataset, but not with respect to the MAE. The absolute values of the errors consistently increase for all methods, most likely due to less available data.

These findings confirm that the more advanced *Embeddings-RF* model is mainly advantageous when larger amounts of training data are available. In particular, for embedding methods based on Word2Vec (which is used in this method), strong effects of the dataset size on the performance were previously reported in the literature (see, e.g., [1]). Our proposed methods are also affected by this problem, but to a lesser extent, since we use two different model types and a richer set of features, so that the embeddings influence them less. However, the analysis shows that traditional methods can lead to decent results when the amount of available data is limited.

In case of the *Children* dataset, the results are closer to those of the last time fold reported in the previous section, but also with somewhat higher errors. The relative ranking stays mostly the same, except that the *MC* method is slightly better than the *Embeddings-RF* method with respect to RMSE, but not MAE. So the same effect regarding embeddings and training data can be observed, but to a lesser extent. This can probably be attributed to the fact that there are both more customers and the sequences span a longer period of time. Thus mitigating the data sparsity problem slightly.

While in general a moving time window evaluation is a better way for assessing performance, we need to stress that the data is getting very small in our case, due to a lack of historical records, especially for the *UKRetail* dataset. For this reason, the last time fold probably provides the best estimate for generalization performance with sufficient data (summarized in the previous Table 5).

5.4 The Effect of Temporal Aspects on Model Performance

In this section, we analyze the dependency of the model performance on temporal aspects. Specifically, we vary both the temporal granularity of the future time window as well as the prediction horizons.

5.4.1 Varying the Temporal Granularity of the Future Window. We first investigate the effects of varying the temporal granularity of the future time window, i.e, the prediction period. As outlined in the introduction, the *S2S* method, by default, generates predictions for individual time steps in the future (days for the *UKRetail* dataset and weeks for the larger *Children* dataset, due to memory constraints). The CLV is then estimated by summing over all individual predictions in the future period. In contrast, the other methods—including our *GBM* method—predict the sum in the future

period directly. The motivation for that is the fact that the *S2S* method is sequential. The only exception is the *ARMA* time series method. But in this case, the granularity in the output must be the same as in the input (e.g., a sequence of days). However, for the *S2S* method, it is possible to choose a different granularity in the output by modeling the aggregated output series. E.g., the highest granularity corresponds to the case when there is an input sequence on a day or week level, while the output sequence consists only of one element representing the sum in the whole future time period. This is the same case as for the other methods, which predict the whole future time period directly.

Note that since we only change the output aggregation, the time series based patterns in the input data are still preserved. E.g., the model can learn from periodic patterns like purchases every other week. Just instead of predicting those days or weeks separately, the model learns the sum over a longer time frame, the higher periodic values being implicitly included.

For the *UKRetail* dataset, we tested an output granularity of daily, weekly, bi-weekly and four-weekly data, the latter corresponding to the entire prediction horizon of 4 weeks. For the *Children* dataset, we used an output granularity of weekly, four-weekly, 13-weekly and 26-weekly data, the latter value again corresponding to the entire prediction horizon. However, for both datasets we found that there is only little difference in the performance for varying granularities of the future time window. The deviation of RMSE values is below 0.2 for the *UKRetail* and below 0.15 for the *Children* dataset. This also shows that the different choice of future granularities for the *S2S* method did not have much of an effect on the outcome. These findings indicate that both low and high output granularities can be used. Nevertheless, predicting on the lowest granularity can provide benefits in practice. No information is lost and this way one can also predict how much profit will be made on which days or weeks, unlike the whole time period only. For example, this would enable better planning for marketing activities.

5.4.2 Changing the Prediction Horizon. Next, we investigate the effect of the length of the forecasting period on accuracy. In general, accuracy can be expected to drop with increasing prediction horizons. This is also the case because our target is a sum of profit values in the future period, which tends to increase with longer periods. To confirm this intuition, we ran additional experiments in which we varied the prediction horizons. Since our objective in this paper is the estimation of the *lifetime* value, we chose quite long prediction horizons in our main experiments with the *Children* dataset. Given in particular the very limited time range covered by the *UKRetail* dataset, we focused on experiments with shorter prediction horizons (4 weeks for the *UKRetail* dataset, while the horizon is 6 months for the *Children* dataset). The time range restriction in both datasets strongly limits any further increase in the prediction range, since a minimum amount of data is needed for training. We therefore also analyzed the performance for shorter prediction horizons. The results are given in Tables 7 and 8 for the *UKRetail* and the *Children* dataset, respectively.

We can observe that the errors increase for increasing prediction horizons. At the same time, the relative performance of the methods stays mostly the same, including our methods. It is notable that for the lowest prediction range (e.g., 2 weeks for the *UKRetail* dataset), the differences between the methods is smaller. This could be explained by the fact that in a shorter time, fewer repurchases happen, which are less predictable. The result for the longest prediction range, in particular for the *UKRetail* dataset, could be an indication that the *S2S* method produces more stable predictions for longer time ranges. For the *Children* dataset, this difference is however less pronounced. Evaluating this for longer prediction horizons is a topic for future work, provided that suitable datasets can be obtained.

Method	Prediction Horizon		
	2 weeks	4 weeks	6 weeks
Adjusted Customer Mean Baseline (*ACMB*) and *ARMA*	49.8475 26.1126	72.6580 49.7365	101.5071 74.5073
BG/NBD [15]	49.0102 25.0682	69.1706 41.5405	91.9457 68.5682
Markov Chain (*MC*) [35]	49.6577 26.3751	68.8441 45.1282	92.1925 62.1353
Embeddings-RF [10]	49.2269 23.3044	68.8211 39.4804	87.7757 62.6608
Our GBM method (*GBM*)	48.1648 18.4127	66.2567 33.7816	86.0570 55.4492
Our Sequence-To-Sequence RNN (*S2S*)	47.9992 17.2934	65.3565 34.5502	84.2075 52.3944
Stacking of our GBM and Sequence-To-Sequence RNN (*GBM-S2S*)	47.8538 16.6043	64.9491 30.7953	84.1413 50.7433

Table 7. Results (RMSE first / MAE second row) for different prediction horizons for the *UKRetail* dataset

Method	Prediction Horizon		
	4 months	6 months	8 months
Adjusted Customer Mean Baseline (*ACMB*) and *ARMA*	15.7979 10.8975	21.0767 11.9316	24.7564 13.2912
BG/NBD [15]	15.7316 6.7495	20.1382 8.0887	22.6628 9.5631
Markov Chain (*MC*) [35]	13.8047 5.9651	18.0898 7.6918	20.6797 9.0803
Embeddings-RF [10]	11.9178 4.6429	17.4837 6.8601	18.8650 7.3209
Our GBM method (*GBM*)	8.5808 2.0739	12.5034 3.4230	14.1701 4.4588
Our Sequence-To-Sequence RNN (*S2S*)	8.0480 2.7621	12.1621 3.9071	13.4394 4.4419
Stacking of our GBM and Sequence-To-Sequence RNN (*GBM-S2S*)	7.6358 1.9859	11.6473 3.1366	13.0866 4.2980

Table 8. Results (RMSE first / MAE second row) for different prediction horizons for the *Children* dataset

5.5 Ablation Analysis

Our results reported so far showed that stacking the *GBM* and *S2S* models is advantageous, and they give insights about the performance of each model individually. In order to better understand how much each component of our sequence-based method contributes to the overall performance, we conducted an ablation analysis. In this analysis, we tested different variants of our model, in each of which one component was removed. The following variants were tested:

- *Without embeddings* is a model without the customer embeddings described in Section 4.2.

- In the *simplified architecture*, only one GRU encoder layer is used which is directly passed to the decoder without the further layers in between. Also, no temporal convolutions are used.
- The *only basic features* variant used a strongly reduced set of features. Specifically, it is based on recency, frequency and monetary value, as well as the customer embeddings (which were removed in the first ablation variant).

The average results over both datasets are shown in Table 9. To emphasize on the relative gains obtained by the different components, we used the results obtained by the best-performing baseline (*Embeddings-RF*) as a reference point, indicated by a zero in the column "Relative Performance" in Table 9. The full model correspondingly has a performance value of 1.

Ablation variant	Relative Performance
Full model	1.0000
Model without embeddings	0.8368
Model with simplified architecture	0.6056
Model with only basic features	0.4218
Embeddings-RF baseline	0.0000

Table 9. Average performance relative to the *Embeddings-RF* baseline and the full model when some components are omitted. The relative performance values are normalized between 0 (lowest performance – the baseline) and 1 (highest performance – the full model). Note that smaller values of the relative performance for an ablation variant suggest that the removed component is more important.

From Table 9 we can see that each of the components contributes in a rather balanced way to the overall performance. The additional features have the highest impact and the model architecture is the second most important factor.[13] The embedding part has a smaller effect, i.e., it only leads to a smaller performance decrease when omitted. Note, however, that in general there are interactions between all components. For example, there is a synergy between the features and a more advanced model, since the latter is capable of better learning interactions between those features.

5.6 Feature Importance and Interpretability Analysis

In order to better understand which features are the most important ones when predicting the CLV, we made some additional analyses. The results of this feature importance analysis for the *Children* dataset are summarized in Table 10. We show the top 20 out of 719 features. The importance values are based on the information gain measure, which reflects how much each feature contributes to the overall model across all trees.

The *traditional features used in RFM models*, i.e., the total profit in the past (M), the number of orders (F) and the days since the last order (R), turn out to be among the most important features. In addition to the profit, which is the target of our interest, the past revenue is an important predictor. This can be explained by the fact that profit and revenue are sometimes anticorrelated, with some products being sold with a negative profit margin. Product diversity features are also relatively important (having ranks 7, 12 and 13), which is consistent with existing literature [10].

We further observe that several others of the most relevant features are based on *sparse item information*, in particular the number of orders for certain subcategories and brands. For example, a high number of past orders of certain consumables such as diapers is a strong indication that the customer will order these again.

[13] Note that the relative performance values are inversely correlated with the impact of a component, since the performance is reported with this component removed.

Another important class of features are those based on *statistics of the past purchase frequency*. In particular, these are the features `medianDaysSinceOrder` and `meanDaysSinceOrder`, which are based on the number of days since the last purchase for each purchase event that happened in the past. Furthermore, the feature `meanGapBetweenOrders` describes the mean number of days between past orders. Similar features are also included in the list of features with a slightly lower information gain. These features, which capture some of the purchase timing patterns of customers, were a main inspiration for using our sequence-to-sequence RNN.

Our results also show that various other statistics such as the minimum, maximum and mean profit per order are helpful. This is due to the fact that they explain typical deviations from the traditional summaries of total profit and number of orders. Providing such ranges as input is useful for dealing with uncertainty.

Finally, the features `maxRelativeNumberOfRepurchases` and `maxRelativeNumberOfCustReordItem` are also intuitive. As described in Section 4.1, they are based on the maximum repurchase probability of all the items a customer has bought.

Generally, note that sometimes features with a comparably low gain value can actually still be quite helpful for the model. This is, for example, the case for *features related to the ages of the children* the customers have. They are not necessarily among the top features, because their values are missing for a large fraction of customers. Therefore, the GBM cannot perform splits on them for the majority of training cases, resulting in a lower gain. However, when this information is available, these features are helpful predictors.

Rank	Feature	Gain
1	totalProfit	0.1847
2	numberOfOrders	0.1498
3	totalRevenue	0.1078
4	brandNumberOfOrderIndicator.1242	0.0673
5	daysSinceLastOrder	0.0491
6	medianDaysSinceOrder	0.0412
7	numberOfUniqueSubcategories	0.0335
8	brandNumberOfOrderIndicator.1104	0.0228
9	subcategoryNumberOfOrderIndicator.155	0.0184
10	meanDaysSinceOrder	0.0154
11	minOrderProfit	0.0130
12	numberOfUniqueBrands	0.0128
13	numberOfUniqueItems	0.0114
14	subcategoryNumberOfOrderIndicator.16	0.0104
15	maxOrderProfit	0.0103
16	maxRelativeNumberOfRepurchases	0.0095
17	meanGapBetweenOrders	0.0085
18	subcategoryNumberOfOrderIndicator.3	0.0083
19	maxRelativeNumberOfCustReordItem	0.0082
20	meanOrderProfit	0.0078

Table 10. Feature importance values of the top 20 features from our experiments

At this point, note that in the context of *interpretable* machine learning alternative approaches exist to analyze how individual features affect the model outcomes. Our combined model is fairly

complex and not fully interpretable, especially in the *S2S* part. For this reason, it may be helpful in practice to resort to other methods that aim to provide interpretability for black-box models. In [30], the authors provide an extensive overview and taxonomy of such approaches. For instance, according to the classification of methods and metrics in [30], the data-driven and local SHAP method [32] should be particularly suitable for our problem. Specifically, this method can yield a more accurate generalization of feature importance values, which can also be applied to particular prediction examples (in contrast to the entire model).

Given our combined model, we could in principle investigate the first few decision trees generated by the GBM model, because these individual trees are interpretable. However, this would only lead to a rough approximation of the full model. At the same time, RNNs lack a similar possibility, which is the application of more advanced methods as described in [30] seem more helpful.

6 CONCLUSIONS AND FUTURE WORK

We will now summarize our approach and findings, provide ideas for new features, discuss its applicability to different domains and address promising technical extensions of our approach.

6.1 Summary of Our Method

We have proposed a novel method for predicting the Customer Lifetime Value by means of a customized sequence-to-sequence recurrent neural network and gradient boosting machines. A key aspect of our approach lies in capturing the time series nature of the problem, and we have demonstrated the benefits of our method based on two real-world datasets. In practice, applying our method enables an accurate detection of the most valuable customers, which in turn can be treated in a special manner by customer relationship approaches, therefore increasing profit and making companies competitive by strengthening their base of loyal customers.

Our modeling approach is based on a set of features that can be derived from basic information that is commonly available in practice. Furthermore, we have seen that the consideration of domain-specific aspects can be beneficial to achieve even higher prediction performance. While including domain-specific information requires some additional feature engineering, these features can be easily integrated into our pipeline by making use of the provided templates.

6.2 Additional Promising Features for CLV Prediction

While we have already introduced a number of new features to the CLV prediction problem, we identified several additional features, which we—based on our experience in related applications—consider helpful for the CLV prediction problem. These types of data were, however, not available in our datasets.

- *Out-of-stock* or *unavailability information of items* a customer preferred in the past or might prefer in the future: When we know that some of a customer's favorite items are not available, we can expect that the amount of money spent might be lower.
- *Evolution of prices and competitor prices*, especially for higher priced products a customer prefers: Such developments can strongly affect a customers' willingness to buy, depending on whether price differences to the competition are low or high.
- *Bank holidays* and indicators for *times before and after holidays, special seasons and global trends*: Holidays and other time-limited phenomena can affect the mean profit over all customers. Furthermore, major business decisions such as reduction of the shipping times can change the overall mean.

- *Marketing promotion information*: Knowledge about ongoing and future marketing activities often affect sales directly. Incorporating such information in the CLV prediction model seems promising.
- *Clickstream data with behavioral information* from the website and external sources: These types of fine-grained information—including customer feedback, discounts used, returned products—can be additional predictors in future models.

Moreover, it appears promising to incorporate features from other works, especially from [19], which derives complex features based on advanced mining processes. In particular, in [19] the authors investigate the relationship between systematic behavior of customers and their profitability and predictability. By using frequent itemset mining and clustering in a new framework, information entropy based measures are defined both on baskets and spatio-temporal dimensions. For example, systematic customers with repeating behavior can be detected this way. These calculated measures for customers could then be incorporated as additional features. Such features can be expected to be valuable, since systematic behavior can be highly predictive for the future profit of customers. We therefore consider this as promising future work.

6.3 Applicability of Our Method to Different Data Sources and Domains

The features used in our model are mostly based on customer and sales data, which usually can be easily obtained from a company's Customer Relationship Management (CRM) or Enterprise Resource Planning (ERP) system. Apart from CLV prediction, our framework could be used for similar problems, like forecasting the future number of purchases.

Our general methodology can also be based on other types of data in a more or less straightforward manner. For instance, if we use page views instead of purchases, similar features can be generated from clickstream data. Instead of measures such as profit and revenue, one can use the duration of sessions on the website or the number of unique page views instead of the number of unique items bought. Furthermore, the number of site revisits can be used instead of repurchases etc.

Our approach can also be used outside e-commerce, for example for online streaming services (e.g., for music or movies). While usage times can be a substitute for profit in this case, it is still meaningful to construct additional domain-specific features. Examples could be aggregate statistics of the browsing histories and rating behavior, in case such a functionality is available on the website. These statistics can be added in the feature engineering component of our method.

Another example from a different domain than traditional e-commerce would be the scenario of a telecommunication provider. In this case, products can be contracts with flexible options, e.g., purchasing of additional traffic volumes. Some item specific features are still relevant in this case, like the number of customers who repurchased a given product. Other features may include statistics based on tariff switching, the usage of optional services like extra traffic volume or phone minutes, as well as geolocation information. The difference in such a scenario is that in case of contracts, a part of the profit is already known in the future because of fixed contract times, which reduces the variability of the CLV compared to other scenarios. Also unlike in e-commerce, it is also explicit in the data when a customer ended his or her relationship with the company. This would therefore represent an important feature to include, i.e., whether customers terminated their contracts or sent their notice of cancellation for the future.

6.4 Future Methodological Extensions

One part of our future work comprises the refinement and extension of our recurrent neural network architecture. This includes deeper architectures with various ways of deepness and information exchange between layers. One further promising approach lies in the incorporation of *attention*

mechanisms [39]. The principle of attention is based on summarizing past information in "context vectors", which are read and updated using a smoothed attention distribution (cf. Neural Turing Machines [18]). Moreover, an augmented RNN can be used for a given RNN that helps to decide which part of the past we should focus on (Attentional Interfaces [2]). For instance, using attention one can directly use information from previous GRU layers instead of passing their compressed weights as initial state to the next layer.

Attention models have been successfully applied in different domains, including in particular NLP applications. However, the value of using such attention layers in our application domain still has to be explored. In our case, we use temporal convolutions and a rich set of features and this approach might already include part of the information that attention might capture. In general, there are also only few works that deal with time series and the benefit of using attention layers in our problem setting is therefore not yet fully clear.

A second direction for future research lies in the use of *neural architecture search* [41], which we consider a promising approach for developing refinements of our proposed RNN model without having to design and tune such a network manually. For this purpose however—in addition to substantial computing resources—more data is needed, which is currently not available for the CLV domain.

REFERENCES

[1] Md Al-Amin, Md Saiful Islam, and Shapan Das Uzzal. 2017. Sentiment analysis of Bengali comments with Word2Vec and sentiment information of words. In *2017 International Conference on Electrical, Computer and Communication Engineering (ECCE)*. IEEE, 186–190.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Jan Beyersmann, Arthur Allignol, and Martin Schumacher. 2011. *Competing risks and multistate models with R*. Springer Science & Business Media.

[4] Derya Birant. 2011. Data mining using RFM analysis. In *Knowledge-oriented applications in data mining*. InTech.

[5] Robert C. Blattberg, Byung-Do Kim, and Scott A. Neslin. 2008. *Database Marketing – Analyzing and Managing Customers*. Springer.

[6] Sharad Borle, Siddharth S Singh, and Dipak C Jain. 2008. Customer lifetime value measurement. *Management Science* 54, 1 (2008), 100–112.

[7] Léon Bottou. 2012. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*. Springer, 421–436.

[8] Peter J. Brockwell and Richard A. Davis. 2016. *Introduction to Time Series and Forecasting* (3 ed.). Springer.

[9] Jan Roelf Bult and Tom Wansbeek. 1995. Optimal selection for direct mail. *Marketing Science* 14, 4 (1995), 378–394.

[10] Benjamin Paul Chamberlain, Angelo Cardoso, Chak H Liu, Roberto Pagliari, and Marc Peter Deisenroth. 2017. Customer lifetime value prediction using embeddings. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1753–1762.

[11] Daqing Chen, Sai Laing Sain, and Kun Guo. 2012. Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining. *Journal of Database Marketing & Customer Strategy Management* 19, 3 (2012), 197–208.

[12] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794.

[13] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[14] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[15] Peter S Fader, Bruce GS Hardie, and Ka Lok Lee. 2005. "Counting your customers" the easy way: An alternative to the Pareto/NBD model. *Marketing Science* 24, 2 (2005), 275–284.

[16] Peter S Fader, Bruce GS Hardie, and Ka Lok Lee. 2005. RFM and CLV: Using iso-value curves for customer base analysis. *Journal of Marketing Research* 42, 4 (2005), 415–430.

[17] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT Press Cambridge.

[18] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* (2014).
[19] Riccardo Guidotti, Michele Coscia, Dino Pedreschi, and Diego Pennacchioli. 2015. Behavioral entropy and profitability in retail. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 1–10.
[20] R. Guidotti, G. Rossetti, L. Pappalardo, F. Giannotti, and D. Pedreschi. 2019. Personalized Market Basket Prediction with Temporal Annotated Recurring Sequences. *IEEE Transactions on Knowledge and Data Engineering* 31, 11 (2019), 2151–2163.
[21] Sunil Gupta, Dominique Hanssens, Bruce Hardie, Wiliam Kahn, V Kumar, Nathaniel Lin, Nalini Ravishanker, and S Sriram. 2006. Modeling customer lifetime value. *Journal of Service Research* 9, 2 (2006), 139–155.
[22] Sunil Gupta and Valarie Zeithaml. 2006. Customer metrics and their impact on financial performance. *Marketing science* 25, 6 (2006), 718–739.
[23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[24] Chun-Yao Huang. 2012. To model, or not to model: Forecasting for customer prioritization. *International Journal of Forecasting* 28, 2 (2012), 497–506.
[25] Dipak Jain and Siddhartha S Singh. 2002. Customer lifetime value research in marketing: A review and future directions. *Journal of Interactive Marketing* 16, 2 (2002), 34–46.
[26] Mahboubeh Khajvand, Kiyana Zolfaghar, Sarah Ashoori, and Somayeh Alizadeh. 2011. Estimating customer lifetime value based on RFM analysis of customer purchase behavior: Case study. *Procedia Computer Science* 3 (2011), 57–63.
[27] Gary Koop and Dimitris Korobilis. 2010. Bayesian multivariate time series methods for empirical macroeconomics. *Foundations and Trends in Econometrics* 3, 4 (2010), 267–358.
[28] Vineet Kumar and Werner Reinartz. 2018. *Customer relationship management: Concept, strategy, and tools*. Springer.
[29] Richard Lewis and Gregory C Reinsel. 1985. Prediction of multivariate time series by autoregressive model fitting. *Journal of Multivariate Analysis* 16, 3 (1985), 393 – 411.
[30] Xiao-Hui Li, Caleb Chen Cao, Yuhan Shi, Wei Bai, Han Gao, Luyu Qiu, Cong Wang, Yuanyuan Gao, Shenjia Zhang, Xun Xue, et al. 2020. A Survey of Data-driven and Knowledge-aware eXplainable AI. *IEEE Transactions on Knowledge and Data Engineering* (2020).
[31] Kenneth S. Lorek and G. Lee Willinger. 1996. A Multivariate Time-Series Prediction Model for Cash-Flow Data. *The Accounting Review* 71, 1 (1996), 81–102.
[32] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*. 4765–4774.
[33] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2018. Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PloS one* 13, 3 (2018).
[34] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. 3111–3119.
[35] Phillip E Pfeifer and Robert L Carraway. 2000. Modeling customer relationships as Markov chains. *Journal of Interactive Marketing* 14, 2 (2000), 43.
[36] David C Schmittlein, Donald G Morrison, and Richard Colombo. 1987. Counting your customers: Who-are they and what will they do next? *Management Science* 33, 1 (1987), 1–24.
[37] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.
[38] Ali Vanderveld, Addhyan Pandey, Angela Han, and Rajesh Parekh. 2016. An engagement-based customer lifetime value system for e-commerce. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 293–302.
[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.
[40] Zhi-Hua Zhou and Ji Feng. 2017. Deep forest. *arXiv preprint arXiv:1702.08835* (2017).
[41] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).

A TECHNICAL APPENDIX

In this appendix, we provide some mathematical details about our encoder-decoder sequence-to-sequence RNN model presented in Section 4.4 and how its individual components interact. For the simplification of the presentation, we assume that all the feature engineering, the causal temporal convolutions and the pretraining of the embeddings has been performed already, and all features are summarized in a tensor X. Moreover, we omit the details of dropout, but present the model in case without dropout and recurrent dropout.

Let us denote the sequence of profit values by $(y_t)_{t\in\mathbb{N}_0}$, which is the sequence of the target we want to predict for a future time period. If we want to estimate the customer lifetime value (CLV) for the period between $T+1$ and $T+L$, its value is given by

$$CLV_L = \sum_{t=T+1}^{T+L} y_t. \tag{5}$$

Formally, the encoder-decoder sequence-to-sequence model aims to learn the distribution of the sequence of future profit values. That is,

$$\mathbb{P}\left((y_t)_{t\in\{T+1,\ldots,T+L\}} \mid X_{:,1:T,:}\right), \tag{6}$$

where $X_{:,1:T,:}$ denotes the abstract input tensor for all customer data captured by all our F features up to time T, describing the training data distribution. I.e., X consists of the dimensions (data realizations, times, features).

In the following, we will use superscripts to indicate the depth of objects in the layer hierarchy shown on the bottom of Figure 8. Like before, let (x_t, h_{t-1}) denote the concatenation of the current time series input and the previous hidden state vector. According to the GRU equations (see Section 4.4 and the references therein), these are passed to neural networks (NNs) with weight matrices W and biases b with corresponding subscripts, leading to the following equations[14] describing the first encoder GRU:

$$z_t^{(1)} = \sigma\left(W_z^{(1)} \cdot \left(x_t, h_{t-1}^{(1)}\right) + b_z^{(1)}\right), \tag{7}$$

$$r_t^{(1)} = \sigma\left(W_r^{(1)} \cdot \left(x_t, h_{t-1}^{(1)}\right) + b_r^{(1)}\right), \tag{8}$$

$$\widetilde{h}_t^{(1)} = \tanh\left(W_h^{(1)} \cdot \left(x_t, r_t^{(1)} \odot h_{t-1}^{(1)}\right) + b_h^{(1)}\right), \tag{9}$$

$$h_t^{(1)} = \left(1 - z_t^{(1)}\right) \odot h_{t-1}^{(1)} + z_t^{(1)} \odot \widetilde{h}_t^{(1)}. \tag{10}$$

After the computation for all input t (corresponding to the chosen input sequence length, which is a hyperparameter), the final hidden state of the first encoder GRU is then passed to a dense neural network:

$$h^{(2)} = \tanh\left(W_d^{(2)} \cdot h_T^{(1)} + b_d^{(2)}\right). \tag{11}$$

Note that the dimension is given by $W_d^{(2)} \in \mathbb{R}^{d^{(3)}\times d^{(1)}}$, where $d^{(1)}$ is the number of hidden units of the first encoder GRU layer and $d^{(3)}$ is the number of hidden units of the second encoder GRU layer.

The output of this network is in turn used to compute the initial state of the second encoder GRU:

$$h_0^{(3)} := h^{(2)}. \tag{12}$$

[14] As before, $\odot$ denotes the Hadamard (element-wise) product and all functions (sigmoid and tanh) are also applied element-wise.

Afterwards, the second encoder GRU is updated in the same way:

$$z_t^{(3)} = \sigma\left(W_z^{(3)} \cdot \left(x_t, h_{t-1}^{(3)}\right) + b_z^{(3)}\right), \tag{13}$$

$$r_t^{(3)} = \sigma\left(W_r^{(3)} \cdot \left(x_t, h_{t-1}^{(3)}\right) + b_r^{(3)}\right), \tag{14}$$

$$\widetilde{h}_t^{(3)} = \tanh\left(W_h^{(3)} \cdot \left(x_t, r_t^{(3)} \odot h_{t-1}^{(3)}\right) + b_h^{(3)}\right), \tag{15}$$

$$h_t^{(3)} = \left(1 - z_t^{(3)}\right) \odot h_{t-1}^{(3)} + z_t^{(3)} \odot \widetilde{h}_t^{(3)}. \tag{16}$$

A second dense layer is then applied to the final hidden state of the second encoder GRU layer:

$$h^{(4)} = \tanh\left(W_d^{(4)} \cdot h_T^{(3)} + b_d^{(4)}\right), \tag{17}$$

where $W_d^{(2)} \in \mathbb{R}^{d^{(5)} \times d^{(3)}}$, with $d^{(5)}$ being the hidden dimension of the second encoder GRU layer.

The output of this network is in turn used to compute the initial state of the decoder GRU:

$$h_0^{(5)} := h^{(4)}. \tag{18}$$

While the second encoder GRU can be seen as a refinement for the first encoder GRU, aiming to correct its mistakes, the purpose of the decoder GRU is to match the output size, which is accomplished by slicing a subset of times. So while the fundamental GRU update equations remain the same, the time subsets as well as the output processing differ. In particular, a time distributed dense layer operation is used.

$$z_t^{(5)} = \sigma\left(W_z^{(5)} \cdot \left(x_t, h_{t-1}^{(5)}\right) + b_z^{(5)}\right), \tag{19}$$

$$r_t^{(5)} = \sigma\left(W_r^{(5)} \cdot \left(x_t, h_{t-1}^{(5)}\right) + b_r^{(5)}\right), \tag{20}$$

$$\widetilde{h}_t^{(5)} = \tanh\left(W_h^{(5)} \cdot \left(x_t, r_t^{(5)} \odot h_{t-1}^{(5)}\right) + b_h^{(5)}\right), \tag{21}$$

$$h_t^{(5)} = \left(1 - z_t^{(5)}\right) \odot h_{t-1}^{(5)} + z_t^{(5)} \odot \widetilde{h}_t^{(5)}, \tag{22}$$

And for all $t = T + 1, \ldots, T + L$ compute:

$$\widehat{y}_t^{(6)} = W_{d,t}^{(6)} \cdot h_t^{(5)} + b_{d,t}^{(6)}, \tag{23}$$

where $W_{d,t}^{(6)}$ is a (usually small) weight matrix for a dense layer with linear activation, dependent on t. This is the case because the dense transformation is applied to every time step t separately, which is useful to learn specific patterns for individual time steps.

Every output $W_{d,t}^{(6)}$ is then compared to the actual value y_t by means of a loss function L, i.e., the average of all $L\left(\widehat{y}_t^{(6)}, y_t\right)$ is considered.

As described in the main text in Section 3.2, a sliding window approach is used for training to shift the value of T backwards, where the maximum time T is chosen such that $T + L$ is still available in the training data (to know actual outcomes to learn from).

In this way, the abstract time series input vectors x_t are fed with actual data. That is, imagine that we construct the entire tensor of possible inputs $X \in \mathbb{R}^{N \times T_{input} \times F}$, where N is the number of data points, T_{input} the temporal input sequence length and F the number of features, we iteratively

set (with an example of some of our features)

$$x_t := X_{n,t,:} \tag{24}$$

$$= \big(\text{PastProfit}_{n,t}, \text{NumberOfOrders}_{n,t}, \text{TimeSinceLastOrder}_{n,t}, \tag{25}$$

$$\text{TimeSinceFirstOrder}_{n,t}, \text{NumberOfItems}_{n,t}, \text{ChildAge}_{n,t}, \text{MaxReorderRate}_{n,t}, \ldots\big) \tag{26}$$

for some training data row $n \in \{1, ..., N\}$ and given time step t, where : denotes the tensor slice consisting of all features. Each n corresponds to a particular customer and time in the past, as constructed by the sliding window approach. And t reflects the time step relative to such an input sequence. Note however, that the full tensor X does not need to be constructed explicitly, it is sufficient to only generate batches of data for training.

The training is then performed with the common backpropagation through time algorithm. I.e., automatic differentiation is applied to the temporally unfolded neural network to obtain all gradients. These can then be used for learning by an optimization variant of stochastic gradient descent (e.g., the Adam optimizer).

Finally, after the training has been performed in the described way, the predicted CLV for the period from $T + 1$ to $T + L$ is then calculated by

$$\widehat{CLV}_L = \sum_{t=T+1}^{T+L} \widehat{y}_t^{(5)}, \tag{27}$$

which is in turn compared with the actual value CLV_L defined above, by taking the average over some loss, like MSE or MAE.