Representation Learning Methods for Sequential Information in Marketing and Customer Level Transactions

Juan Pablo Equihua Linares

A thesis submitted for the degree of
Doctor of Philosophy

Department of Mathematical Sciences

University of Essex

Date of submission for examination December, 2022

Abstract

The rapid growth of data generated by businesses has surpassed human capabilities to produce actionable insights. Modern marketing applications depend on vast amounts of customer labelled data and supervised machine learning algorithms to predict customer behaviour and their potential next actions. However, this process requires significant effort in data pre-processing and the involvement of domain experts, which can be costly and time-consuming. This work reviews representation learning techniques as an alternative approach to feature engineering, aiming to eliminate the need for hand-crafted features and accelerate the process of extracting insights from data. Techniques such as Bayesian neural networks, general embeddings, and encoding-decoding architectures are explored to compress information obtained directly from raw input data into a dense probabilistic space.

This thesis introduces the necessary technical aspects of neural networks and representation learning, from traditional methods like principal component analysis (PCA) and embeddings, to latent variable and generative methods that use deep neural networks, such as variational auto-encoders and Bayesian neural networks. It also explores the theoretical background of survival analysis and recommender systems, which serve as the foundation for the applications presented in this work to predict when individuals are likely to stop their relationship with businesses in a non-contractual settings or which items individuals are the most likely to interact with in their next purchase. Experiments conducted on real-world retail and benchmark datasets demonstrate comparable results in terms of predictive performance and superior computational efficiency when compared to existing methods.

Acknowledgements

The work presented in this thesis would not be possible without the efforts and motivation of many people over these years. I want to deeply thank the KTP team at the University of Essex, the KTP team at Profusion (Henrik Nordmark, Asaf Levy, and Anne Huber), and my KT advisor Mark Lynch for all the effort invested in my enrolling in the Ph.D. while working in Profusion and make it possible in the first place. I want to thank my supervisors Berthold Lausen and Maged Ali for all the invaluable feedback and all the hours of discussion on how to improve the work presented in the papers and in this thesis.

I am infinitely grateful to all the people that supported me emotionally through these years, my mom who has given me everything in this world. Narda Hernandez who has been there in the moments I need to talk and always willing to listen and make fun of my tragedies. Tom Weber who has been there to read my work even on weekends and provide invaluable feedback and has acted as a Ph.D. role model to me. And Anastasiia Bondarchuk who motivated me and kept me on track to write this work end-to-end over the last few months.

List of Submitted Manuscripts

Contributions included in this thesis

Juan Pablo Equihua, Henrik Nordmark, Maged Ali, and Berthold Lausen. *Modelling customer churn for the retail industry in a deep learning based sequential framework*. Journal of the Academy of Marketing Science. (Submitted for publication in April 2023)

Juan Pablo Equihua, Maged Ali, Henrik Nordmark, and Berthold Lausen. *Sequence-aware item recommendations for multiply repeated user-item interactions*. Journal of Retailing. (Submitted for publication in April 2023)

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

In the last years, the vast amount of data produced by businesses around the world often exceeds the ability of humans to produce reliable insights. Nowadays, the majority of modern applications in marketing to obtain insights about customers and potential buyers rely on the use of large amounts of customer labelled data and supervised machine learning or artificial intelligence algorithms to obtain automatic predictions about customer's behaviour and their likely next actions. For example, predicting which customers are likely to make their next purchase over the next few days, or which ones are likely to be interested in a particular product or offering. At having such information about customers, companies are well equipped to produce better targeted offerings to customers to enhance customer experience and satisfaction, therefore increasing overall revenue to generate more businesses. As noted by [Davenport et al., 2019], artificial intelligence is likely to impact significantly marketing strategies, sales processes and customer opportunities over the next years.

However, the use of labelled data and supervised learning typically induces the need of dedicating large efforts in data pre-processing to create high-quality attributes to predict the desired variable of interest, which is a process not only expensive for companies, but also requires the involvement of specialists in the field to ensure relevancy of the generated attributes from data. This process limits directly the predictive ability of machine learning models to make accurate predictions and make automated decisions about customer's journey

once they are deployed in production applications, such as recommending a specific product, offering a discount at check-out, or include the customer in a targeted marketing campaign.

To overcome the need of obtaining hand-crafted features from specific knowledge experts and speeding up the process of obtaining insights from data, representation learning techniques, such as bayesian neural networks, general embeddings, and encoding-decoding architectures aim to compress the information obtained directly from raw input data and its relationship with the target variable of interest into a dense probabilistic space. This space then can be used to obtain data representations and train machine learning models without the need of observing the original data or attributes created from it. Representation learning techniques can be useful in several business applications, particularly when obtaining high-quality attributes or acquiring additionally data for experimentation is expensive or not possible at all.

The main goal of this thesis is exploring an alternative approach to feature engineering approaches commonly used in machine learning to obtain hand-crafted attributes from data and train machine learning models with similar or better prediction performance. Instead, the process of encoding the necessary information to predict the target variable from raw data is carried by a representation learning technique, such a neural network.

The primary target audience for this thesis is the scientific community of machine learning and marketing who are interested in exploring efficient techniques to obtain representations of data and their potential applications. Although the experiments presented in this thesis are focused on marketing science and obtaining insights about customers, these can be easily implemented in other industries and applications with a different goal.

1.2 Research objectives

This thesis aims to explore representation learning techniques as an alternative approach to feature engineering in machine learning, with a special focus in marketing applications. The work and experiments presented in following chapters aim to eliminate the need for hand-crafted features in feature engineering and accelerate the process of extracting insights from data by compressing information obtained directly from raw input data into representations from which machine learning algorithms can learn. Specific objectives of this work include:

Explore the use of representation learning techniques as a substitute to hand-crafted

features and feature engineering.

The primary goal of this thesis is to investigate the potential of representation learning techniques as a substitute for traditional feature engineering in machine learning, with a particular emphasis on marketing applications. By compressing raw input data into a dense probabilistic space, the research seeks to eliminate the reliance on hand-crafted features and ease the process of deriving insights from data.

Investigate technical aspects of neural networks and representation learning.

This thesis examines the technical aspects of neural and representation learning. It covers well-stablished methods to obtain representations from data like principal component analysis (PCA) and embeddings, which are widely used in natural language processing (NLP) and computer vision to encode data without the need for creating probability distributions. Additionally, it explores latent variable and generative methods that use deep neural networks, such as variational auto-encoders and Bayesian neural networks, which combine ideas from deep learning and probabilistic modelling to capture relevant information from data with respect to the desired target variable.

Review theoretical background of survival analysis and recommender systems and understand how these are applied in marketing settings.

This work examines the theoretical background of survival analysis and recommender systems, which serve as the foundation for the applications, baseline models, and experiments presented in the following chapters. Survival analysis focuses on predicting the time until an event occurs, such as customer churn or next purchase, while recommender systems aim to suggest items or actions to users based on their preferences and behaviour.

Explore the application of deep learning-based representation learning methods for enhancing marketing strategies.

Introduce novel approaches to analyse item purchases and customer churn as sequential information seen as a series of time-ordered events rather than static snapshots. This approach acknowledges the dynamic nature of customer interactions and purchasing patterns, which evolve over time and are influenced by various factors such as marketing campaigns, seasonal trends, and personal preferences.

Discuss the potential benefits, challenges, and drawbacks of implementing representation learning methods in machine learning.

This work aims to discuss potential benefits, challenges, and drawbacks of implementing representation learning methods in machine learning. It highlights the importance of considering the specific characteristics of marketing data and the need for interpretable models in certain applications.

1.3 Contributions to knowledge

Despite its importance and relevance in several applications, the development of representation learning in the field of marketing has been slower than in fields such as image processing or reinforcement learning, where finding robust representations of data is at the core of most production-ready application developed in industry, mainly do the fact that creating hand-crafted features for every image or for every experiment to be conducted is highly inefficient or impossible. One of the main reasons for this is also the abundant availability of well-established methodologies in statistics to create human-interpretable predictions from low dimensional structured data usually found in marketing settings, such as logistic regression and decision trees. However, as nowadays most of customer's behaviour data is produced by automated systems in the form of logs, the available data is more often complex and extremely high-dimensional, which limits considerably the capacity of using the standard and interpretable machine learning techniques.

Broadly speaking, most representation learning techniques combine ideas from deep learning and probabilistic modelling to capture relevant information from data with respect to the desired target variable by defining generative models based on neural networks with the use of known multi-dimensional distributions, to estimate the inherent probability distribution from which available data was generated. For example, variational autoencoders (VAEs) [Kingma and Welling, 2013], are deep latent variable models are widely used in the field of computer vision to create low-dimensional representations of images by capturing the distribution of pixels in a set of images, or in unsupervised learning of to generate low-dimensional representations of high-dimensional data distributions. Nevertheless, there are representation learning techniques purely based in deep learning, such as embeddings that are widely used in natural language processing (NLP) and computer vision to encode data without the need of creating probability distributions.

This thesis contributes to the academic knowledge in the field of representation learn-

ing and artificial intelligence, particularly in the field of marketing, by proposing novel approaches to obtain insights from customer data and improve prediction performance by proposing novel methodologies for marketing applications, demonstrating their effectiveness through extensive experiments, and discussing their potential benefits, challenges, and future research directions. Overall, these contributions can be broadly categorized into three areas: Advancements Deep learning representation learning techniques for marketing applications, Novel applications in customer interactions, and Discussion on benefits, challenges, and future research directions.

Advancements Deep learning representation learning techniques for marketing applications.

This thesis provides a comprehensive introduction to deep learning and representation learning techniques, including recurrent neural networks, Bayesian neural networks, and autoencoders. It discusses the advantages of these techniques over traditional hand-crafted feature engineering methods, such as reliability, scalability, flexibility, and ease of implementation. This foundational knowledge is essential for understanding the novel applications presented in the later parts of the thesis. Furthermore, this work explores the use of representation learning techniques in marketing applications, where obtaining high-quality attributes or acquiring additional data for experimentation is expensive or not possible. By leveraging techniques such as Bayesian neural networks, general embeddings, and encoding-decoding architectures, this work demonstrates the potential of these methods to compress information obtained directly from raw input data and its relationship with the target variable of interest into a dense probabilistic space.

Novel applications in customer interactions.

This thesis introduces two innovative applications of representation learning techniques for customer interactions: predicting customer churn (Chapter 3) and item recommendation (Chapter 4). These applications demonstrate how customer transactions can be analysed as sequential information using Bayesian and recurrent neural networks, without the need for additional hand-crafted attributes. The successful implementation of these applications showcases the potential of representation learning techniques in marketing and other industries.

Discussion on benefits, challenges, and future research directions.

The work highlights the challenges and limitations associated with implementing representation learning techniques, such as interpretability, evaluation, and experimentation. By discussing these challenges, the work contributes to a better understanding of the current state of the field and helps researchers and practitioners to identify areas where further research is needed. Furthermore, this work outlines potential questions for further research, paving the way for future advancements in the field of representation learning and artificial intelligence applied to marketing and other industries. These contributions not only advance the state-of-the-art in representation learning techniques but also provide valuable insights for practitioners aiming to leverage these methods in real-world marketing scenarios.

1.4 Thesis outline

Chapter 2 introduces the necessary technical aspects of neural networks and representation learning, from traditional methods to perform dimensionality reduction of high-dimensional spaces like principal component analysis (PCA) and embeddings, to latent variable and generative methods which aim to discover and represent data as their generative probability distribution with the use of deep neural networks such as variational auto-encoders, and Bayesian neural networks. This chapter also explores the theoretical background of survival analysis and recommender systems, which are the foundations to introduce the applications, baselines models, and experiments presented in the following chapters.

Chapter 3 presents the methodology and the main results of the submitted manuscript *Modelling customer churn for the retail industry in a deep learning based sequential framework* that introduces a probabilistic deep learning approach to compress information of purchases to estimate the likelihood of next customer purchase over a period of time. Under the assumption that customer interactions can be represented as sequences of data through time, the main focus of this chapter is to find a representation space for marketing sequences by combining unsupervised learning, deep neural networks, and survival analysis to avoid time consuming hand-made feature engineering and estimate what is the most likely future time for customers to make their next purchase. This is achieved by creating a latent space from sequences of time arrivals and make predictions of the survival distribution of event-times at customer level.

Predictions made with this methodology are compared against two well-established

survival analysis methods commonly used to predict customer churn: Cox Proportional Hazard (CPH), and Kaplan-Meier (KM) estimator with respect to survival and classification metrics for experiments using a dataset containing transactions of retail customers, as well as a simulated dataset designed to resemble customer transactions. The results obtained show comparable prediction performance of predictions made with sequential representations of purchases against methods that use hand-crafted features at training, besides allowing to obtain individual estimations of churn probability for individual customers.

Chapter 4 presents the methodology and main results of the submitted manuscript *Sequence-aware item recommendations for multiply repeated user-item interactions*, an innovative approach to analyse item purchases as sequential information to make product recommendations. Inspired by natural language processing techniques to process, and analyse sequences of text, this paper proposes a recommender system that accounts to the order of user-item interactions in a sequential framework to make recommendations. This new technique analyses all purchases made by customers and processes them with a recurrent neural network to make predictions of which are the most likely items to be bought next.

Experiments were conducted for two retail datasets and a popular recommender systems benchmark dataset showing outstanding performance in terms of recommendations quality against recommendation techniques used in industry such as matrix factorisation and collaborative filtering, besides showing a considerable reduction in the computational time to make predictions and item recommendations to customers. This new technique was additionally tested against a proprietary recommender system in a live A/B test for 500,000 customers recommendations in an email marketing campaign obtaining an uplift of 13% in total revenue for the business and deployed in production for on-going usage.

Chapter 5 finally discusses the potential benefits, challenges, and drawbacks of implementing representation learning methods in machine learning, as well as highlighting potential questions for further research.

CHAPTER 2

Theoretical background

2.1 Artificial Neural Networks

Artificial neural networks (ANN) are popular machine learning models capable of learning complex non-linear patterns present in data with the use of nodes and weighted connections interrelated in an architecture design that is mainly inspired in the structure of the human brain. Neural networks have proof to be models capable of outperforming several machine learning techniques such as logistic regression and tree-based models in different domains, such as Natural Language Processing (NLP) [Camacho-Collados and Pilehvar, 2018], computer vision [Rawat and Wang, 2017], and machine translation [Sutskever et al., 2014]. ANN's have been also widely explored in the fields of survival analysis by [Kvamme et al., 2019] and churn prediction by [Sharma and Panigrahi, 2011], for tasks where the observations may allow to presence of censoring and covariates can be extracted from the input data.

2.1.1 History of Neural Networks

The concept of artificial neural networks is not new in the field of machine learning, the concept of an artificial neuron is introduced by McCulloch [McCulloch and Pitts, 1943] which is commonly treated as the origin of neural networks in machine learning. Nevertheless, it is until late 1950's when the first practical application is introduced with the use of the perceptron by Rosenblatt [Rosenblatt, 1958] with demonstrated the ability of artificial neurons to perform pattern recognition.

Unfortunately, the single-layer perceptron was widely criticised due to main two issues, the first, its inability to solve the exclusive-or (XOR) problem. And the second one, that advanced computers back then were incapable to perform the necessary calculations to train large neural networks [Minsky and Papert, 1969]. Leading into many researchers losing interest in further development in the field for over a decade. It was until 1980's when research in artificial neural networks increased dramatically and new concepts had been introduced. Hopfield [Hopfield, 1982] used statistical mechanics to explain the operation of neural networks with associative memory. Werbos [Werbos, 1974] introduced the back-propagation algorithm as a generalisation of the delta/chain rule to train multi-layer perceptron. And Rumelhart et al. [Rumelhart and McClelland, 1987], showed the efficiency of the backpropagation method for the family of semi-linear activation functions, which required activations to be non-decreasing and differentiable to overcome all the issues mentioned by Minsky and Papert in 1969.

In the 1990's and 2000's researchers tried to develop neural networks by using backpropagation and stochastic gradient descent during training, unfortunately most of the networks would not train as quick as to be useful for real applications. It was just until Hinton et al. [Hinton et al., 2006] when learning of complex architectures such as the convolution neural network was efficient enough to be used in modern applications.

By 2010's, Deep Learning (DL) with artificial neural networks had already became in a popular methodology for training new machine learning models. Its popularity is often attributed to their ability to approximate any non-linear decision function for a wide range of applications in Computer Vision (CV) for image classification, Natural Language Processing (NLP) and Understanding (NLU) for text analysis, and Robotics for task automation. A considerable amount of deep learning research has focused on improving state-of-the-art benchmarks for several applications, typically by assigning either more computational resources or data to the task in hand, or by simple increasing the size of the network by attaching more layers to the architecture.

It was not until 2017, when Vaswani et al. [Vaswani et al., 2017] introduced the Transformer, a model architecture fully based on the attention mechanism to draw global dependencies between input and output signals. Nowadays, most popular applications in the Artificial intelligence (AI) industry rely on the use of transformer-based neural networks to make predictions. For instance, the Generative Pre-trained Transformer 3 (GTP-3) is a

specialised transformer trained from dozens of NLP datasets to find data representations in an unsupervised fashion and then fine-tuned for specific natural language generations and understanding tasks, the largest version of this model contains 175 billion parameters and is capable of achieving state-of the-art results in relatively all NLP applications.

2.1.2 Architecture

The typical neural network consists of the concatenation of simple processing units interconnected between them to process data or information over a large number of weighted connections. In the neural architecture, each unit performs a relatively simple job, the unit receive input from the input or the previous layer and use it to compute the output signal as the sum of weighted inputs plus a bias terms followed by an activation function.

Mathematically, a neural network ($NN : \mathbb{R}^n \rightarrow \mathbb{R}^m$) with n-dimensional input and m-dimensional output can be seen as a linear combination or a function of $X \in \mathbb{R}^n$ features (nodes) with their corresponding weights $W \in \mathbb{R}^{n \times m}$ and bias term $b \in \mathbb{R}$ (connections), followed by a non-linear activation function, i.e., $NN(X) = \varphi(b + W^T X)$, where $\varphi : \mathbb{R}^m \rightarrow \mathbb{R}^m$ represents the chosen activation for the network. Figure 2.1 shows a visual representation of this process for a single unit with n inputs. In practice, deep neural network architectures contain multiple nodes in their input and intermediate layers to allow the learning of complex non-linear mappings from input to output data during the training process. To simplify the notation, we denote NN_X, as the neural network learnt from training data X.

Common choices of activation are the linear ($\varphi(x) = x$), exponential ($\varphi(x) = exp(x)$), softmax ($\varphi(x) = exp(x) / \sum_j exp(x_j)$) for a vector of dimension j, and sigmoid ($\varphi(x) = \frac{1}{1+e^{-x}}$) functions. Figure 2.1 shows a visual representation of this process for a unit with n inputs.

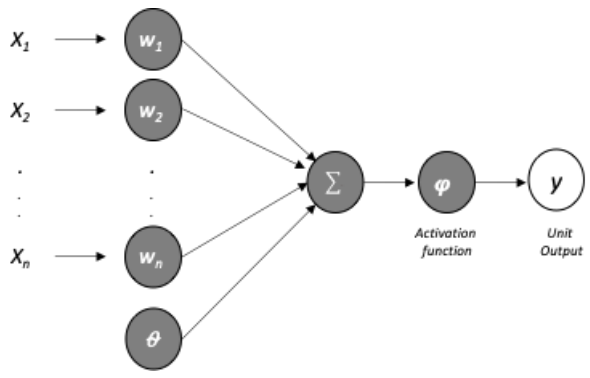

Figure 2.1: Visual representation single unit neural network.

Although neural networks are powerful machine learning models, these also have their caveats. Firstly, due to their structure with usually multiple intermediate layers, the model might contain thousands or millions of parameters that need to be optimized in the training process, so these techniques require large volumes of input data at training to avoid overfitting and provide reliable predictions. Secondly, neural networks typically provide only point estimates of the target variable and do not capture the uncertainty in their predictions, which is not favourable when the goal is estimating a probability distribution such as in the experiments presented in Chapter 3.

A common way to approach these issues is using representation learning methods by taking a probabilistic approach and assume a distribution over the target variable Y, i.e., $Y \sim Q(\theta)$ with unknown parameter vector θ, then the neural network outputs are only estimates of the parameter distribution instead of estimates of the target variable, i.e, $NN_X(x) = \varphi(b+W^T x) = \hat{\theta}$, and target point estimates can be obtain as the expectation of the distribution w.r.t. the input data as $\hat{y} = E[Q(\hat{\theta} \mid x)]$. Naturally, this process can be implemented not only for the output layer, but in each or some of the intermediate layers as well, leading into the field of Bayesian neural networks that are trained by minimizing the Kullback-Leibler divergence between the estimated posterior distribution $Q(\hat{\theta})$ and a defined prior distribution as noted by [Lampinen and Vehtari, 2001].

2.1.3 Training Process

A neural network must be configured in such way that the application of inputs and weights throughout the architecture resembles the desired set of outputs. Typically, the initial weights in the network are set as random using a uniform distribution between 0 and 1 and then the architecture can be 'trained' by processing input teaching data and updating the network weights with respect to a loss or error function. This process is commonly known in machine learning as 'supervised learning' of the network, where the pair of input-output tuples are fed in the network in batches.

Werbos [Werbos, 1974] proposed in 1974 the backpropagation algorithm as a generalisation of the delta rule for non-linear activations and multi-layer networks, with the only restrictions that the activation need to be non-decreasing and differentiable. The main goal of the learning process via backpropagation is the activation values are propagated to the previous layers by comparing the networks' outputs with the desired true output values, and

distributing the error of the model to all the hidden layers that each unit is connected to by applying the chain rule to the derivative of the loss function with respect to all the weights of the model.

2.1.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are an extension of traditional neural networks that have proof efficiency in modelling sequential data in several applications domain over the last decade. For example, in the field of machine translation [Sutskever et al., 2014] to translate textual documents from one language into another, and computer vision [Phong and Ribeiro, 2020] to perform image and video classification and object recognition. In RNNs, the hidden cell structure can be used to encode information of a temporal-dependant random variable at time t denoted as $X_{<t} = [x_1, x_2, ..., x_{t-1}]$ into a latent variable $h_t = g(X_{<t})$, which depends of the time t and then is used to define the output distribution $p(X_t|h_t)$. For an observation of $X_{<t}$, the state of the latent variable h_t evolves over time, and at each time step incorporates information from the previous state by using a non-linear function g, i.e., at each time-step $h_t = g(h_{t-1}, X_t)$ as shown in Figure 2.2.

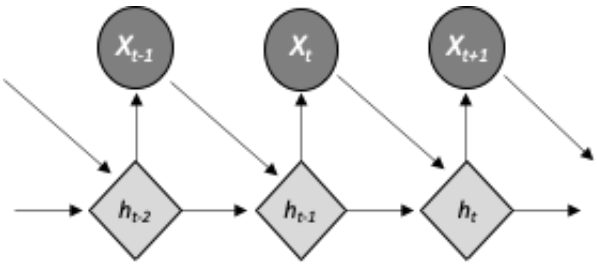

Figure 2.2: Graphical representation of the hidden state structure of Recurrent Neural Networks.

However, the differentiable function g must to be powerful enough to capture long-term dependencies in the data. Popular choices for g are memory cells units such as LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Chung et al., 2014], that use a gated mechanism to store and forget information through the recurrent training process and are specially designed to avoid the vanishing gradient problem in RNNs [Hochreiter, 1998].

Long-short Term Memory (LSTM)

One of the initial approaches to model sequential data with the use of neural networks was the long-short term memory (LSTM) unit proposed by Hochreiter and Schmidhuber in 1997

[Hochreiter and Schmidhuber, 1997], it shares many of the properties of the Gated recurrent unit [Chung et al., 2014] proposed later in 2014, although the LSTM is slightly more complex in its inner structure.

The LSTM unit incorporates Four gates in its structure, and it is designed to decide when to remember or ignore inputs into the hidden latent variable h_t. The first gate is the *output* gate which reads entries from the cell itself. The second gate is the *input* gate, which decides whether to read the data to be processed by the unit. The third gate is the *forget* gate which decides when to reset the contents of the cell. And finally, the *memory* cell which stores the information of the hidden state h_t at each time-step. In the LSTM unit, the input, output, and forget gates include a sigmoid activation to compute the values for each gate, which ranges between the interval [0,1], but the memory cell includes a tanh as the activation function with a value range for [-1, 1]. In each time-step, the data fed into the LSTM gates is the input at the current time-step x_t and the hidden state at the previous time-step h_{t-1}. Figure 2.3 shows a representation of the LSTM unit with each of its internal gates.

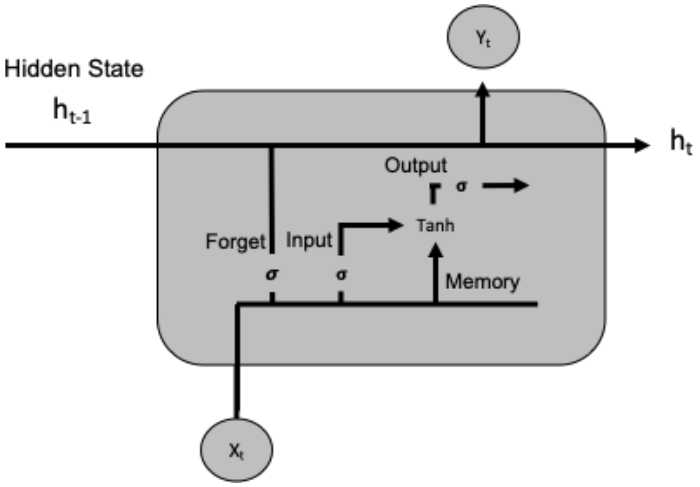

Figure 2.3: Computation of memory cells in an LSTM.

In practice, the training process of the LSTM can be done via the backpropagation algorithm presented in section 2.1.3, while the gated mechanism in the LSTM aids to mitigate the vanishing gradients problem [Kolen and Kremer, 2001] of long-term dependencies in neural network.

2.1.5 Asymmetric loss functions

Typically, the training process of neural networks is based in adjusting the networks' weights and biases via the back-propagation to find the weights that minimise the error between model's predictions $\hat{y}$ and real observed values y with respect to a loss function $L(\hat{y}, y) : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$, which measures the discrepancy between predictions and target values. In classification or regression tasks the target variable is fully known and given by a set of true labels y, thus the model predicts $\hat{y} = NN_X(x)$ and the model's error can be obtained straightforwardly. Popular choices of loss function are the Mean Squared Error ($MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$), and Mean Absolute error ($MAE = \frac{1}{N}\sum_{i=1}^{N} \mid y_i - \hat{y}_i \mid$) for regression, and the binary or categorical cross entropy ($Entropy = -\sum_{i=1}^{N} y_i \cdot log(\hat{y}_i)$) for classification. However, these loss functions do not consider the presence of censored labels in the data and only provide reliable estimations when over-predicting or under-predicting the real value of the target does not have a significant impact during model training, which is not the case at estimating the parameters of a time-to-event distribution.

The estimation of parameters for an exponentially shaped time-to-event distributions has been widely explored over the last decades, as several life problems such as waiting time problems or time intervals between events usually distribute similar to an exponential shape. Several authors, such as [Zellner, 1986, Varian, 1975, Srivastava and Tanna, 2007] have shown that the use of asymmetric loss functions outperform the estimation of parameters carried with quadratic-type losses such as MSE and MAE, due to their inherent structure that consider both, the goodness of fit against the distribution to estimate and the precision of individual estimations. In addition, generalized linear models [Nelder and Wedderburn, 1972] provide a statistical framework to model non-symmetric distributions.

Popular choices heuristically motivated for asymmetric loss functions are the LINEX loss function [Varian, 1975], which increases exponentially on one side of zero and linearly in the other side. Similarly, balanced loss functions (BLF) [Zellner, 1986] combine the distance of a given estimator to the target distribution and to its unknown parameters. Balanced loss functions are usually considered in the field of Bayesian statistics as these may consider prior knowledge of parameters that can be captured in the form of a prior distribution.

2.2 Representation learning

The success of most machine learning methods is heavily dependent of the data representation (or features) used in their construction [Bengio et al., 2013]. For this reason, most machine learning and artificial intelligence practitioners in research and industry allocate large amounts of resources in data preparation, cleaning, and feature engineering with the final goal of improving predictions. Representation Learning aims to overcome the need of obtaining expensive handcrafted features from data by leveraging robust mechanisms such as neural networks the process of building a representation of data themselves. In this domain different unsupervised techniques like Principal Component Analysis (PCA) [Pearson, 1901], Embeddings [Chamberlain et al., 2017], and Gaussian Mixtures Models (GMM) have been used to learn a latent representation of data for non-temporal domains.

In general, the goal of these techniques is to obtain a reliable representation of data without any previous knowledge of the full probabilistic generative process from which the data was extracted. For example, in computer vision, convolutional neural networks (CNN) are extremely efficient in obtaining a low-dimensional feature representation of images, which then can be used to into classifiers to make predictions [Rawat and Wang, 2017]. In the field of probabilistic deep learning, the Variational Autoencoder (VAE) [Kingma and Welling, 2013] has shown to be a powerful technique in learning the posterior distribution of data $p(X|Z)$, given a prior distribution $p(Z)$ where Z is a unknown latent variable and its distribution is estimated from the data with neural networks. An introduction to Variational Autoencoders and their applications can be found in [Kingma and Welling, 2013].

In this work, in order to to obtain reliable a representation of customer interactions and capture the dynamic preferences of customers over time for chapters 3 and 4, we need to analyse customer interactions as complete sequences instead of as static snapshots. Learning sequential models is a long-standing challenge in machine learning and statistics, commonly approached with the use of Dynamic Bayesian Networks (DBNs) such as Hidden Markov Models (HMM's) and Kalman filters, where the information of the temporal-dependant random variable X_t is usually modelled as the likelihood function in an auto-regressive manner: $p(X_t) = \prod_{t=1}^{t} p(x_t|X_{<t})$, i.e. as the product of the individual densities at each time step. However, over the last decade, Recurrent Neural Networks (RNN) have gained special attention from the research community as being a special type of neural networks which

recursively models data by updating and maintaining internal hidden states as stated in section 2.1.4.

2.2.1 Latent Modelling

Historically, statistical modelling has been focused on modelling the probability distribution of a random variable $X \sim p(X)$ as a parametric distribution $p(X, \theta)$ given a data sample of n observations of the original random variable $\{x_1, x_2, ..., x_n\}$, where the parameter vector θ of the model is unknown and the observations in the data sample are commonly assumed to be identically and independently distributed. Then, the goal of is to find a value $\hat{\theta}$ of the real parameter θ so that the parameterised distribution $p(X, \theta)$ matches closely the distribution of the data $p(X)$.

Machine learning methods which focus in using observed data to make predictions typically rely on the use of hand-crafted attributes extracted from the data sample, where the distribution of these attributes is typically known and joint likelihood can be obtained as the product of individual densities $p(X) = \prod_i p(x_i)$, then the goal of these systems is to learn a mapping function $f : X \rightarrow Y$ to make predictions using only the set of independent variables X and the target or dependant variable Y. This approach is considerably efficient in different applications, particularly when the size of the data sample is large and the dimensionality of X is relatively small. Unfortunately, as technology evolves and more data and more abundant types of data such as images, videos, and audio are available for analysis, the process of extracting hand-crafted attributes is not feasible in modern applications, as the dimensionality of every observation could be in the thousands or millions. For example, a modern smartphone camera can take a picture of 12 megapixels resolution (4000 x 3000 individual pixels), which means that a model that a machine learning model designed to classify images with this resolution will require to consider 36,000,000 parameters for every observation or picture in the training data.

Instead, latent modelling techniques try to resemble the generative process from which the data was created. These rely on the use of an auxiliary and unobserved random variable Z which captures all the necessary information of X. As we cannot directly observe the distribution $p(Z)$ of Z due to its large dimensionality, their properties must be inferred indirectly from the available data sample obtained from X. In practical scenarios, it is necessary to induce a prior distribution over the latent variable Z and obtain the joint

distribution over observed and latent variables as

$$p(X, Z) = p(X|Z)p(Z),$$

which allows to express the complex marginal distribution $p(X)$ in terms of a more simple distribution built from the conditional distribution $p(X|Z)$ and the prior distribution of the latent variable $p(Z)$. As the goal at training is achieving the learning of the distribution of Z it is necessary to obtain the distribution of latent variables $p(Z)$ with respect to the input data by using the Bayes' theorem

$$P(Z|X) = \frac{p(X|Z)p(Z)}{p(X)}.$$

Latent variables have been refereed in different domain of statistics with different names such as '*common factors*', '*latent factors*', '*underlying variables*', among others. These models can be used in multiple applications, such as creating *synthetic data*, which resemble the original distribution of X by simply sampling a new set new observations X' from the conditional distribution $p(X|Z')$, where the set of initial data points Z' can be obtained from sampling the known prior distribution of Z [Dilokthanakul et al., 2016], or in *dimensionality reduction* by choosing a dimensionality of Z much lower than the initial dimensionality of X.

Examples of widely used techniques which resemble the goal of latent variables for dimensionality reduction are Singular Value Decomposition (SVD), Principal Component Analysis (PCA) [Pearson, 1901] and Gaussian Mixture Models (GMM). For instance, PCA aims to obtain a set of uncorrelated and orthogonal eigenvectors which capture certain amount of variability from the original data, these eigenvectors can be seen as latent variables as their distribution is unknown before the analysis is done and during training, although these are not obtained with the use of the Bayes' theorem as stated above, the eigenvectors capture the main characteristics of the original data sample and its correlation with the target variable. However, in practical applications for images and audio processing, usually the non-correlated and linearity assumptions limit its efficiency at working with complex data distributions. Other methods, like Independent Component Analysis (ICA) [Hyvärinen et al., 2009] or sparse coding [Lee et al., 2006] assume independence of the obtained components, but in practice they usually fail in obtaining reliable representations of data. To relax the independency constrain in these methods, semi-parametric latent models

like [Teh et al., 2005, Murray and Adams, 2010] implement Gaussian mixtures to allow the existence of correlated latent factors and improve the representation learning in various tasks.

2.2.2 Embeddings

As a method to find efficient representation from textual data, word embeddings were proposed by Mikolov et al. [Mikolov et al., 2013a, Mikolov et al., 2013b] as an efficient alternative to techniques previously used extensively in literature like one-hot encodings, PCA, Latent Semantic Analysis (LSA), latent Dirichlet Allocation (LSA) and Gaussian mixtures. Unlike previously proposed methods, the goal of embeddings is finding representations of words that are somehow useful to predict the surrounding based on its context. Then, word representations are mapped into dense vectors over a representation space in which similar words vector are relatively close one to each other with respect to their context. For a given sequence of training words $[w_1, w_2, ..., w_T]$ of size T, this is achieved by maximising the average log probability as part of a supervised learning framework

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} log\ p(w_{t+j}|w_t)$$

where c is the size of the training context for the embedding, which is typically a function of the center word w_j and might affect the overall performance and computational time of the process. As a result, the learnt embeddings can preserve their semantic relations under simple linear operations [Mikolov et al., 2013b]. Embedding techniques require of large amounts of textual data during training (potentially billions of words), and large computational resources. Fortunately, as of today, several implementations are made available online for user consumption, such as Word2Vec [Mikolov et al., 2013a], and GloVe [Pennington et al., 2014].

Word Embeddings

One-hot Encoding

One-hot encoding is a popular technique in machine learning to convert categorical features into a numerical format in a way that can be easily processed by algorithms. In one-hot encoding, each category for a feature is represented as a binary vector with a length equal to the number of unique categories in the dataset. Only one element in the vector is set to 1, corresponding to the specific category, while all other elements are set to 0. For instance,

suppose we have a dataset containing information about different products available in stock, and the purchases of customers. To apply one-hot encoding to these products, we would first create a binary vector for each unique category, with length equal to the total number of products, then for each individual we would create a vector where each entry will correspond to the position of the specific product in the list of unique products, if the customer has interacted with that product in the past, and all other elements set to 0. If *'User A'* has only interacted with product 1, and *'User B'* has interacted with products 2 and 3, their corresponding one-hot vectors would look as follows:

$$\text{User A: } [1, 0, 0], \text{ User B: } [0, 1, 1]$$

One-hot encoding offers several advantages when used in recommender systems, including its simplicity, interpretability, and compatibility with popular techniques presented in section 2.4. This technique allows for an easy conversion of categorical variables into a numerical format without losing any information or introducing artificial relationships between categories and it is particularly useful when dealing with nominal data, where there is no inherent order or relationship between the categories. However, there are also some drawbacks, in particular the resulting high-dimensional representation and its sparse representation, which can lead into inefficient storage and computational processes.

Word2Vec Embeddings

Word2Vec embeddings were introduced by [Mikolov et al., 2013a] in 2013 as an approach to learn high-quality dense vector representations of words for dictionaries with potentially billions of tokens, while trying to keep similarity of words in terms of their semantics and position within sentences.

Particularly, the *Continuous Bag-of-Words* (CBOW) [Mikolov et al., 2013a] is an unsupervised neural network with a single fully connected hidden layer (a.k.a shared-projection matrix) for all words to represent each token in the network and predict the current word based on its context or surrounding words. Despite its simplicity, CBOWs are popularly used in textual applications to find scalable word representations as an alternative to the highly sparse bag-of-words representation. Although Word2Vec embeddings are considered unsupervised methods, these are learned as part of a supervised framework such as classification

or next token prediction, as in this work, by simply connecting the embedding layer within the overall architecture and training via backpropagation as shown in figure 4.1.

GloVe embeddings

Global Vectors (GloVe) [Pennington et al., 2014] is a popular method for learning word embeddings, It combines the both matrix factorization and local context window methods to capture global relationship of token within the corpus while maintaining the ability to learn from local contexts. The main goal of Glove is to establish a co-occurrence matrix that represents the frequency of words appearing together within a specified context window. Then, this matrix is decomposed into lower-dimensional matrices, which serve as the word embeddings, which are obtained by minimising the difference between the dot product of the resulting word embeddings and the logarithm of the co-occurrence probabilities.

GloVe embeddings offers several advantages to obtain word and token representations, such as scalability for large-scale corpus of text maintaining semantic relationships between words, dimensionality reduction, and enabling transfer learning between application domains. However, this method also has some caveats, including a fixed-size vocabulary that cannot directly represent out-of-vocabulary words, significant memory consumption for storing and processing large co-occurrence matrices, and the computational intensity and time-consuming nature of training GloVe embeddings from scratch, particularly for large datasets and high-dimensional embeddings.

ELMo embeddings

Embeddings for Language Models (ELMo) [Peters et al., 2018] is a method that learns context-aware word embeddings by training a deep bidirectional language model on a large corpus of text, and it has shown outstanding performance in sentiment analysis, named entity recognition, and question answering applications [Liu et al., 2020, Liu, Wenbin et al., 2020]. Its architecture consists of a character-based convolutional neural network or a token embedding, followed by two layers of bidirectional long short-term memory (BiLSTM) networks, where its training process incorporates both forward and backwards steps, the forward step predicts the upcoming word based on preceding words, while the backward step anticipates the prior word considering its subsequent words.

BERT embeddings

Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2018] is a pre-trained deep bidirectional transformer model that learns contextualized word embeddings by predicting masked words in a sentence. BERT embeddings extend the initial architecture proposed in ELMO embeddings by implementing the transformer architecture [Vaswani et al., 2017] which consists of a multi-layered stack of self-attention mechanisms and position-wise feed-forward networks, allowing for efficient parallel computing and better handling of long-range dependencies.

BERT embeddings are pre-trained on large corpus of text using the masked language modelling, where some words in the input sentence are randomly masked, and the model is trained to predict the original words based on their surrounding context. After pre-training, BERT can be fine-tuned on task-specific data for most NLP applications.

While BERT embeddings have achieved outstanding performance on word representations for several NLP tasks, there are multiple shortcomings in this approach. BERT embeddings have a significant computational complexity and high memory requirements due to their transformer-based architecture, which may represent a significant challenge at training, and deployment on resource-constrained environments. Furthermore, may not always generalise well to specific domains without extensive fine-tuning, and can overfit when dealing with relatively small datasets.

2.2.3 Autoencoders

The autoencoder [Ballard, 1987, Hinton and McClelland, 1987], is a special type of neural network that aims to replicate its output to its input. In its architecture, the autoencoder contains a hidden layer h which can be interpreted as a representation of the data. This architecture is built from two main sub-networks $h = e(x)$ named the encoder network, which aims compress the input data into the hidden layer, and $r = d(h)$ called decoder network, which reconstruct the input data from the hidden layer. Figure 2.4 shows a visual representation of this architecture.

The autoencoder architecture has been widely explored in applications for dimensionality reduction [Hinton and Salakhutdinov, 2006], feature engineering [Torralba et al., 2008], and generative modelling due to its efficiency in learning data distributions and its properties

in a relatively simple manner, particularly when the dimensionality of the input data in considerably large. Differently to the neural networks presented in section 2.1 which are trained via backpropagation, Autoencoders may also be trained with the use of recirculation [Hinton and McClelland, 1987] by comparing the activation of the networks on the inputs with respect to the reconstructed outputs, although this practice is not commonly used by practitioners.

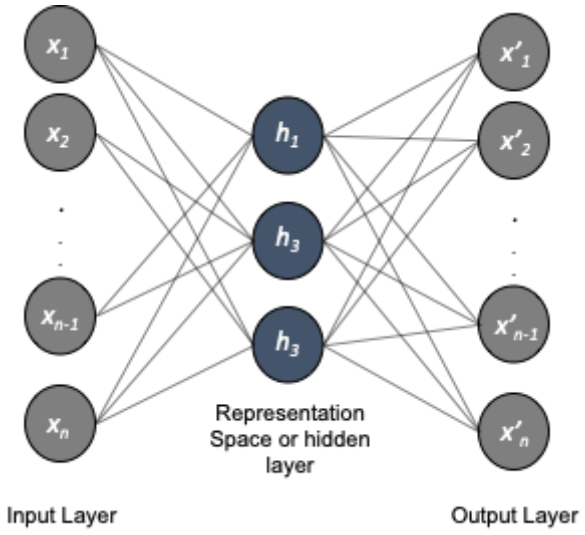

Figure 2.4: Visual representation of the Autoencoder architecture.

2.2.4 Bayesian Neural Networks

As presented in previous sections, neural networks and deep learning have proof their efficiency in achieving outstanding performance in multiple domains. However, neural based architectures also have their caveats, for instance, large architectures are prone to overfitting when the architectures stack multiple layers and become 'too deep' [Szegedy et al., 2014], which could mean that the network become overconfident about their predictions during training and its weights stop updating during the training process. Among several techniques proposed in the literature to alleviate this issue, the Bayesian paradigm provides a robust approach to control the uncertainty of predictions made by the networks.

Bayesian neural networks (BNN's) are stochastic-based neural networks that are trained from data using Bayesian inference. Their main goal is making predictions using a mapping $y = \phi(x)$, where ϕ is a probabilistic distribution. This can be achieved by assigning a stochastic activation in the output of their neurons rather than a single weight value as in the method

presented in section 2.1.3, an illustration of this method is shown in Figure 2.5.

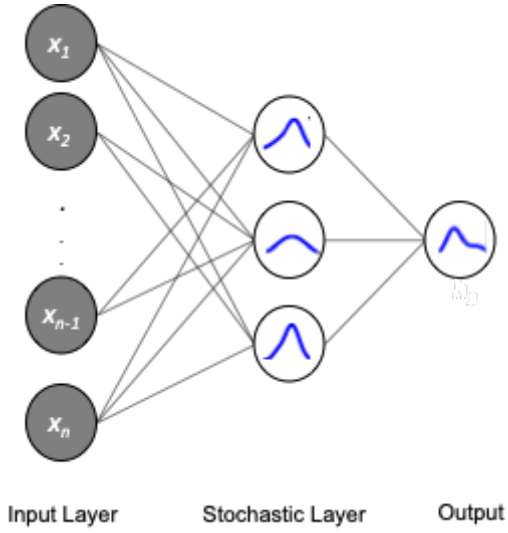

Figure 2.5: Visual representation of Bayesian neural network with stochastic output.

In practice, each node in the network is associated to a prior distribution $p(\theta)$, which are then trained via variational inference [Gal and Ghahramani, 2015] or back propagation with the reparametrization trick [Kingma and Welling, 2013, Rezende et al., 2014]. As BNN's make predictions using probability density function in each of their nodes, these architectures can be considered as an infinite ensemble of neural networks [Doshi et al., 2011],

2.2.5 Variational Autoencoder

In just few years, Variational Autoencoders (VAE's) [Kingma and Welling, 2013] have emerged as one of the most popular approaches in unsupervised learning for complex probabilistic distributions, as their main goal is to express the generative process from which the data was extracted by using a set of latent factors. Surprisingly, the structure of VAEs has little to do with traditional autoencoders [Ballard, 1987, Hinton and McClelland, 1987] presented in section 2.2.3 or to similar architectures like sparse autoencoders [Lee et al., 2006] or denoising autoencoders [Bengio et al., 2014, Vincent et al., 2008], and the reason why the VAE is called an "autoencoder" is due to the fact that their final training objective function is obtained from a encoder-decoder setup, that somehow, resembles a traditional autoencoder shape shown in Figure 2.4.

VAE' s estimate the data distribution $p(X)$ of a random variable X by introducing an unobserved latent variable Z and define a conditional distribution $p(X|Z)$ for the data, also

known as likelihood. Each of the elements of the observed variable X depends only on the latent variable Z. By introducing a prior distribution $p(Z)$ over the latent variables, the joint distribution over observed and latent variables can be obtained with the use of the Bayes' theorem $p_\theta(X, Z) = p_\theta(X|Z)p(Z)$, where the likelihood $p_\theta(X|Z)$ is known as decoder from the latent representation of Z, $p(Z)$ is the prior distribution of Z, and parameter θ are the weights and biases of the decoder neural network. Typically, the decoder network is assumed to be distributed as a multivariate Gaussian with mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{v}$, i.e. $p_\theta(X|Z) \sim N(\boldsymbol{\mu}, \mathbf{v})$, and two neural networks are used independently to estimate its parameters, i.e., $\boldsymbol{\mu} = NN_1(Z)$ and $log\,(\mathbf{v}) = NN_2(Z)$. For simplicity, the prior distribution of Z is often considered a multivariate Gaussian with zero mean and identity covariance matrix, i.e $p(Z) \sim N(\mathbf{0}, \mathbf{I})$, although it does not necessarily need to be the case.

Unfortunately, introducing a non-linear mapping from Z via neural networks induces intractable inference for the posterior distribution, as it is necessary to obtain $\int_\infty p(x|z)p(z)dz$. Instead, VAE uses variational inference to approximate the true distributions of latent factors $p(Z)$ with a distribution $q_\Phi(Z|X)$ which can be obtained by maximising the *Evidence Lower Bound* (ELBO) defined as

$$log\, p_\theta(X) \geqslant -KL[q_\Phi(X)||p_\theta(X|Z)] + E_{q_\Phi(Z)}[log\, p_\theta(X|Z)]$$

where KL is the Kullback-Leibler divergence expressed as follows

$$KL[q_\Phi(x)||p_\theta(z|x)] = -E_{q_\Phi(z)}\left[log\frac{p_\theta(z|x)}{q_\Phi(z)}\right],$$

and where $E_{q_\Phi(Z)}$ denotes the expectation over $q_\Phi(Z)$. In a very similar manner, VAE uses two additional neural networks to parametrise the inference network of latent factors $q_\Phi(Z|X)$ known as encoder, which is usually chosen to be again Gaussian with mean vector $\boldsymbol{\mu_q}$ and covariance matrix $\mathbf{v}_q$. i.e $q_\Phi(Z|X) \sim N(\boldsymbol{\mu_q}, \mathbf{v_q})$, with $\boldsymbol{\mu_q} = NN_3(X)$ and $log(\mathbf{v}_q) = NN_4(X)$, where parameters Φ are the weights and biases of NN_3 and NN_4

As both generative model $p_\theta(X|Z)$ and inference model $q_\Phi(z|x)$ are defined from neural networks, gradients of the ELBO with respect to θ and Φ can be computed with back-propagation by using the *reparametrization trick* [Kingma and Welling, 2013, Rezende et al., 2014] expressing $Z = \boldsymbol{\mu}_\theta + \boldsymbol{\sigma}_\theta \odot \xi$, where $\xi \sim N(\mathbf{0}, \mathbf{I})$. The reparametrization trick allow us to obtain a low-variance differentiable unbiased estimator of the ELBO that simplifies the training process and allows us to back-propagate through a random node in the network. Figure 2.6

shows a visual representation of the reparametrization trick.

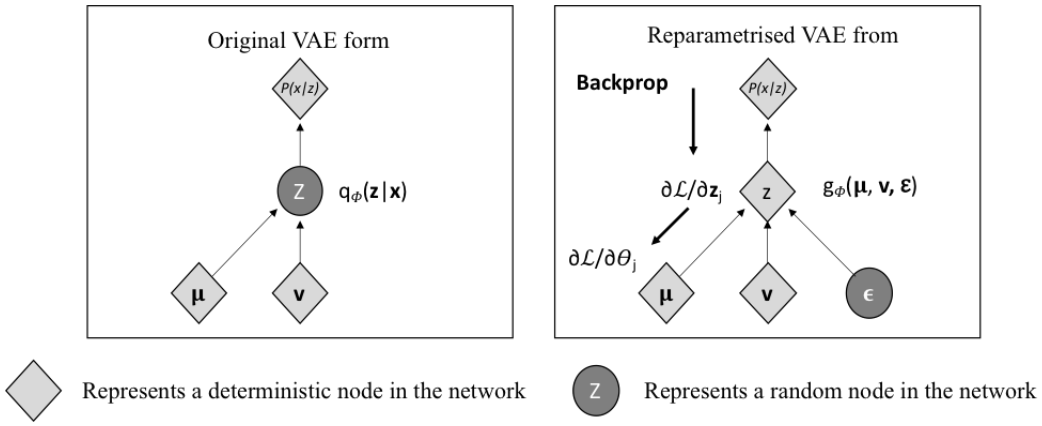

Figure 2.6: Visual representation of reparametrisation trick applied to a random node in of the network.

Finally, at inference time predictions can be obtained simply by computing the expectation from the model with respect to the latent variable Z, i.e. $p_\theta(Y|X) = \int p_\theta(Y|Z)p_\theta(Z)dZ$ where the integration can be estimated via Monte Carlo sampling from the latent layer.

2.3 Survival Analysis and Time-to-event modelling

Survival analysis (SA) is the field of statistics focused in modelling time-to-event data over future lifespans, i.e. estimating the probability of an event occurring beyond a certain time in the future. In contrast to Machine Learning methods such as regression and tree-based models, where all events have already occurred at observation time, survival analysis assumes that the event might not have happened at the time of evaluation for some individuals, but it could happen in the future if the observation period were to be extended, this effect is known as right censoring. Survival analysis estimates the probability of the outcome event not occurring up to a time t and accounts for the presence of censoring in data with a survival function $S(t)$ defined as

$$S(t) = P(T > t) = 1 - F(t),$$

where T is a random variable defined from the distribution of events over time, and $F(t)$ is the cumulative distribution of the event times, which is usually modelled with respect to a set of subject attributes X (a.k.a covariates, or predictors, or features), i.e., $S(t \mid X) = P(T > t \mid X) = 1 - F(t \mid X)$. For a survival function $S(t)$, the hazard ratio, or hazard function $\gamma(t)$ defines the event rate at time t, if the event has not occurred up to time t the hazard functions is expressed as

$$\gamma(t) = \lim_{\triangle t \to 0} \frac{P(T < t + \triangle\, t \mid T \geqslant t)}{\triangle\, t} = \frac{S'(t)}{S(t)},$$

with $S'(t) = -f(t)$ and $f(t)$ probability density function of the events time distribution.

Survival analysis methods can be classified into three main different categories: 1) Parametric methods, which assume a specified distribution of survival times as well as a functional form for model covariates. 2) Semi-parametric methods, which enforce a functional relation between covariates and the survival function, but do not impose a specific form of the hazard function. And 3) Non-Parametric methods, which do not impose any assumption on the survival function nor the covariates distributions. Out of all different techniques proposed in the literature, two common methods used in industry applications are the Cox Proportional Hazard (CPH) model [Cox, 1972] and the [Kaplan and Meier, 1958] estimator due to their flexibility and easiness process at implementation. The CPH semi-parametric form allows to linearly combine distributions from multiple covariates with a baseline hazard

to obtain time-dependant probability estimates of individuals' risk at time t. Whereas the Kaplan-Meier non-parametric structure allows companies and researchers to have a robust baseline and reliable predictions of individuals' risk over time in an easy and scalable way. The Kaplan-Meier estimator $\hat{S}(t)$ of the survival function of individuals is defined by

$$\hat{S}(t) = \prod_{k:t_k<t} \left(1 - \frac{d_k}{n_k}\right),$$

where d_k is the number of individuals that experienced the event at the time t_k and n_k is the total number of individuals at risk at time t_k.

Survival models have been widely used by several companies and marketers around the world to predict customer churn due to their simplicity and flexibility to include multiple covariates into the hazard function estimation. [Van den Poel and Lariviere, 2004] explored the use of proportional hazards to model customer attrition in European financial services. [Wong, 2011] used the Cox regression to identify demographic and temporal covariates that impact customer retention in a telecommunication company with base in Canada. [Jamal and Bucklin, 2006] linked time-dependant covariates from customer service, payments, and recovery systems with the use of a Weibull hazard to identify churning customers for a satellite TV service in South America. [Mavri and Ioannou, 2008] examine potential predictors in customer switching behaviour for the Greek banking sector.

However, CPH does not directly model survival probabilities, but the hazard function of individuals at time t as $\gamma(t \mid X) = \gamma_0(t)exp(X^T\beta)$, which is the probability that an individual will experience the event of interest within a time interval given that the event has not happened up to the beginning of the interval, and it is obtained from a baseline hazard that only depends on time γ_0, and a time-independent function obtained from the individual's covariates as $X^T\beta$, where β is the n-dimensional weights vector associated to each individual covariate. Once the hazard function is known, the survival function can be retrieved with the use of the cumulative hazard function $\Gamma(t) = \int_0^t \gamma(s)ds$, as $\hat{S}(t) = \exp(-\Gamma(t))$.

Typically, the CPH model is fitted in two steps [Kvamme et al., 2019]. Firstly, the parametric part of the model that only depends on individual's covariates $exp(X^T\beta)$ is fitted by maximising the Cox partial likelihood, as it does not depend on the baseline hazard function γ_0, then, the non-parametric baseline is estimated based on the parametric results obtained in the previous step.

Although survival techniques have proofed their efficiency to predict customer churn

accurately in several applications, these methods also have their drawbacks. Firstly, these techniques commonly assume that the event of interest can occur only once and it will happen with probability of one if individuals are observed for enough time, which is not the case in modelling customer behaviour where inherently individuals can make several purchases over their lifetimes [Mavri and Ioannou, 2008, Spanoudes and Nguyen, 2017, Tamaddoni et al., 2010] or not come at all again after certain time. A pragmatic approach to model these issues is by resetting the customers' survival probability to 1 immediately after they make a new purchase and introduce a cure factor which represents the probability that individuals will not make any further purchase [Amico and Van Keilegom, 2018].

Secondly, performance assessment of survival models applied to purchasing behaviour might not be straightforward, as survival techniques estimate the probability of events occurring over time, whereas evaluating traditional customer purchases is modelling a binary classification problem 'customer will make a purchase eventually vs customer will not make a purchase ever again'. Allowing solutions which can be optimal point-wise, but not overall.

Finally, CPH still enforce the use of a constant hazard function for all individuals, which is an unrealistic assumption at modelling purchases for the non-contractual retail industry, where customers can change their buying patterns at any time. Different studies have looked to overcome this challenge by using mixture models on top the original Cox model [Nagpal et al., 2019], by training CPH models in adversarial frameworks [Chapfuwa et al., 2018] or using time-dependant hazard functions [Fisher and Lin, 1999], or combining survival models with network-based architectures, for instance, [Nagpal et al., 2020] propose a fully parametric mechanism called *Deep Survival Machines* to learn non-linear representations of covariates without the need of the strong hazard assumption. Whereas [Ren et al., 2019] propose a deep recurrent survival model to predict the likelihood of an event without assuming any specific distribution on the survival function while accounting censorship presence in data.

2.3.1 Neural Networks in Survival Analysis

Recent research have proposed methods to combine RNN with survival analysis to outperform traditional CPH models and its variants. In these methods, typically the survival model is fully parameterised with the output of a recurrent network to predict the empirical distribution of future events and make use of time-dependant covariates. [Giunchiglia et al., 2018]

proposed RNN-SURV for the medical field, this method takes characteristics of patients over a period of time, at each time-step the model computes both, the risk score, and the survival function of each patient in a personalised manner. [Martinsson, 2017] proposed WTTE-RNN (Weibull Time-to-Event RNN) which consist mainly in the use of a recurrent network to estimate the parameters of a Weibull distribution, then this estimated Weibull distribution is used to predict engines time-to-failure in the field of machinery maintenance. [Chen et al., 2018] proposed MAT-RNN (Multivariate Arrival Times RNN) to extend prediction of survival frameworks to multiple arrivals setting, such as prediction purchases in demand forecasting. [Bennis et al., 2020] proposes a recurrent architecture to model the parameters in a mixture Weibull distribution for time-to-event analysis.

2.3.2 Performance Metrics in Survival Analysis

This section introduces briefly the main methodologies used to evaluate models where censoring is present in the evaluation data. As mentioned previously, accounting for censoring in predicting customer churn induces further challenges in model evaluation overall, as we are not able to fully distinguish customers who are already churned and will not make any further purchase against the ones that are taking a pause between transactions but will return eventually.

Brier Score

The Brier score introduced by [Graf et al., 1999] is a common evaluation metric used in survival analysis to evaluate the accuracy of survival probabilities. It represents the average squared distance between the observed survival status against the predicted survival probability for all subjects. In the absence of censoring, the expected brier score can be obtained as

$$BS(t) = \frac{1}{K}\sum_{k=1}^{K}(1_{\{t_k^* > t\}} - S_k(t))^2,$$

where t_k^* represents the first arrival time for customer k in the validation period. However, to account the presence of censoring in survival models, it is necessary to adjust the score with the inverse probability of censoring weights. For each individual it is considered $t'_k = min(t_k, C_k)$ along with $\delta_k^* = 1_{\{t_k^* \leqslant C_k\}}$, where C_k represents the current time under

observation for each individual k. Let $G(t) = P(C > t)$ be probability of censoring for a time t, usually obtained via Kaplan-Meier estimation [Graf et al., 1999]. The estimated time-dependant brier score for censored data under the assumption that the event of interest will happen con probability of one if individuals are observed for long enough time is defined as

$$BS(t) = \frac{1}{N}\sum_{k=1}^{N}\left(\frac{(0 - S_k(t))^2 \cdot 1_{\{t_k^* \leq t, \delta_k^* = 1\}}}{G(t_k)} + \frac{(1 - S_k(t))^2 \cdot 1_{\{t_k^* > t\}}}{G(t)}\right)$$

Finally, the Integrated Brier Score (IBS) provides an overall estimation of model performance for all times up to a given time t_{max}.

$$IBS(t_{max}) = \frac{1}{t_{max}}\int_0^{t_{max}} BS(t)dt$$

Concordance index

Harrell's Concordance index, also known as C-index or C-statistic [Harrell et al., 1982], is one of most used performance metrics for survival models due to its inherent design to account censoring in data. Contrary to metrics that assess the predictive power of a model by measuring the error in predictions, such as brier score, the C-index assess the discriminating power of a risk score by comparing the correlation between predicted scores $\hat{S}(t)$ and true observed times for pairs of comparable individuals k_i and k_j, with $i \neq j$, who experienced the event at different times t_{k_i} and t_{k_j} respectively. The C-index is defined as:

$$\text{C-Index} = P(S_{k_j}(t_{k_j}) > S_{k_i}(t_{k_i}) \mid t_{k_i} > t_{k_j})$$

where large values of C-index indicate a more informative prediction of which individuals are more susceptible to experience the event of interest.

Time-dependant Area under the ROC curve

The receiver operating characteristic (ROC) is a well-established technique in machine learning to assess classification power of binary classifiers, particularly when the time horizon of the target variable is fixed [Fawcett, 2006]. For an arbitrary classifier $\hat{Y} : X \to [0, 1]$ with binary outcome $Y = \{0, 1\}$ and classification threshold c, the ROC curve compares the classifier sensitivity, obtained as $sensitivity(c) = P(\hat{Y}_1 > c \mid Y = 1)$ a.k.a True Positive Rate (TPR), against the one minus the specificity, where $specificity(c) = P(\hat{Y}_0 \leq c \mid Y = 0)$ a.k.a. False

Positive Rate (FPR), over all possible values of $\hat{Y}$, where $\hat{Y}_1$ is the predicted probability for a positive instance, and $\hat{Y}_0$ is the predicted probability for a negative instance for the defined classification threshold c. Thus, the AUC (Area Under the ROC Curve) can be defined as the total area under the ROC curve, and can be interpreted as the probability that a randomly selected pair of observations are correctly classified by $\hat{Y}$ [Kamarudin et al., 2017], i.e :

$$AUC = P(\hat{Y}_1 > \hat{Y}_0)$$

AUC is an aggregated performance metric over all possible classification thresholds in a model, a large AUC indicates that the classifier possesses a high predictive power to distinguish between different classes.

Different methods have been proposed to extend this metric to consider variable time-horizons and the probability of event occurrence non-constant, such as in survival models presented in [Kamarudin et al., 2017, Heagerty and Zheng, 2005, Lambert and Chevret, 2016]. In simple terms, when extending the ROC curve to a time-dependent outcome, both sensitivity and specificity become time-dependent measures with respect to a time-dependant random variable T, which are defined by *'cumulative cases at t'*, individuals who experienced the event before t, i.e., $t_k^* \leq t$, and *'dynamic controls at t'*, individuals who experienced the event after time t, i.e., $t_k^* > t$. As such, sensitivity and specificity can be expressed as a function of time t [Heagerty and Zheng, 2005] as follows:

$$sensitivity(c, t) = P(S_k(t) > c \mid t_k^* \leq t)$$

$$specificity(c, t) = P(S_k(t) \leq c \mid t_k^* > t)$$

Thus, the Cumulative/Dynamic ROC (C/D ROC) measures how well a model can classify subjects who experienced the event at different points in time, and the Cumulative/Dynamic AUC (C/D AUC) [Heagerty et al., 2000, Hung and Chiang, 2010] provides a single aggregated measure of the total area under of the C/D ROC curve, which represents the probability that the estimation of the non-event will be larger for individuals who have already experienced the event at time t compared against those who have not. The estimated C/D AUC is defined at time t as

$$AUC(t) = P(S_{k_i}(t_i) > S_{k_j}(t_j) \mid t_i \leq t < t_j)$$

As mentioned previously, most survival analysis techniques assume that the event of interest will happen eventually for all individuals, leading into a potential evaluation bias when considering $AUC(t_1, t_2)$ as a performance metric when $t_2 < \infty$, which happens potentially in every real-world application. Thus, if the event of interest might or might not happen for all individuals, metrics like Brier score and C-index might be more useful than the time-dependent AUC in survival applications.

2.4 Recommender systems

Recommender systems are perhaps one of the most successful applications in Machine Learning and Artificial intelligence in the last two decades. These systems have been designed, tested, and implemented successfully in a wide range of application domains to expose users to a large collection of relevant items, and are particularly helpful when the catalogue of potentially recommendable items is large enough to make it impossible to do recommendations manually for humans. Recommender systems in general are recognised as an efficient approach to provide users with personalised content. For example, the streaming service Netflix displays user-level predicted movie ratings to its customers to help them in deciding which content is more suitable and interesting to watch. The global online retailer Amazon provides predicts item ratings to users based on the previous purchase history of similar customers. As the final goal of these tools is to suggest which items are more suitable to individual users or which items will be liked by customers, these systems are typically categorised as recommender systems.

Recommender systems have their roots in the field of information retrieval; identifying which online written content might be most relevant with respect to a given user query, and then sorting the retrieved list of top relevant documents based on easy user consumption. However, although this approach was widely used in the '90s by different companies for recommending specific items, it was quickly replaced with more advanced techniques. These include content-based Collaborative Filtering (CF) [Balabanovic and Shoham, 1997] and Matrix Factorization (MF) [Koren et al., 2009] recommender systems, which aim to learn a relationship between user preferences and items by using historical information of user's actions and purchases in a matrix-completion framework, such as Singular Value Decomposition (SVD) [Zhou et al., 2015], where the goal is to predict future user preferences in a user-item rating matrix. These well-established techniques also have their strengths and weaknesses, and many researchers, companies, and AI practitioners have chosen to combine techniques in different ways to provide better recommendations for users and increase either overall revenue, customer engagement, or model performance. This led into the development of Hybrid Recommender systems [Burke, 2002], where techniques such as weighted recommenders [Claypool et al., 1999], mixed recommenders [Smyth and Cotter, 2000], and feature combined recommenders [Basu et al., 1998] are perhaps the most popular methods

used in the industry. Although these techniques usually can deal with some of the issues in recommender systems, such as the cold-start problem of adding new items or users into the system [Jazayeriy et al., 2018], they still have several performance issues in several application domains [Marchand and Marx, 2020].

Across industry, recommender systems are powerful tools to enhance user experience via personalisation and increase sales and overall revenue by identifying which items are most likely to be relevant for users [Aggarwal, 2016, Gunawardana and Shani, 2009, Zhao et al., 2014]. Mirroring its use in other applications such as computer vision and natural language processing (NLP), deep learning is capable of great achievements in the field of recommender systems, where uncovering non-linear complex relationships between user-items interactions with the use of deep neural networks can easily outperform longer-standing techniques, and these models are capable of learning complex user-item relationships from the usually-abundant data itself [Zhang et al., 2019]. Implementation of deep learning for recommender systems have proven to significantly outperform other techniques without requiring major efforts at the deployment stage. Further detail of these kind of approaches is included in the third section.

Despite techniques such as CF and MF empowered with the use of deep learning have achieved tremendous success in real-world applications at being able to capture nonlinear user-item relationships, these still have their caveats and tend do not perform well when users' preferences change over time, or when users interact several times with only a few items within the total catalogue, which is commonly the case for many specialised retailers and small and medium enterprises (SMEs) settings. Furthermore, as CF and MF are matrix completion approaches, these commonly make the assumption that the exact user preference of items, a.k.a user rating, is known for all user-item interactions in the data. There has been research to propose synthetic ratings for user-items pairs [Su and Khoshgoftaar, 2009, Sidana et al., 2021], such as by creating functions which only depend on how many times users have purchased or interacted different items. The construction of these synthetic ratings is often completely arbitrary and can lead to unrealistic assumptions and inaccurate predictions when the systems are tested in real-world scenarios .

2.4.1 Notation

In the typical recommendation setting, there is a set of users $U = \{u_1, u_2, ..., u_{|U|}\}$ with size $\mid U \mid$, and a set of items $I = \{i_1, i_2, ..., i_{|I|}\}$ with size $\mid I \mid$, although most of the time due to business constraints, it is useful to focus on ranking only a subset of potentially recommendable items $I' \subset I$, due to the fact that items might not be available at all times or are discontinued from stock. Generally speaking the goal of a recommender system is to produce a list of *relevant* items $L_{u_j} \subset I$ for each user $u_j \in U$. This is typically achieved by learning a mapping function $r(X_U(u_j), X_I(i_k)) : X_U \times X_I \rightarrow \mathbb{R}$ that can assign a prediction of how *relevant* the item i_k is for user u_j, where X_U and X_I represent the feature space of the total information available for users and items respectively, such as user's ratings, user and item characteristics, and hand-crafted features from data.

Once predictions are computed, the recommendation list L_{u_j} can be obtained by sorting items from highest to lowest *relevancy* for each user u_j. The following sections explore different methods used in research and industry applications to learn this function r from implicit feedback settings.

2.4.2 Implicit Feedback Recommender Systems.

In the early 90's, recommender systems relied heavily on the use of explicit feedback of recommendations collected from user's ratings and reviews for each item [Balabanovic and Shoham, 1997, Resnick et al., 1994, Claypool et al., 1999], which typically are a clear indicative of the likeness or preference to a product of its characteristics, and an indicator of customer loyalty [Ravula et al., 2022]. However, with the large scaling and overwhelming use of recommender systems by companies, obtaining reliable explicit feedback is getting more difficult over time. Furthermore, for some domains, it is inherently difficult or impossible to obtain explicit feedback from users. For example, in the retail environment users typically include a large number of items in their shopping basket for a single visit to the store. Asking a user to provide feedback for each of the items after the purchase was made would not be practical for the user or beneficial in terms of user experience. In this case, we must rely solely on implicit feedback. This may be obtained as a pseudo-rating or estimated with a function typically defined arbitrarily during the data pre-processing phase and computed from different data user's signals, such as number of purchases for an item or different types of events carried

during an online session.

Several implicit feedback-based recommender systems have been proposed due to the scarcity of reliable explicit ratings provided by users. For instance, [Nunez Valdez et al., 2018] explore different alternatives to recommend electronic books using implicit feedback obtained from the logging information of users in an e-commerce platform, such as duration of the session, number of clicks, users reading time, number of comments. [Hu et al., 2008] proposed obtaining information about the positive or negative preference of users associated via an association with varying confidence intervals to recommend television shows at large scale. [Lee et al., 2008] performed item recommendation on an e-commerce platform with the use of a pseudo-rating and introducing temporal information by including the time that users interact with items and the time elapsed since the item was initially introduced in the platform, affecting the recommender accuracy by promoting brand new items to users.

2.4.3 Matrix-Completion Recommender Systems

Collaborative Filtering

Collaborative filtering was proposed in the late 90's as a convenient alternative to content-based and feature-based recommender systems. These had mostly focused on extracting human-engineered characteristics of users and/or items data, aiming to find similarities between purchases to predict future users' relevant items. Instead, collaborative filtering methods such as *GroupLens* [Resnick et al., 1994] aim to estimate how much a user will like a specific item based on how much a set of users liked similar items previously. In Collaborative filtering, ratings are estimated from the user-item matrix shown in Figure 2.7, where the known previous user ratings are denoted as $r_{u,i}$ for user u and item i, and future unknown ratings $\hat{r_{u,i}}$ are estimated as

$$\hat{r}_{u,i} = \frac{\sum_{j \in I} r_{u,i} \cdot w_{i,j}}{\sum_{j \in I} w_{i,j}},$$

where $w_{i,j}$ is the similarity between the item i and an item $j \in I$, which can be obtained via any similarity function such as cosine similarity, Jaccard similarity, KL divergence, among others.

In practice, estimated ratings $\hat{r}_{u,i}$ are obtained only from a subset of items $J \subset I$ usually called *neighbourhood of item i* rather than from the whole set of items, this to improve scalability

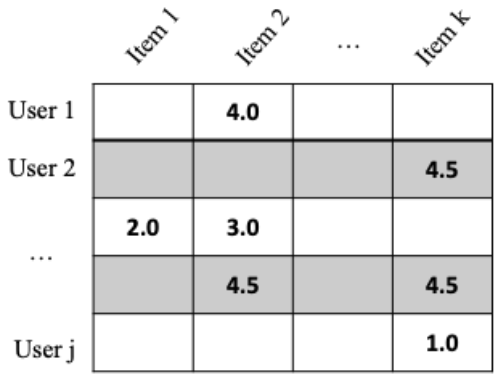

Figure 2.7: User-item matrix of ratings representation for a set of users $u \in U$ and a set of items $i \in I$

and reduce variability at using only the most similar items.

Several successful applications in industry have used collaborative filtering to improve sales and revenue. For example, [Linden et al., 2003] deployed an *item-item collaborative filtering* recommender in the large e-commerce *Amazon.com* to personalise the content that is displayed for each of the millions of users visiting the website for shopping on a daily basis regardless of the number of ratings or purchases made by previous customers. Additionally, collaborative filtering methods can be easily combined with a large number of machine learning techniques such as clustering [Ungar and Foster, 1998], latent semantic analysis [Hofmann, 2004], or Markov decision processes [Guy Shani, 2005] to improve rating estimation performance and scalability.

Although CF techniques are relatively easy to implement in production environments and have proven to improve revenue and customer satisfaction in several applications for different industries, these methods have several limitations. Firstly, the fully known user ratings should explicitly and reliably express users' preferences for items, which is usually difficult to achieve in implicit feedback settings, as typically the rating is only obtained from users' signals. Secondly, as future users' ratings are directly estimated from items similarities $w_{i,j}$ highly sparse datasets where only few items are rated by individuals induce extra variability in estimated ratings, leading into overall lower performance and miss leading recommendations.

Matrix Factorisation

To overcome the data sparsity issue in recommender systems, different approaches implement dimensionality reduction techniques such as *Singular Value Decomposition* (SVD) and *Principal Component Analysis* (PCA), to compress the highly sparse user-item interactions matrix into a low-dimensional dense representation of users and items.

Matrix Factorisation [Koren et al., 2009] characterises both users and items by latent factors of dimension k directly inferred from the user-items interactions matrix, in such a way that unknown ratings can be easily estimated by the inner product of users and items latent factors. Mathematically, each item $i \in I$ is associated with a hidden latent vector $q_i \in R^k$, and each user $u \in U$ to a vector $p_u \in R^k$, where for each user and item, the corresponding vectors p_u and q_i measure to what extend the u and i associate to the corresponding latent factors positively or negatively, and typically $k << min(\mid U \mid, \mid I \mid)$. Thus, the estimated user rating for an item, can be estimated as the *'similarity'* between user and item latent factors, i.e.,

$$\hat{r}(u,i) = q_i^T \cdot p_u$$

At training phase, latent factors can be learnt through minimising the error between observed and predicted ratings for which the rating is fully known, typically by using the mean squared error ($MSE = \frac{1}{|U| \times |I|} \sum_{j=1}^{|U|} \sum_{k=1}^{|I|} (r_{u_j,i_k} - \hat{r}_{u_j,i_k})^2$) as loss function. Then, recommendations can be made by selecting the predictive ratings for which the inner product of latent factors is the largest. Several pieces of research have extended this concept: [Bell et al., 2007] combine a neighbourhood-based collaborative filtering with SVD MF at a higher level to improve estimation performance on large datasets without performing any imputation for missing ratings and avoid parameter shrinkage at training. [Paterek, 2007] proposed a weighted SVD by including additional biases to SVD and additional post-processing via kernel ridge regression for each item.[Salakhutdinov and Mnih, 2007] introduced probabilistic matrix factorization (PMF) which includes adaptive priors in model parameters to outperform SVD while maintaining model scalability for large datasets even for users with few item interactions.

2.4.4 Deep learning-based recommender systems

As mentioned in the first section, recommender systems and neural networks such as feed-forward neural networks (FNN), convolutional neural networks (CNN), and recurrent neural networks (RNN) can be combined in order to find nonlinear and non-trivial user-items interactions and provide better recommendations to users [He et al., 2017, Covington et al., 2016, Hariri et al., 2012, Zhang et al., 2019]. These types of architectures typically built from multiple neural building blocks can be defined into single differentiable function, and trained end-to-end with classification or ranking losses to foster recommendation performance and item lists sorting, besides having the ability to incorporate data from multiple shapes like users reviews, tweets, item images, or sound in their input data, aiming to resemble the behaviour and benefits of hybrid-recommender systems while avoiding expensive human-based feature engineering.

Thanks to the easy accessibility to deep learning frameworks such as *Tensorflow* and *Pytorch*, and the increasing computational power available in modern computers, deep learning-based recommender systems have been applied in several research and commercial applications for different industries over the last decade. For example, [He et al., 2017], replaced the inner product in collaborative filtering with a multi-layer perceptron (MLP) that can learn an arbitrary function from data and find non-linear relationships in the user-item matrix and outperform several Collaborative filtering methods. [Covington et al., 2016], used two neural networks combined in a candidate-ranking classification framework to produce highly scalable recommendations for users of the large video streaming platform *YouTube*, and reduce the ranking space from several millions of items to just a few thousand. In this work, authors use information about different types of users' actions combined with item embeddings to produce a list of relevant items with high precision. [Volkovs et al., 2017] proposed a method called *DropoutNet* that combines matrix factorisation and neural networks to address the cold-start problem under the assumption that not having information of users is similar to handling performance missing data efficiently.

2.4.5 Sequence-aware recommender systems

Sequence-aware recommender systems have emerged as a powerful approach for modelling and predicting user behaviour by leveraging the order of user-item interactions, provide

personalised recommendations based on users' preferences, and overcome many of the caveats of matrix-completion and deep learning-based systems, which often fail to capture the sequential nature of user behaviour. Over the last few years, there has been a vast amount of research around sequence-aware recommender systems, [Guy Shani, 2005, Wu et al., 2017, Quadrana et al., 2018, Zhang et al., 2019, De Souza Pereira Moreira et al., 2021] particularly in the context of implicit feedback, which is when the exact user rating of items is unknown for all or most user-item interactions in the data.

Sequence-aware recommenders have been widely researched over the last years for multiple application domains powered by several ML methods like Markov chains, recurrent neural networks, or transformer-like architectures. For instance, [Guy Shani, 2005] proposed a recommender system based on a Markov Decision Process (MDP) in a reinforcement learning framework to improve revenue of recommended items in an online bookstore. [Baeza-Yates et al., 2015] studied how to improve mobile application usage to provide personalised user experience via prediction of which application is likely to be used in the near future by the user, with the use of popular recommendations and session-based feature engineering authors can outperform different prediction methods like Naive Bayes and Support Vector Machines (SVMs), besides approaching the cold-start problem of having new apps constantly. [Hariri et al., 2012] presented a context-aware recommender system for music recommendations by analysing sequences of previous songs listened by the user within the current session and a database of human-compiled playlist mapped into sequences of topics, achieving better recommendation performance than collaborative or content-based filtering methods. [Wang and Zhang, 2013] proposed a repeated-interaction recommender system for the e-commerce industry which combines the proportional hazard assumption from survival analysis to model the joint probability of user interacting with items over a period of time.

Taking inspiration from NLP, sequence-aware recommender systems are inherently different from the traditional matrix-completion approaches. Sequential methods process customer transactions as sequential information by considering each item in the catalogue of products as a single word in a dictionary or token [Quadrana et al., 2018], and taking all transactions made by each customer to build a sequence of tokens, similarly to processing sequences of words. Then, the new sequences generated for each user can be used to embed customers into a common representation space and make predictions of the most likely token or item to appear next in the sequence, and item recommendations can be as simple as selecting the

most next likely item to be part of the sequence of purchased items. [Quadrana et al., 2018] categorise the sequence-aware recommendation setting into four different categories: *Context adaptation*, when it is important to understand the context of the user, such as geographic location, the current weather, or the time of day to make relevant recommendations. *Trend detection*, where it is critical to have information about community and individual trends of popular items. *Repeated recommendations*, which is when users might interact repeatedly with each item in a single or multiple sessions. *Order constrained*, where the actual order on which user's actions were made reveal the inherent most likely action to be taken next by the user.

Advantages of sequence-aware recommender systems

Due to their inherent ability to capture user preference over time, sequence-aware recommender systems offer several advantages over traditional collaborative filtering and content-based approaches:

Model temporal dynamics in data: As user preferences and item popularity might change drastically over time, such as in social media recommendations, sequence-aware recommenders can capture these temporal dynamics by considering the order of user-item interactions, allowing them to make more relevant item recommendations [Hidasi and Karatzoglou, 2018].

Contextual Information: By incorporating the order of user-item interactions over time, these systems can better understand the context in which a user is likely to be interested in an item, leading to more personalized and accurate recommendations [Quadrana et al., 2018, Kim et al., 2022].

Handling Cold-Start Problem for users: Matrix-completion methods typically rely on past user-item interaction, which might be an issue in cases where limited or no historical data is available for new users or items. Sequence-aware recommenders can overcome this by incorporating short-term interaction sequences within a session, providing meaningful recommendations even for new or anonymous users [Quadrana et al., 2018].

Handling short-term user preferences: Matrix completion recommender systems typically focus on long-terms user preferences, and although the scope of the training data can be modified to capture shorter user preferences, in most cases final predictions may not accurately reflect users' current interests. Sequence-aware recommender systems can

capture short-term preferences by analysing recent interaction sequences, allowing them to recommend items that better reflect immediate users' needs or changing preferences.

Cross-domain Recommendations: Sequence-aware recommender systems can identify sequential user preferences across different domains, enabling cross-domain recommendations. For example, users' browsing history could potentially be used to recommend relevant videos on a streaming platform, providing targeted recommendations and enhancing user experience [Fernandez-Tobias et al., 2012].

Disadvantages of sequence-aware recommender systems

Nevertheless, sequence aware recommender systems have their drawbacks against other methods. Despite their ability to provide more accurate and incorporate efficiently the context available, they also face challenges related to computational complexity, data sparsity, and sensitivity to hyperparameters [Quadrana et al., 2018]:

Computational Complexity: These systems often require complex models to capture the sequential patterns in user preferences, leading to increased computational complexity that can result in longer training times and higher resource requirements, particularly when dealing with large-scale datasets.

Sensitivity to Hyperparameters: Many sequence-aware recommenders, especially those based on deep learning models or transformer-like architectures, have numerous hyperparameters that need to be fine-tuned for optimal performance. Leading into requiring extensive experimentation and validation to find the right combination of hyperparameters for each application domain.

Data Sparsity: Although sequence-aware recommender systems can overcome the cold-start problem for new users in the system, they still face challenges at dealing with sparse data. Sparse interaction data can make it difficult to identify meaningful sequential patterns and generate accurate recommendations, particularly for items with few interactions.

Simultaneous Events: When there are event ties in the training data, i.e., item interactions for the same user that happened simultaneously, sequence-aware recommender systems may face challenges in accurately modelling the user behaviour. This effect might be present in

data due to multiple factors, including the granularity of timestamps available in data, which may not be capture the exact order of events, and inherent user behaviour that involves performing multiple actions concurrently, such as users listening multiple songs on a music streaming platform, or interacting with multiple chatbots sessions within a short period of time.

There are some alternatives to handle simultaneous events in training data, such as aggregating them into a single interaction, or randomising the order of simultaneous events within interaction sequences. However, these approaches may result in the loss of sequential information captured by the system and potential decrease in recommendation performance. Some attention-based mechanisms or transformer models, may inherently handle event ties due to the masking process at training to mask randomly a proportion of items in the input sequence during training, and predict those based on the surrounding context within the user-item sequence in both directions [Zhou et al., 2020].

2.4.6 Performance Metrics in Recommender systems

Evaluating recommender systems introduces an extra level of complexity compared to evaluating traditional classification or regression techniques for a number of reasons: The potential absence of knowledge about the real user preference over items in implicit feedback settings may induce algorithmic bias in model evaluation, as it is not possible to compare model predictions against the explicit user-item rating via error metrics like MSE used in matrix-completion techniques. Different techniques used to perform recommendations may also perform significantly differently depending on some characteristics of the dataset like number of users and items, data sparsity, rating scale, among others. This may lead to having to depend on scenario-specific metrics and manual analysis for evaluating recommendations. Finally, popular evaluation metrics used for recommender systems completely ignore the fact that most users have not had the opportunity to interact with products due to unfamiliarity to them, and not due to a lack of preference. Furthermore, these metrics are designed only for off-line evaluation from previous user-item interactions and might not be scalable to production settings where efficiency must be tested immediately, and designing on-line evaluation experiments to assess recommendation performance might require much more effort and usually these are more expensive.

[Fouss and Saerens, 2008] propose performance evaluation of recommender systems by

taking into account different characteristics like coverage, to measure the percentage of the dataset for which the recommender is able to provide recommendations, computing time to measure how quickly the system can make recommendations for large set of users, and robustness to assess how good the model is in presence of added noise in the data.

For the purpose of this work, some of these metrics might or might not apply in our case, as we are trying to estimate the probability of user interaction with items, rather than item relevance or how much the user will like each item. The following sections briefly outline the performance metrics used to evaluate model performance and user recommendations for our approach from a technical perspective.

Normalised Discounted Cumulative Gain

Introduced by [Järvelin and Kekäläinen, 2002] the normalised discounted cumulative gain (NDCG) is a popular ranking quality metric widely used in information retrieval and recommender systems to evaluate list of items with length p, it takes into account the degree of relevance of items via an information gain function (Discounted Cumulative Gain) defined as

$$DCGp = \sum_{k=1}^{p} \frac{relevance_{i_k}}{log_2(k+1)},$$

as well as the ranking of relevant items in the list via a discount function with respect to the best ranking possible (Ideal Discounted Cumulative Gain).

$$IDCGp = \sum_{k=1}^{REL_p} \frac{relevance_{i_k}}{log_2(k+1)},$$

where REL_p represents the list of all possible relevant items up to position p and $relevance_{i_k} = 1$ if the item is considered relevant for the user, and 0 otherwise. Thus, the normalised discounted cumulative gain is defined as

$$NDCG_p = \frac{DCGp}{IDCGp},$$

which ranges from 0 to 1, and where higher values of NDCG are associated with a better ranking of items in the recommendation list.

Mean Average Precision

The Mean Average Precision (MAP) is a metric originated in the field of information retrieval, it provides insight about how relevant a list of items is with respect to all possible user queries. In recommender systems evaluation, this metric evaluates how good recommendation lists of K items are by obtaining the mean of the Average Precision (AP@K) for each each list defined as

$$AP@K = \frac{1}{rel_K} \sum_{k=1}^{K} \frac{\text{\# of relevant items at k}}{k},$$

where rel_K is the number of total relevant items in the top K results, and can be interpreted as the proportion of relevant items for the user in the recommendation list.

Sales and Revenue

As mentioned in section 2.4, one of the main goals of recommending the right items to users is aiming to increase overall sales and revenue of the businesses [Gunawardana and Shani, 2009], this can be achieved either by cross-selling relevant items during the purchase order, by renewing a subscription to a service due to the highly personalised offering, or by displaying advertising with the right offers to customers to encourage them to take an action, such as purchase an item that they were not thinking in buying or by staying shopping longer. A common practice to test recommender systems performance in production settings is via A/B testing, where two recommenders are in charge of pushing item recommendations to users automatically and overall sales and revenue are tracked for both samples of users.

CHAPTER 3

Modelling customer churn for the retail industry in a deep learning based sequential framework

3.1 Introduction

Finding innovative methods to mitigate customer churn and improve retention has historically been an active task for businesses across a wide range of sectors such as finance, technology, banking, insurance, among others. Particularly, in the retail sector, different studies such as the ones conducted by [Reichheld, 1990, Van den Poel and Lariviere, 2004] have shown that acquiring new customers is usually between 5 to 12 times more expensive for companies than retaining existing ones. Although this ratio varies across companies. Marketers recognise several advantages of dedicating major efforts to identify which customers are likely to churn in order to design more competitive marketing strategies for customers. [Reichheld, 1990] showed that an improvement of just 5% in customer retention leads to an increase of 85% in profits for the banking sector, 50% for insurance brokerage, and 30% in the automotive industry. Unfortunately, the underlying reasons that lead customers to churn might vary for different businesses and industries, although marketers have identified that bad customer service, poor value proposition, low-quality communications, and lack of brand engagement as the main four reasons for voluntary customer churn.

A common approach used by companies to identify future churning customers in non-contractual settings such as retail, where customers are not subject to a subscription model

and can change their purchasing habits without informing the company, is by using statistical and machine learning methods to predict which customers are likely to stop doing business with the company within a certain time window. Then, individuals with high probability of churning can be targeted in one or several retention campaigns which commonly offer product promotions specifically designed to provide an incentive to customers to make a purchase and keep them engaged with the brand, as noted by [Borah et al., 2020]. While the concept of predicting customer churn is relatively intuitive, the realities involved in designing such systems can be challenging for multiple reasons. Firstly, in a non-contractual setting such as retail, customers can change their purchasing habits at any moment, and typically the longer a customer takes to make their next purchase, the lower the probability is of that customer returning at all. Secondly, the use of multivariate statistical methods and machine learning techniques involves the extraction of hand-crafted characteristics for each customer that are usually proposed by subject matter experts. The quality, quantity, and type of these characteristics will have a direct impact on the final performance of any classification method used to find churning customers [Bengio et al., 2013]. Furthermore, as customers' behaviour differ in different sectors and in different companies, finding a relatively good set of characteristics that generalizes well across companies may be difficult to find and would thus lead one to the time-intensive task of finding such characteristics in each new context.

To overcome these issues, several research have explored the use of methods such as artificial intelligence, artificial neural networks, and representation learning to obtain customer representations without the need of human intervention, with the main goal of avoiding the time-consuming feature engineering step that companies need to carry while designing machine learning models and assuming that AI will change marketing strategies and customer behaviours over the next decade as noted by [Davenport et al., 2019]. For instance, [Spanoudes and Nguyen, 2017] use abstract feature vectors in a 4-layer neural network to predict which customers are likely to churn in a monthly defined horizon. However, this and similar methods like the ones proposed by [Coussement and Van den Poel, 2008, Hung et al., 2006, Tamaddoni et al., 2010] do not take into account that the event of interest might have not happened yet for all individuals at observations period, as it is considered in survival-based techniques proposed in the literature, such as the Kaplan-Meier estimate that has also been used to predict customer churn due to their inherent design to model probabilities of an event occurring over a lapse of time without the extra complexity of

obtaining large amounts of features to represent customers as in [Jamal and Bucklin, 2006, Wong, 2011, Gul et al., 2020].

To effectively predict customer churn and design targeted marketing campaigns, it is important to design more effective methods that consider both, the inherent purchasing behaviour of individual customers and the censoring effect induced by the uncertainty of not being capable to identify customers that have already ended their relationship with the business against the ones that simply are during a pause between transactions. Traditional techniques to predicting which customers are likely to churn soon, usually approach just one of these two desired characteristics.

This chapter suggests a new approach for modelling customer churn in non-contractual settings, such as the retail industry, with two main goals. Firstly, outperform traditional machine learning and survival-based methods that use hand-crafted features extracted from behavioural information of customers, this is achieved by replacing the feature engineering phase with a deep learning-based approach that performs model parameters estimation with the use of recurrent neural networks. Secondly, obtain reliable time-to-event models capable of capturing the real buying patterns of customers over time by obtaining individual-level distributions of each customer's arrival times.

3.2 Customer churn prediction methods

Predicting and mitigating customer churn has become an essential activity of modern marketing strategies for businesses that operate in non-contractual settings, such as retailers, as in these types of environments, customers have the freedom to stop their relationship with the business at any point [Tamaddoni et al., 2010]. By making use of machine learning models, businesses can obtain valuable insights of customer's behaviours, identify potential churn risks, and develop retention campaigns to provide customers with tailored products and services and minimise revenue losses [Lu, 2002].

There are numerous methods widely used to predict customer churn in research and industry, each with its strengths and limitations, although ultimately the quality of the final predictive model is highly dependent on the quality of data used [Hashmi et al., 2013]. Broadly speaking, churn prediction methods can be classified into three different categories depending on their underlaying goals and processing of the data:

Behavioural Techniques: These methods focus on analysing customer behaviour patterns to identify potential churn risks. They typically involve segmenting current customers based on their past interactions with businesses and specific customer related attributes, such as purchase frequency, amount spent in a recent period of type, income level, age, and gender. For example, a widely used technique by marketers in this category is the Recency, Frequency, Monetary (RFM) analysis, a simple technique that segments customers based on their current purchase behaviour [Naz et al., 2018].

Although behavioural methods are relatively simple to implement in production environments and their outputs are typically highly interpretable, these techniques have their disadvantages, such as their limited predictive power due to their structure being based on a simple segmentation and not beings able to capture complex relationships between various customer attributes, or their lack of adaptability as these methods typically do not account for changes in customer behaviour over time and new market conditions [Naz et al., 2018].

Probabilistic techniques: These methods focus on estimating the probability of customers making their next purchase based on past transactions, statistical distributions, and time-to-event data. For example, the Pareto/Negative Binomial Distribution (P/NBD) estimate the probability of customer churn based on the frequency and recency of past transaction by assuming a gamma-distributed purchase rate of time between customer purchases and a Pareto distribution for the customer drop-out rate [Schmittlein et al., 1987]. Alternatively, Markov Models model the transition probabilities between different states of customer behaviour, such as active, inactive, or churned to predict their churn status, and the prediction of the next customer churn status only depends on the current state [Rothenbuehler et al., 2015], and survival models estimate the probability of a customer making their next purchase based on the time elapsed since the last event and the total duration of a customer's relationship with the company, as highlighted in section 2.3

However, these methods come with certain caveats. Probabilistic models are typically assumption-driven, relying on specific assumptions about customer behaviour and data distributions of event times. If these assumptions do not align with a particular business or data available, predictions may be inaccurate and difficult to implement due to the computational challenges associated to their parameter estimation [Fader et al., 2005]. Additionally, these techniques often have limited feature consideration, focusing primarily on frequency and recency of transactions while potentially ignoring other factors influencing customer churn

like customer demographics or current engagement.

Classification-based techniques: These methods leverage supervised machine learning and classification algorithms to classify customers into different churn risk categories by identifying complex patterns and relationships in customer data. These techniques can handle large-scale, high-dimensional data and often provide more accurate predictions than probabilistic and behavioural methods[Naz et al., 2018]. Decision Trees and Random Forests, Support Vector Machines, Gradient Boosting Machines (GBM), and neural networks are the popular choices among practitioners due to their flexibility to incorporate a variety and large volumes of data easily [Hadden et al., 2006, Shaaban et al., 2012, Spanoudes and Nguyen, 2017].

For example, survival trees [LeBlanc and Crowley, 1993] are a class of decision tree-based models specifically designed for analysing time-to-event data, where the primary objective is to predict the time until an event of interest occurs. Their main difference against conventional decision trees lies in the splitting criteria used to determine the optimal branch partitions, while standard decision trees typically rely on criteria such as Gini impurity or information gain, survival trees implement integrated squared error of the Kaplan-Meier estimator to take into account the main characteristics of time-to-event data. Particularly in customer churn prediction, Survival trees offer a non-parametric and flexible approach to modelling survival data aiming to estimate the time until this event occurs for each individual.

Most modern machine learning approaches are inherently designed to handle multiple features simultaneously. By incorporating hand-crafted customer attributes such as demographics, product preferences, engagement metrics, and historical purchase data, these models can identify patterns and relationships that may not be apparent to behavioural and probabilistic approaches. Although as aforementioned, the overall effectiveness of the churn prediction will ultimately depend on the effectiveness, quality, relevance of the attributes.

Unfortunately, classification-based techniques often require of large amounts of high-quality data for training and validation, and features typically hand-crafted by domain experts, which can be challenging to obtain for businesses with limited historical data or those without well-developed data culture, strategy, and governance. Additionally, interpretability challenges arise with many advanced machine learning techniques, such as deep learning models and support vector machines, making difficult for businesses to understand the factors driving churn predictions and develop targeted retention strategies accordingly.

The process to select the appropriate method to make customer predictions commonly involves several factors to ensure the effectiveness and accuracy of their predictions, such as the type and quality of the available data, the complexity of customer behaviour patterns, and the desired level of interpretability for the results. For instance, when dealing with small datasets characterized by limited features, simple models such as RFM (Recency, Frequency, Monetary) analysis and segmentation methods may prove to be adequate choices. However, when the data available is high-dimensional or highly available, more advanced techniques like neural networks and survival models may be required to effectively capture the intricacies of complex customer behaviour.

3.3 Methodology

This section describes the mathematical representation of customer transactions data and the model architecture used to estimate how likely individuals are to make purchases over time. We aim to train a bayesian neural network capable of estimating the parameter of exponentially distributed arrival times by using the information of previous observed and censored event times. Then, this network can be used to predict the next event time and therefore, estimate the survival distribution for future events at customer level.

3.3.1 Data Representation

Let's D_K be the set of all purchases made by K customers in a portfolio. In the non-contractual setting, purchases can happen at any date and time, thus, let's denote the date that customer $k \in K$ made its $i-th$ purchase as $d_{k,i}$ as shown in Figure 3.2. Typically, transnational data of all customers is arranged in a single dataset with three columns *'Customer id'*, *'Order Number'*, and *'Purchase date'* as shown in Table 3.1. This dataset is sorted by each customer id in date-ascending order, which means that $d_{k,i_1} \leqslant d_{k,i_2}$ for all $i_1 \leqslant i_2$ for a fixed customer k. Although some of the time-to-event methods described in section 2.3 approach the churn prediction problem with this form of data, in this work it is necessary carry an extra transformation by defining a random variable T_k as the time difference between consecutive customer transactions, i.e., $t_{k,i} = d_{k,i} - d_{k,i-1}$ for $i \geqslant 2$. This new random variable T_k is assumed to be exponentially distributed at customer level, i.e., $T_k \sim Exp(\lambda_k)$ as illustrated in Figure 3.1.

To account for right censoring of purchasing events, we consider the time elapsed since

customers made their last purchase against the observation date, and define this distance as $t_{k,n_k+1} = analysis\ date - d_{k,n_k}$ as the censored time for all customers.

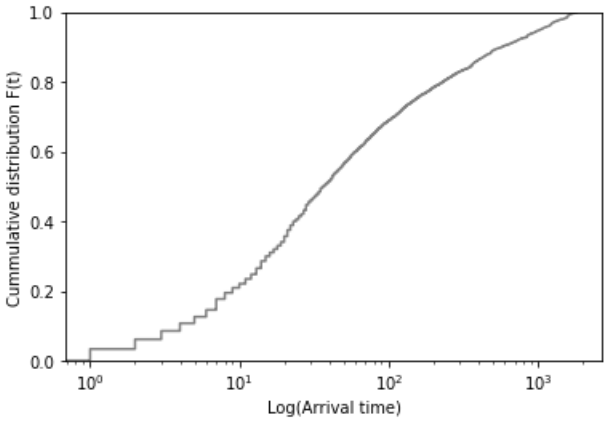

Figure 3.1: Distribution of time between consecutive customer purchases $t_{k,\cdot}$ in logarithmic scale.

Table 3.1: Original Transactional data

Customer ID	Order No.	Purchase date
1	0	$d_{1,0}$
1	1	$d_{1,1}$
2	0	$d_{2,0}$
...	...	...
k	i	$d_{k,i}$
k	i+1	$d_{k,i+1}$
...	...	...

Table 3.2: Customer sequential-transactions

Customer	Arrival-times Sequence	δ_k-sequence
1	$[t_{1,1}, t_{1,2}, ..., t_{1,n_1}, t_{1,n_1+1}]$	[1,1, ..., 1, 0]
2	$[t_{2,1}, t_{2,2}, ..., t_{2,n_2}, t_{2,n_2+1}]$	[1,1, ..., 1, 0]
...	...	...
k	$[t_{k,1}, t_{k,2}, ..., t_{k,n_k}, t_{k,n_k+1}]$	[1,1, ..., 1, 0]

In order to model the $k-th$ customer inter-arrival time t_k in a sequential framework, we compress all the arrival times $t_{k,j}$ of each customer into a new sequential vector $\mathbf{t}_k =$

$[t_{k,1}, t_{k,2}, , t_{k,n_k}]$ for each customer $k \in K$, where each arrival time $t_{k,j}$ is assumed to be exponentially distributed. As expected, each sequence has a different length depending how many purchases customers have made in the company over their lifetime, thus $length(\mathbf{t}_k) = n_k$, where n_k is the total number of purchases made by customer k. Then, to consider right censored event times, which are event times that are only partially observed due the fact that customers may make their next purchase after the time of the analysis, we concatenate the time since each customer was last observed with respect to the time of the analysis t_{k,n_k+1} at the end of the sequence, and create a binary identifier $\delta_{k,j}$ for each event time in the sequence, which take the values $\delta_{k,j} = 1$ when the event is fully observed and $\delta_{k,j} = 0$ for censored or partially observed times, by construction, the sequence δ_k for each customer k will only contain one single censored event at the last position of the sequence. Then, the training data T_X is defined as the union set of all independent $\mathbf{t}_k$ vectors, i.e., $T_X = \bigcup_{k \in K}(\mathbf{t}_k \cup t_{k,n_k+1})$ as shown in Table 4.2, with their corresponding $\delta_{k,j}$ identifiers. In practice, these vectors $\mathbf{t}_k$ can be padded at both, training and serving to an arbitrary length vector.

3.3.2 Training process

The training goal is developing a neural network based in machine learning assuming that the arrival times t_k are exponentially distributed with some specific parameter λ_k for each customer $k \in K$. To achieve this, we implement a Recurrent Neural Network followed by a Multilayer Perceptron with a single output unit with sigmoid activation to estimate $\hat{\lambda}_k = NN_{T_X}(\mathbf{t}_k)$, which is then used to parameterise the exponential density function g of the arrival-times for each customer. At each j-th time-step the input data for the model is the sequence $\mathbf{t}_{k,j} = [t_{k,j-s}, t_{k,j-s+1}, , t_{k,j}]$ with target $t_{k,j+1}$, where s is the sequence-padding parameter which can be set arbitrary for each application, and it is usually lower than $max(n_k)$ and fine-tuned during model training. Figure 3.3 shows the final model architecture proposed, where estimation of model parameters can be performed in two different ways, firstly, by minimising a weighted asymmetric loss function which considers censored events by using the identifier δ_k created for each observation, with $\delta_k = 1$ for fully observed event-times and $\delta_k = 0$ for censored times, at the same time of penalising for large predicted values of $\hat{t}$ for censored observations:

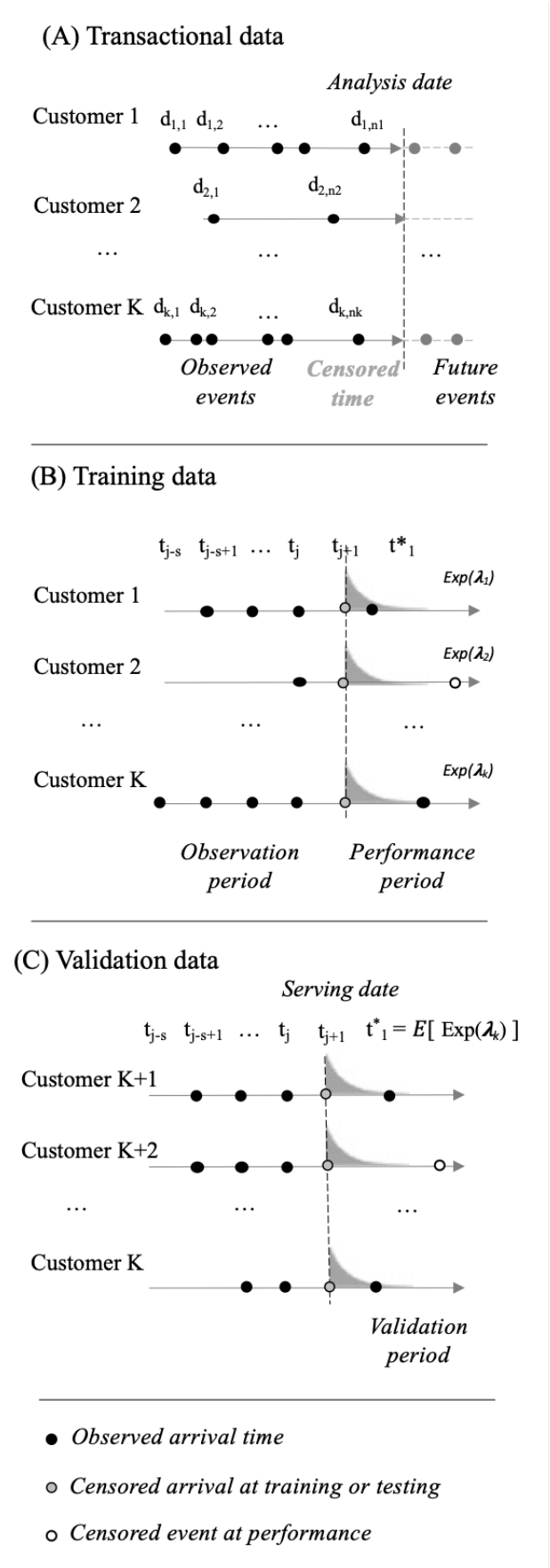

Figure 3.2: Data transformation required to obtain the exponentially distributed $t_{k,j} = d_{k,j} - d_{k,j-1}$. Each customer sequence is observed up to time j to predict $t_{k,j+1}$.

$$Loss = \left(\frac{1}{K_{\delta=1}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} \omega_{t_k} (\hat{t_{k,j}} - t_{k,j})^2 \cdot 1_{\{\delta_{k,j}=1\}} + \frac{1}{K_{\delta=0}} \sum_{k=1}^{K} \sum_{j=1}^{n_k} (1 - \omega_{t_k}) (\hat{t_{k,j}} - E[t])^2 \cdot 1_{\{\delta_{k,j}=0\}} \right)$$

where the weighting factor $\omega_{t_k} = P(\delta_k = 1 \mid T_k = t)$ represents the probability of fully observed events at time t_k in the training data for customer k, and $E[t]$ is the estimated expected value of the target distribution of T which can be obtained by estimating the parameter of the exponential distribution under the presence of censored events via maximum log-likelihood in each training batch as

$$\hat{\lambda} = \frac{K_{\delta=1}}{K \cdot \bar{t}},$$

where $\bar{t}$ is the mean of observed and censored event times [Kalbfleisch and Prentice, 2002].

Alternatively to the aforementioned asymmetric loss function, as it is commonly seen in similar methods mentioned in section 3.1, the model can be also trained by maximising the partial likelihood function of the model for censored data given by

$$L = \prod_{k=1}^{K} \prod_{j=1}^{n_k} f(t_{k,j} \mid \hat{\lambda}_k)^{\delta_{k,j}} \cdot S(t_{k,j} \mid \hat{\lambda}_k)^{1-\delta_{k,j}}$$

which is commonly transformed for training into a minimisation task where the objective is minimising the negative log-likelihood for censored events denoted as

$$LL = -\sum_{k=1}^{K} \sum_{j=1}^{n_k} \left(log(f(t_{k,j} \mid \hat{\lambda}_k)) 1_{\{\delta_{k,j}=1\}} + log(S(t_{k,j} \mid \hat{\lambda}_k)) 1_{\{\delta_{k,j}=0\}} \right)$$

Although the model can be trained with two different methods, by using an asymmetric loss function or maximum likelihood as loss function. For our experiments presented in section 3.5, we decided to train the model by using the negative log-likelihood as loss function, which takes into account that the distribution of errors is not symmetric.

Finally, at serving phase, the survival probability at time t for a customer k can be easily estimated once $\hat{\lambda}_k$ is known, as

$$\begin{aligned} S_k(t) &= 1 - P(T_k < t) \\ &= 1 - \int_0^t \lambda_k exp(-\lambda_k x) dx \\ &= exp(-\lambda_k t) \end{aligned}$$

Then, we define the estimator of $S_k(t)$ at customer level by

$$\hat{S}_k(t) = exp(-\hat{\lambda}_k t)$$

$$S_k(t) = P(T_k > t \mid \lambda_k)$$

Furthermore, to compute the survival status of each customer at an specific serving date we evaluate $\hat{S}_k(t)$ at the current recency time of each customer, i.e, the time elapsed between last customer purchase and the serving date. Additionally, it is also possible to obtain a future deferred event probability over a period simply by computing $\hat{S}_k(t_1) - \hat{S}_k(t_2)$, which is the probability that the next customer purchase will happen between the time interval (t_1, t_2). A pseudo-code description of training and prediction steps is sketched in Appendix A.

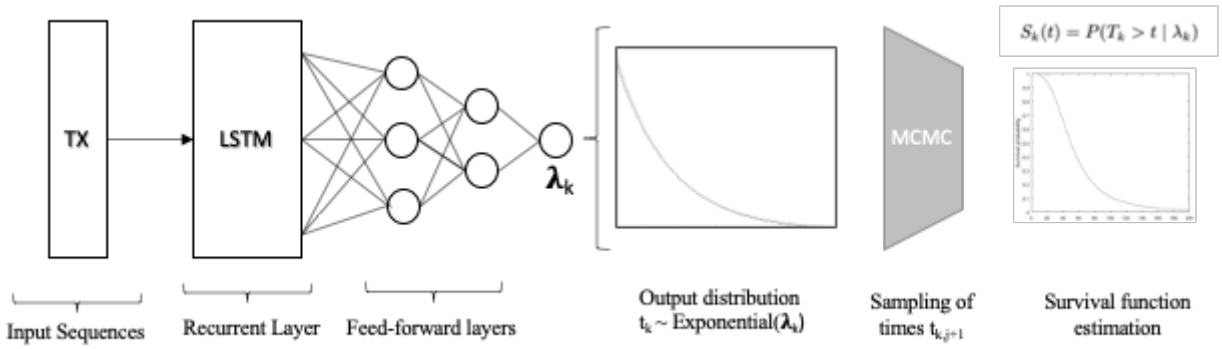

Figure 3.3: Proposed Neural Survival model. The input transactional sequences are passed through a Long-Short Term Memory cell (LSTM) and a Multilayer Perceptron followed by an exponential activation of size 1 to parameterise the customer lever exponential model $\mathbf{t}_k$. The survival model S_k at customer level is drawn from the posterior distribution with Hamilton sampling.

3.4 Advantages and Limitations of the Method

The model architecture presented in figure 3.3, uses a Recurrent Neural Network architecture to estimate the parameter of a survival distribution, this architecture is particularly well-suited to predict customer churn in non-contractual settings due to its ability to process sequential data, handle variable-length sequences, and learn features automatically:

Sequential Data Processing: In non-contractual settings, customer purchases occur at irregular intervals, and the time between these transactions may provide valuable information

about a customer's likelihood of churning. RNN architectures can process data sequences effectively by maintaining a hidden state that captures the temporal dependencies within the data. Leading into capturing information about the arrival times, or purchases for every individual.

Handling Variable-Length Sequences: In most applications, customers have different transaction histories, resulting in variable-length sequences of purchase data. RNN's can naturally accommodate variable-length input sequences without requiring any additional data pre-processing or padding. This flexibility enables the proposed method to learn from the individual transaction patterns of each customer, leading to more accurate and personalized churn predictions.

Capturing Long-Term Dependencies: Customer churn prediction may be influenced by long-term dependencies in the factors. RNN architectures can capture long-term dependencies in data [Bengio et al., 2003], allowing the model to better predict churn based on historical transaction patterns.

Feature Learning: As mentioned in section 3.2, most machine learning methods to predict customer churn often require extensive feature engineering to extract information from raw data to be processed by the algorithm. In contrast, RNNs automatically learn a data representation from the input data during the training process [Hochreiter and Schmidhuber, 1997, Szegedy et al., 2014]. This not only the time required for making churn predictions but allows the model to discover hidden customer behaviours that may not be apparent through manual feature engineering.

However, while the proposed methodology presented in this section using Recurrent Neural Network offers several advantages for customer churn prediction in non-contractual settings, there are also some caveats to consider if this model is applied in commercial settings, including model complexity in compared to approaches presented in section 3.2, computational resource requirements, hyperparameter tuning challenges, and sensitivity to data quality:

Model Complexity: Architecture shown in figure 3.3, involves a relatively complex model to estimate the parameters of a survival distribution for every individual, which may be difficult to implement in many applications.

Hyperparameter Tuning: Similar to many other machine learning techniques, the performance of the model strongly depends on the choice of hyperparameters, such as the number

of hidden layers, learning rate, and the choice of the survival function. Finding the optimal set of hyperparameters can be a time-consuming process, involving trial and error or grid search techniques. Furthermore, inadequate hyperparameter tuning may result in suboptimal model performance or overfitting.

Sensitivity to Data Quality: This technique may be sensitive to the quality of the input data, its sequential dependencies, and the amount of data available. Thus, it is essential to ensure that the input data is clean and reliable before using this method for customer churn prediction.

3.5 Experiments and Results

This work aims to improve classification power at making customer churn prediction for a large retail company in the UK. Due to confidentiality constrains, name of this company will not be shown, and it will be denoted as $company$. Additionally, model performance is also assessed in a synthetic dataset which resembles the main characteristics of transactional data in the retail industry. To compare our methodology, we established two baseline models, an initial transactional only CPH model, and an individual-level Kaplan-Meier estimator. Although CPH allows to include customer level characteristics as model covariates, such as age, gender, and type of customer, with the purpose of having a fair comparison in model performance at assessing transactional-only input data, the CPH baseline used in this work is based only on the time elapsed between consecutive events, ignoring all additional customer-level information available for individuals. Similarly to the process presented previously, a censored event time obtained from the distance between customer last purchase and analysis dates is assumed for each customer.

As mentioned, CPH does not model directly the event time, but the hazard function of individuals at time t, which is the probability of individuals experiencing the event of interest at time t, and once the hazard function is known the survival function of individuals can be estimated as $\hat{S}(t) = \exp(-\Gamma(t))$, where $\Gamma(t)$ is the cumulative hazard function.

Additionally, we compare model performance against an individual Kaplan-Meier baseline, for this, we estimated the survival function $\hat{S}_k$ for each customer by considering all the observed and censored event times for customer k, and made predictions of its current survival status with respect to its current censoring time.

3.5.1 Dataset 1: *Retail data*

The first dataset contains transactional data of $company$, a large retailer in the UK with almost 200 physical stores and online ordering. Between 29/03/2016 and 20/04/2021, the $company$ had a total of 17.8 million transactions made by 2.8 million customers, with an average time between observed events of 80 days for customers with at least 2 purchases, i.e., the estimated λ parameter for the time-to-event exponential model considering all customers is $\hat{\lambda} = 0.012$. Although some demographic information about individual customers is available, such as type of customer and location, in order to model customer churn as described in section 3.3.1, we only include the customer id and the purchase date in our analysis to obtain times between events. Due to the non-contractual setting of the retail industry, this dataset presents an estimated monthly customer drop-out of 14.7% after 1400 days, i.e., 14.7% of customers will not make any further purchase in the company. Figure 3.4 shows the cumulative logarithmic survival probability of this dataset for a 12 different months sliding-window.

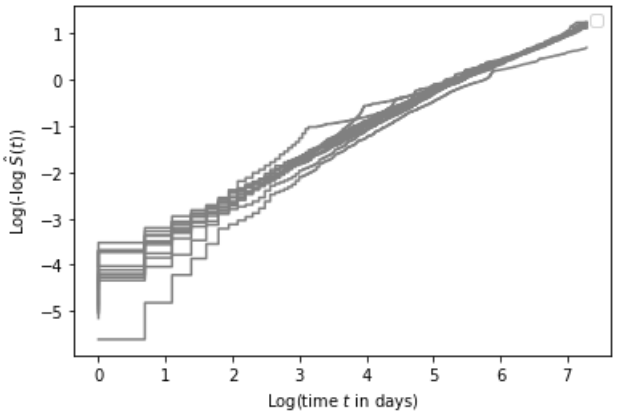

Figure 3.4: Estimated survival function of retail dataset via Kaplan-Meier curve for 12 monthly cohorts of customers who made a purchase in the last 30 days.

Finally, in model training we consider all transactions made by a random sample of 100,000 customers.

3.5.2 Dataset 2: *Synthetic data*

Following [Bender et al., 2005] a synthetic dataset of realistic multi-event survival data with known λ_k for each individual can be obtained. To obtain a large enough dataset, we simulate a set of 100,000 customers where λ_k for each customer is drawn from a Gaussian distribution

with mean $\mu = 0.08$, and standard deviation $\sigma = 0.02$, i.e., the expected customer return time for this synthetic dataset is every 12.5 days. To avoid negative values of λ_k we specify a minimum threshold of 0.01, thus, $\lambda_k = max(0.01, N(\mu, \sigma^2))$. Then, to introduce the effect of customers not making any further purchase due to the inherent non-contractual setting present in the retail industry, a stopping probability of 15% is introduced for every customer at each sampling iteration of $t_{k,j} \sim exp(\lambda_k)$. For this dataset in which the estimated monthly customer drop-out is 8.7% after 120 days. Figure 3.5 shows the cumulative logarithmic survival probability of this dataset for a 6 different months sliding-window. And Table 3.3 presents a list of main summary statistics for both experimental datasets, including dataset size, frequency of the events in the data, training and performance periods, and testing split size.

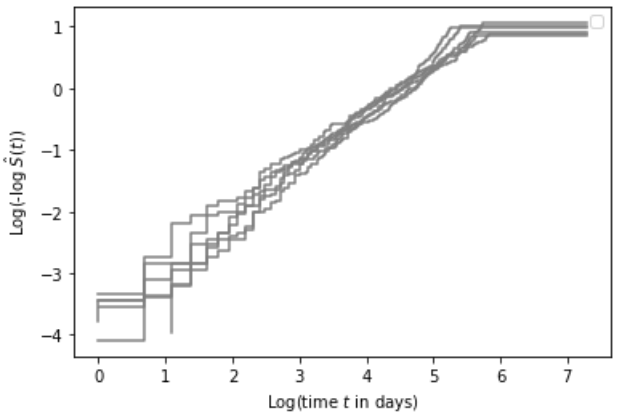

Figure 3.5: Estimated survival function of synthetic dataset via Kaplan-Meier curve for 12 monthly cohorts of customers who made a purchase in the last 30 days.

3.5.3 Results

As stated previously, the time between purchases is exponentially distributed for both datasets analysed. In all cases, the probability of a customer making its next purchase decreases significantly as time passes, and just few transactions have happened after a period of 100 days after the last purchase. Table 3.3 presents a list of summary statistics for each datasets.

The data pre-processing for all datasets is carried as stated in section 3.3.1. For each dataset, we compress the transactional data prior to an arbitrary analysis date into customer level sequential representations to obtain a dataset with a similar form than in table 4.2, i.e.,

for each dataset we create the training data as $T_X = \bigcup_K [t_{k,n_k-s}, t_{k,n_k-s+1}, ..., t_{k,n_k} \cup t_{k,n'_k}]$ for every $k \in K$, where s is the sequence-padding parameter used to consider all sequences to be of the same length at training, these padding parameters are shown in table 3.3 for each dataset. As it is commonly done in modelling sequential data, the analysis date for training should be chosen in a way that the performance period excludes completely the first date available for serving. In our experiments, analysis dates for all dataset are set to at least 6 months before the model serving date, besides we specify a validation set of customers completely disjoint to the customers used at training for each model. Figure 3.2 shows a visual representation of the overall data processing needed to create the the vectors $\mathbf{t}_k$ for each customer.

Table 3.3: Main summary statistics for both experimental datasets.

	Retail Dataset	Synthetic Dataset
Number of customers	2.8 M	100 K
Training size	10%	80%
Validation size	5%	20%
Observation period (MM/YYYY)	03/2016 - 05/2020	N/A
Performance period (MM/YYYY)	06/2020 - 04/2021	N/A
Median customer purchases	2	12
Median time between purchases	32 days	12.5 days
Length of padded sequences (s)	5	7

At prediction time, we create the input sequences $[t_{k,n_k-s}, t_{k,n_k-s+1}, ..., t_{k,n_k}, t_{k,n_k+1}]$ as described in section 3.3.1 to estimate $\hat{\lambda}_k$ for each customer $k \in K$ in the performance dataset, which is set to be the complement of the training dataset for both experiments. Then, the survival probabilities $S_k(T > t)$ for any time t can be estimated by integrating the exponential model parameterised by $\hat{\lambda}_k$ for each customer. Figure 3.11 shows the estimated survival curves obtained for individual customers belonging to four different segments with respect to their frequency of purchase at the time of the analysis. Figure 3.6 shows the overall estimation of the survival function for customers with high and low purchase frequency in the corresponding validation datasets using all three different methods to estimate survival probabilities. Additionally, Table 4.4 shows the performance results obtained in both datasets of the metrics mentioned previously in section 2.3.2, as shown, the recurrent model can

outperform both baseline methods in terms of the brier score, however, both the C-Index and the time-dependant AUC are significantly higher for the Cox model.

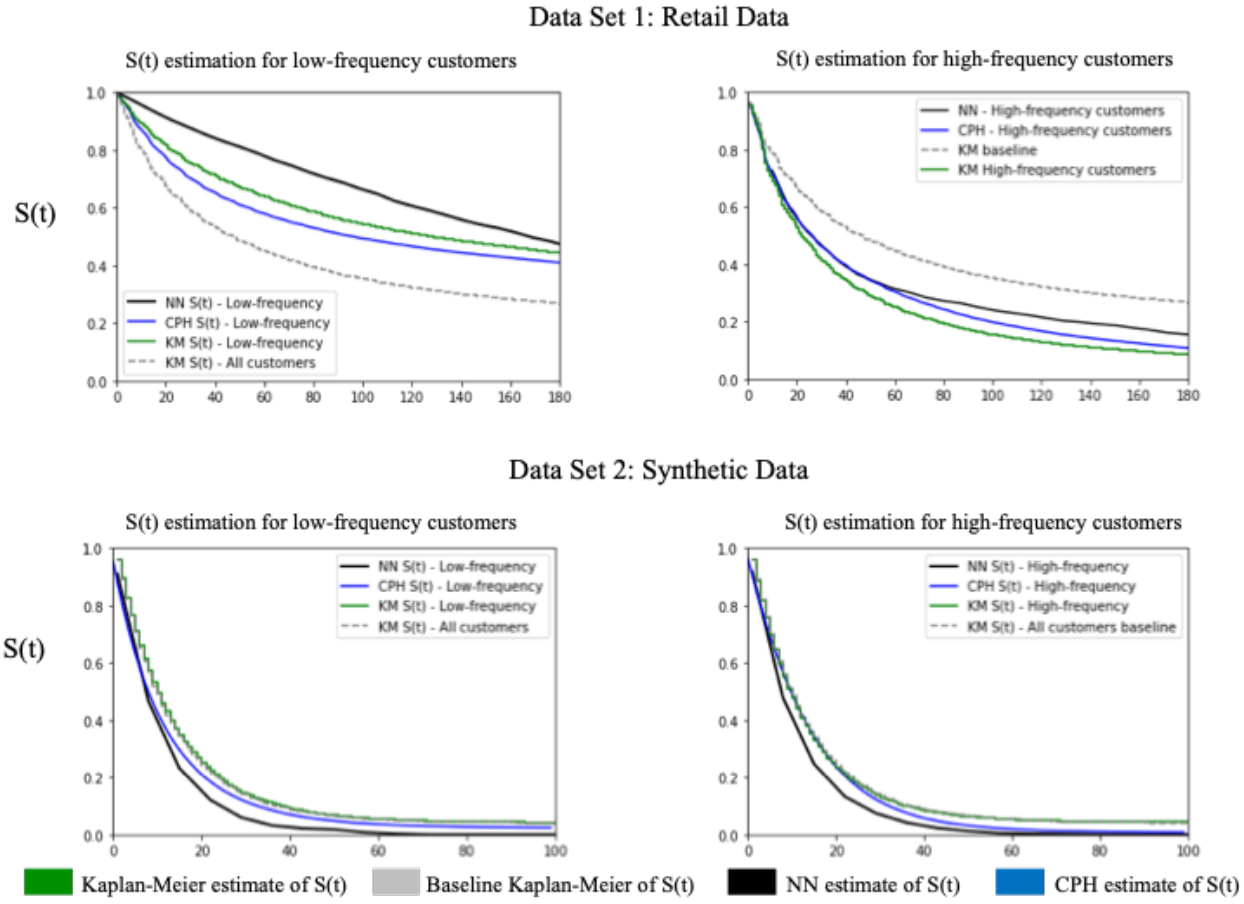

Figure 3.6: Estimated function $S(t)$ for an average customer in both validation datasets with respect to low and high customer purchase frequency.

As expected, it is seen from Figure 3.6 and Figure 3.7 that the Recurrent Neural Network model can learn somehow efficiently a survival distribution of event times close to the Kaplan-Meier estimate of $S(t)$, although due to the exponentially distributed assumption under the event times distribution, the overall survival function $S(t)$ estimated with the NN will never match perfectly the Kaplan-Meier estimated survival function. As the synthetic dataset was designed in a way that frequency of events and recency (time since last event) do not affect customer's churn rate, it is expected to not see a large impact in the estimated survival curves for customers with different frequency of purchase, as it is shown in Figure 3.6 for synthetic dataset.

Additionally, we carried an analysis with the retail dataset to analyse how our method using neural network to estimate survival probabilities captures changes in the frequency of

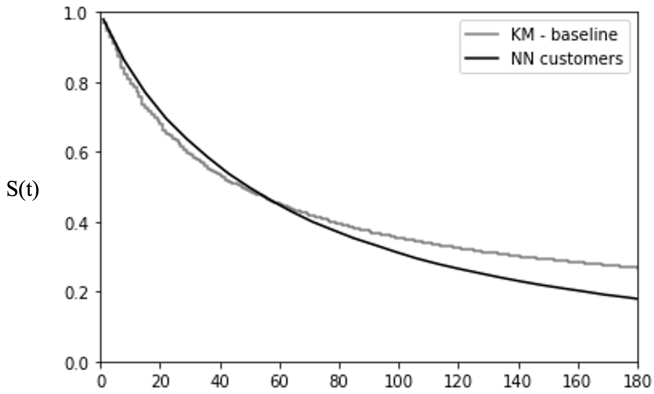

Figure 3.7: Point-wise average of estimated survival functions $S(t)$ at customer level by proposed neural network and baseline Kaplan-Meier estimator.

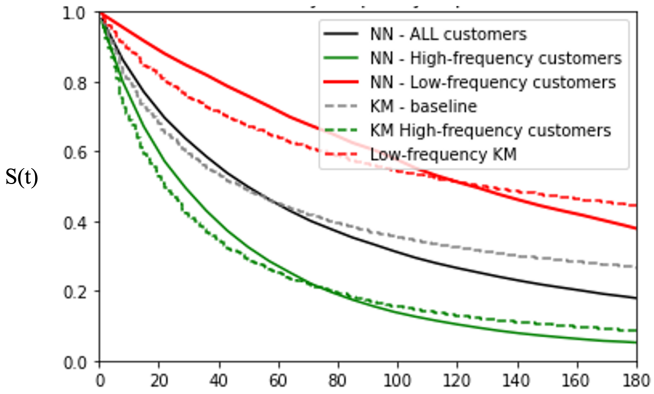

Figure 3.8: Point-wise average of estimated survival functions $S(t)$ at customer level by proposed neural network and baseline Kaplan-Meier estimator with respect to high and low frequency of purchase.

purchases for single customers. For this, we analyse the slope of a linear regression obtained from the event times of each single customer, and check whether this slope is high, indicating that the time to events increase and the frequency of purchase decreases, low, indicating that the time to events decreases and the frequency of purchase increases, and constant, indicating that there is no change in frequency of purchase with the sequence. Figure 3.9 shows the estimated lambda distribution for these three different groups of customers, and Figure 3.10 shows the estimated survival distribution $S(t)$ for customers belonging to these three groups of customers.

Finally, we evaluate models' performance in terms of accuracy, sensitivity and specificity by assessing models as binary classification tasks, where the goal is using the estimated survival probability of customers to predict if the event occurs before time t. For this, we compute the median survival probability of the event at time t, denoted by $m(t)$ and use a

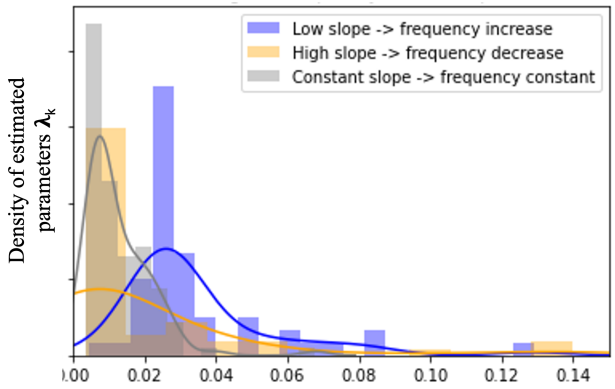

Figure 3.9: Distribution of estimated parameters λ_k in retail dataset by customers with observed change in frequency of purchase.

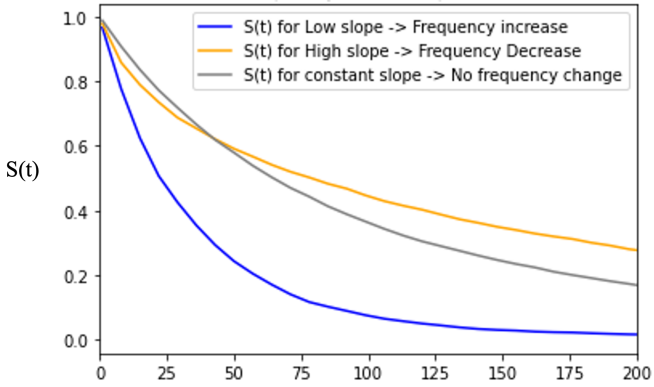

Figure 3.10: Point-wise average of estimated survival function of retail dataset by customers with observed change in frequency of purchase.

classification rule that predicts a positive event for observations with $S_k(t) < m(t)$, i.e., the event might happen sooner than later for customer k, and a negative prediction otherwise. Table 3.5 shows results obtained for this evaluation.

3.6 Comparison against hand-crafted feature-based techniques

The rapid evolution of the retail industry in customer-driven marketing, has led to a growing interest in accurately predicting customer churn driven by technological advancements and increased focus on analytics-based and machine learning approaches, particularly in those that make use of hand-crafted features in their construction. This section aims to compare the performance of the proposed methodology in section 3.3, which combines recurrent neural networks and survival analysis to predict the next purchase time of customers, against methodologies widely-used in industry to predict customer churn, such as Cox proportional

Table 3.4: Model performance obtained in the experiments carried for both datasets in testing and validation splits. Large C-index, large time-dependant AUC, and small Brier score indicate high model performance.

	Retail data		Synthetic data	
	Test	Validation	Test	Validation
	Brier Score at 180 and 100 days			
RNN	0.303	0.369	0.350	0.431
CPH	**0.070**	**0.197**	**0.021**	**0.186**
Ind - KM	0.384	0.437	0.378	0.292
	C-Index			
RNN	0.391	0.545	0.500	**0.498**
CPH	**0.580**	0.233	**0.675**	0.421
Ind - KM	0.555	**0.574**	0.495	0.465
	Time-dependant AUC			
RNN	0.348	0.458	0.487	**0.493**
CPH	0.259	0.307	0.364	0.482
Ind - KM	**0.591**	**0.590**	**0.494**	0.468

hazard (CPH) and survival trees with hand-crafted features, particularly using RMF-based attributes (Recency, Frequency, and Monetary value), as these are commonly available in the marketing industry.

Cox Proportional Hazard Model with Hand-Crafted Features

As mentioned in section 2.3, the Cox Proportional Hazard (CPH) and survival trees models are popular techniques in the marketing industry to analyse survival data. CPH assumes that the hazard function for each individual is proportional to a baseline hazard function, with the proportionality constant determined by a linear combination of the individual's covariates. In the context of churn prediction, CPH models can be applied by adding hand-crafted and customised features that capture various aspects of customer behaviour for the application

Table 3.5: Model performance obtained in the experiments carried for both datasets considering using the expected survival function to predict whether the event will happen before or after a time t, $t = 180$ days and $t = 100$ days for retail and synthetic data respectively. As a binary classification problem, larger values of accuracy, sensitivity and specificity indicate higher model prediction capabilities.

	Retail data		Synthetic data	
	Test	Validation	Test	Validation
		Accuracy		
RNN	**0.654**	0.525	0.560	0.210
CPH	0.485	0.462	0.366	**0.511**
Ind - KM	0.592	**0.646**	**0.712**	0.322
		Sensitivity		
RNN	0.659	0.457	**0.883**	**0.833**
CPH	0.539	0.541	0.491	0.427
Ind - KM	**0.664**	**0.616**	0.780	0.769
		Specificity		
RNN	0.325	0.683	0.147	0.115
CPH	0.419	0.436	0.146	**0.623**
Ind - KM	**0.560**	**0.718**	**0.258**	0.227

domain of interest, such as recency, frequency, and monetary value of transactions, customer demographics, engagement level metrics, or variations of these attributes.

Survival trees with Hand-Crafted Features

Survival trees [LeBlanc and Crowley, 1993] use the Kaplan-Meier estimator to identify the best splitting point for each feature that maximizes the separation between the resulting subgroups in terms of their survival patterns. For each candidate splitting point, the survival tree computes during training the KM estimator for the two resulting subgroups and evaluates the difference between their survival functions. The optimal splitting point for a given feature is the one that maximizes the separation between the subgroups with respect to their

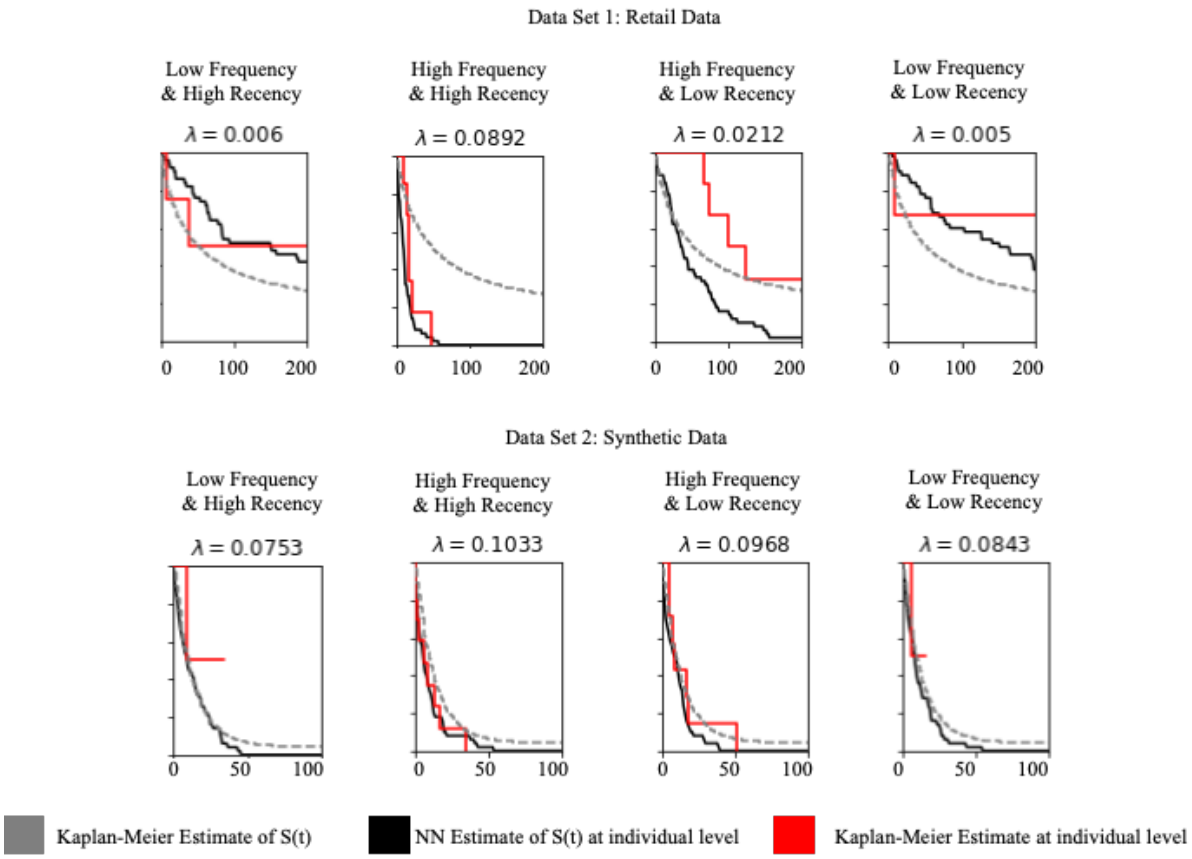

Figure 3.11: Estimated survival function at customer level up to 200 and 100 days for customers belonging to different groups in the dataset with respect to their frequency of purchase and current recency at the time of the analysis.

estimated survival probability. Once the survival tree has been constructed, the survival tree can be used to make prediction by first, determining the terminal node to which the individual belongs based their input features. Then, the Kaplan-Meier estimator is used to compute the survival function for this terminal node, which is obtained by averaging the survival functions of all individuals within the node.

To compare the model performance of the recurrent model presented in section 3.3 against hand-crafted based models, a CPH and a survival tree with hand-crafted attributes were trained to estimate the survival probability of individuals for both datasets presented in sections 3.5.1 and 3.5.2. Table 3.6 describes the attributes used during model training, which include slight variations of RFM features, as well as features that may indicate the distribution of time elapsed between purchases for individuals, these features may provide valuable insights into the purchasing patterns and engagement levels of customers, which are crucial factors in determining the likelihood of churn.

Table 3.6: Hand-crafted attributes used to train CPH and survival tree baselines.

Attribute	Description
Recency	Time since last customer purchase
Time observed	Time since fist customer purchase.
Number of purchases	Total number of purchases made by the customer.
Minimum event time	Minimum time elapsed between purchases.
Maximum event time	Maximum time elapsed between purchases.
Average event time	Average time elapsed between purchases.

Recency, or the time since the last customer purchase, is a commonly used feature as it usually reflects the current engagement level of the customer with the business. In non-contractual settings, typically customers that have not made a recent purchase are typically more likely to churn in compared to those with recent transactions. Time observed, or the time since the first customer purchase, provides information about the customer's relationship duration with the business, which may be indicative of individual's loyalty and engagement levels with the business. Number of purchases captures the overall transactional activity of the customer, where a higher number of purchases may generally suggest a stronger engagement with the business and lower churn risk. Minimum event time, maximum event time, and average event time are features that describe the distribution of time elapsed between purchases. These features may help to identify patterns in customer behaviours, such as regularity or seasonality in their purchasing habits.

While these features offer valuable insights for predicting customer churn, additional features could also be considered to enhance the CPH model's performance. For instance, demographic and psychographic variables, such as age, gender, and preferences, might also help identify specific customer segments with varying churn risks. However, these attributes are not included during model training to perform a more realistic comparison against the recurrent approach with only has access to the time between purchase events. Table 3.7 shows the results in terms of model performance for the recurrent method against a CPH model and a survival tree using hand-crafted features. Figure 3.12 shows the estimation of the survival functions for these methods with respect to the baseline Kaplan-Meier estimator, showing that for the two datasets included in this work, the survival function generated by the survival tree method is closer to the baseline Kaplan-Meier.

Table 3.7: Model performance obtained in experiments for both datasets using hand-crafted features in the training of a Cox proportional hazard model and a survival tree.

	Dataset 1: Retail datataset					
	Brier Score	C-index	Time-dependent AUC	Accuracy	Sensitivity	Specificity
RNN	0.369	0.545	0.493	0.525	0.457	0.683
CPH with features	0.080	0.828	0.883	0.839	0.607	0.896
Survival Tree	0.248	0.579	0.707	0.337	0.809	0.222

	Dataset 2: Synthetic datataset					
	Brier Score	C-index	Time-dependent AUC	Accuracy	Sensitivity	Specificity
RNN	0.369	0.545	0.493	0.525	0.457	0.683
CPH with features	0.115	0.766	0.807	0.528	0.202	0.986
Survival Tree	0.278	0.640	0.740	0.658	0.841	0.402

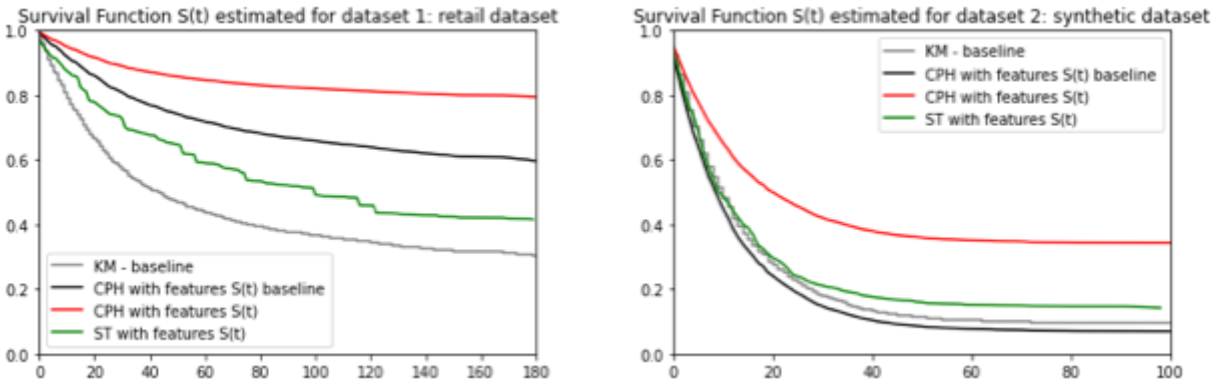

Figure 3.12: Estimated survival functions in the validation set for methods including hand-crafted features with respect to the Kaplan-Meier estimator baseline.

Naturally, the choice of the most appropriate model for predicting customer churn depends on the specific needs and constraints of the business, the level of interpretability desired, the computational capacity available in production environments. While each technique offers unique advantages and limitations, as shown in table 3.8, and considerable differences in performance, as results of table 3.7 suggest, the data available plays a crucial role in determining the most appropriate choice.

If the dataset contains strong temporal dependencies or sequences of customer interactions, or no additional data about individuals is provided but the purchase times, the combined RNN-survival model approach be a good approach to predict customer churn by

Table 3.8: Advantages and disadvantages of hand-crafted feature-based approaches at predicting customer churn.

Benefits of Recurrent Approach	Benefits of CPH Model	Benefits of Survival Trees
Modeling Temporal Dependencies: RNNs are particularly useful to capture complex temporal patterns in data, as their hidden state stores information from previous time steps. This ability allows them to identify patterns that may be missed by other survival models, leading to more accurate event-time predictions.	**Flexibility**: CPH models allow inclusion of multiple explanatory features or covariates, this flexibility enables the model to capture complex relationships between features without making strong assumptions on the hazard function.	**Non-parametric approach:** Survival trees do not make assumptions on the functional form of the baseline hazard function, making them more robust to deviations from the assumed model.
Automatic Feature Learning: RNN can automatically learn a data representation for event-times without the need of manual feature engineering. This can potentially uncover hidden patterns in data that might be overlooked in a feature engineering process.	**Interpretability:** CPH models provide interpretable frameworks for understanding the relationship between features and the hazard function, allowing for meaningful insights into the factors driving customer churn.	**Ability to handle non-linear relationships:** Survival trees can capture non-linear relationships between features and the time-to-event distribution, which may be difficult to model using other techniques.
Scalability: RNNs can be trained on large datasets which are increasingly common in today's business environments. Furthermore, these techniques can be deployes with minimal human supervision.	**Proven performance:** CPH model are popular choices in multple industries for predicting time-to-event outcomes, and in many cases they outperform other more comeplex ML methods.	**Interpretability:** The structure of survival trees allows for easy interpretation of the relationships between features and the time-to-event outcome. This can be particularly valuable in situations where transparency is a key requirement for decision-making or regulatory compliance.

Caveats of Recurrent Approach	Caveats of CPH Model	Caveats of Survival Trees
Complexity: The combination of RNNs and survival models can result in complex models that may be unecessary difficult for many application domains. Furthermore, they typically lack of transparency and interpretability.	**Feature engineering:** The CPH model relies on hand-crafted features to capture various aspects of customer behavior, such as RFM attributes. These features require domain knowledge and manual engineering and may not always capture all relevant information.	**Instability:** Due to their training process to create partitions of data, survival trees can be sensitive to small changes in the data, leading to different tree structures and predictions.
Overfitting: RNNs, like other deep learning models, are prone to overfitting when dealing with high-dimensional data, which will require of regularisation techniques to control the complexity of the model.	**Limited capacity to model non-linear relationships:** CPH models may not be able to always capture complex non-linear relationships between features and the time-to-event outcome. In such cases, more advanced ML techniques or non-linear transformations of the input features may be required.	**Overfitting:** Like other tree-based models, survival trees are prone to overfitting, especially when dealing with high-dimensional data. In which case, regularisation technqies such as pruning, cross-validation are required.

leveraging RNNs' ability to capture intricate temporal patterns without the need to compute hand-crafted features. Alternatively, if the dataset primarily consists of static features, such

as demographic information or aggregated behavioural metrics, the CPH model or survival trees might be more suitable choice. The CPH model is particularly useful when interpretability of results is a key requirement, as it provides a transparent framework for understanding the relationship between each feature and the hazard function. Survival trees, in contrast, can handle non-linear relationships in data and automatically select important features during training but may suffer from instability and overfitting issues.

3.7 Conclusion

Companies around the world are interested in knowing which customers are likely to churn in order to make proactive retention efforts in keeping them engaged with the brand and incentive interaction between customers and products. By predicting the probability of customers making their next purchase over time, our model is capable of estimating the individual-level survival function for each customer instead of an overall survival model for the entire population.

Using recurrent neural networks in time-to-event modelling to predict customer churn allows to model customer purchasing behaviour entirely from transactional data, leaving aside all customer level characteristics, such as age, gender, and income, which are commonly used by companies to estimate how likely is a customer to engage with the brand, and therefore, to purchase again. Additionally, by modelling the time-to-purchase as a sequential problem with a recurrent network architecture, such as the LSTM, the model can learn dependencies in historical interactions to match or improve churn prediction performance of well-established survival techniques with a minimum effort in performing a feature engineering phase or obtaining expensive hand-crafted characteristics from the input data, particularly in cases where the data may be subject to temporal dependencies or seasonality. Furthermore, treating item purchases and customer churn as sequential information enable more effective targeting of marketing efforts, allowing businesses to tailor their strategies based on individual customer behavior patterns.

However, our approach also has its limitations, firstly, assuming an exponential distribution over the event times for every customer can potentially lead into an underestimation of the survival probability remaining at time t for some the most loyal segments of customers, in which the probability of the next purchase should remain high even after long observation

periods without purchases. Secondly, our method requires a large number of purchases made by each customer to provide reliable predictions and the model might be less accurate in customer with short purchasing history, as seen in the experimental results, the model achieved better performance for the synthetic dataset, in which the frequency of purchases is considerably larger than in the retail data. Therefore, this method is more suitable for retail companies where the frequency of customer purchases is high. Nevertheless, our methods is capable to provide estimation of customer churn status and survival probability at individual level for customers with only few event times, which is not possible or not using other methods such as individual Kaplan-Meier, or not accurate in methods such as CPH.

Future work can explore the generalization of this method by combining multivariate time series from different signals as input data for the LSTM layers in the model, as well as compressing seasonal information of purchases or incorporating context information about the purchased items, which is most often easy available in the retail industry.

CHAPTER 4

Sequence-aware item recommendations for multiply repeated user-item interactions

4.1 Introduction

Modern recommender systems deployed in production environments rely significantly on the use of matrix-completion techniques combined with users and item characteristics to detect long-term user preferences. However, most of the time these techniques do not perform well in settings where user preferences can change significantly over time, or when users interact repeatedly with the same set of items. Sequence-aware recommender systems typically consider user-item interactions in sequential frameworks to detect drifts in user preferences over short periods of time, and to identify short-term popularity trends quickly and efficiently [Quadrana et al., 2018]. Typically, the input for these systems is an ordered and timestamped list of past user actions and the output is an ordered list of items most likely to be relevant for the user, just as in the traditional item recommendation setup introduced in section 2.4.3.

Over the last few years, there has been a vast amount of research around sequence-aware recommender systems [Guy Shani, 2005, Wu et al., 2017, Quadrana et al., 2018, Zhang et al., 2019, De Souza Pereira Moreira et al., 2021] particularly in the context of implicit feedback, which is when the exact user rating of items is unknown for all or most user-item interactions in the data. As aforementioned in section 2.4.5, sequence-aware recommender systems offer several benefits against matrix completion techniques, including improved recommendation

performance for multiple applications, adaptability to adapt to changing trends and user behaviours over time, context-awareness by incorporating additional contextual information, such as seasonal trends, promotional events, or time of day, cold-start mitigation for new users by identifying similar patterns in the sequences of existing users, and real-time responsiveness by processing variable-length sequences for each individual.

Inspired in the approach that NLP applications takes to process and analyse sequences of tokens or words, this chapter suggest a new approach to process sequences of user-item interactions to make predictions of future products that users might be likely to buy in future, with three main goals. Firstly, being able to predict efficiently repeated interactions of users with items, a behaviour that is commonly seen in industries such as retail, where customers interact with the a single product multiple times over time, but not considered in matrix-completion recommender systems [Quadrana et al., 2018]. Secondly, achieve comparable recommendation prediction performance to matrix-completion techniques, particularly in the top of the recommendation list. Finally, make recommendations faster and capture dynamic user preferences, which is achieved by leveraging the sequential nature of user-item interactions and is a crucial factor in domains where customer preferences change rapidly over time.

4.2 Methodology

This section describes the mathematical formulation of modelling user-item interactions over time and the architecture used to estimate how likely users are to interact with a specific set of items $I' \in I$ over a defined horizon. Sequential modelling techniques are such as Markov Decision Processes (MDP), Latent Dirichlet Allocation (LDA), and Recurrent Neural Networks (RNNs) are useful approaches for tasks where the data to be analysed has a temporal dependency or a inherently sequential form, like in natural language process and understanding tasks such as sentiment analysis, text classification, and next token prediction. Although the problem of performing item recommendations for users has typically not been seen as a sequential task, there are several sequential techniques that can be implemented to achieve better recommendation performance. In this work, we aim to process user transactions in a sequential framework using inspiration and data pre-processing techniques from the field of NLP, such as tokenisation and sequential embeddings, in a recurrent neural network to make

probability estimations for each user interacting with specific items in the dataset.

Following notation introduced in section 2.4.1, we can describe the problem of sequential recommendation as follows, let $U = \{u_1, ..., u_{|U|}\}$ be the set of users, and $I = \{i_1, ..., i_{|I|}\}$ the set of items. In contrast to matrix-completion methods presented previously where the goal is estimating the overall user estimated rating $\hat{r}(u_j, i_k)$ for each user $u_j \in U$ and item $i_k \in I$, our goal is finding the probability that the user u_j interacts with the item i_k in a defined horizon, represented as $P_{u_j}(i_k) = P(i_k \mid Seq(u_j))$, where $Seq(u_j)$ is the ordered sequence of items previously bought by the user u_j, and its detailed construction is presented in section 4.2.1.

To estimate the probability $P_{u_j}(i_k)$ we use the recurrent neural network shown in Figure 4.1, which consists of an embedding layer followed by two Long-short Term Memory (LSTM) units and a five-layer feed-forward network with sigmoid activation and output size $\mid I' \mid$ corresponding to all potentially recommendable items.

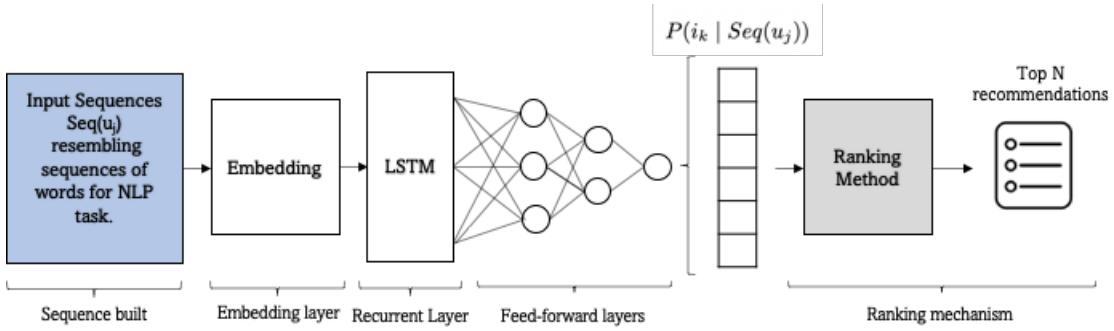

Figure 4.1: Proposed recurrent model to estimate probability of interactions of future tokens. The input transactional sequences $Seq(u_j)$ are processed through an embedding layer and two Long-Short Term Memory cell (LSTM) followed by a Multilayer Perceptron with sigmoid activation of size $\mid I' \mid$ to obtain the probabilities of user-item interaction $P_{u_j}(i_k)$.

4.2.1 Data Representation

Inspired by the approach that NLP techniques take to compress and process sequential data, we aim to process user-items interactions as a sequential task. To achieve this, we take the original transactional data shown in Table 4.1, which is typically used to build the user-item matrix presented in section 2.4.3, and build a sequence of item interactions $Seq(u_j) = [i_{u_j,1}\, i_{u_j,2} \ldots i_{u_j,u_{n_j}}]$ for every user $u_j \in U$ as shown in table 4.2, where the elements of this list are the space-separated and timestamp-ordered item ids that the user u_j has

interacted with. Like this, it is straightforward to process the sequence $Seq(u_j)$ as a textual sequence where each item id would simply resemble a single word in a dictionary for an NLP task, assuming that $t_{u_j,n} \neq t_{u_j,n+1}$ for every user u_j and interaction n, i,e. there are no simultaneous user-item interactions in the data.

Table 4.1: Original Transactional data

User ID	Item ID	timestamp
u_1	i_1	$t_{1,0}$
u_1	i_2	$t_{1,1}$
u_2	i_1	$t_{2,0}$
...	...	...
u_j	i_d	$t_{j,d}$
u_j	i_{d+1}	$t_{j,d+1}$
...	...	...

Table 4.2: Customer sequential-transactions

Customer ID	$Seq(u_j)$
u_1	$[i_{u_1,1}\, i_{u_1,2} \ldots i_{u_1,n_1}]$
u_2	$[i_{u_2,1}\, i_{u_2,2} \ldots i_{u_2,n_2}]$
...	...
u_j	$[i_{u_j,1}\, i_{u_j,2} \ldots i_{u_j,n_k}]$
...	...

Then, the target vector for the model for a user u_j is built as $Y_{u_j} = [y_{u_j,i}, y_{u_j,2}, , y_{u_j,|I'|}]$, where $y_{u_j,k} = 1$, if the user u_j interacted with the item i_k over the performance period, and 0 otherwise.

This method of representing user-item interactions has several advantages over the matrix-completion representation presented in figure 2.7. This approach orders item ids based on timestamps, accounting for the temporal dynamics of user behaviour. By doing so, it enables the model to learn patterns and trends in user preferences over time, leading to more accurate predictions. As each user's sequence is processed independently, this method can efficiently

handle large datasets with numerous users and items without compromising performance. This makes it a practical solution for businesses dealing with increasing amounts of data. Finally, the resulting sequences of user-item interactions are easily interpretable and visualized, allowing for better understanding of user behaviour and preferences. This interpretability can be valuable for both model development and business insights, providing a robust and scalable approach to representing user-item interactions while providing valuable insights into user behaviour as an alternative to the matrix-completion representation.

4.2.2 Tokenisation

Tokenisation is a simple but powerful technique in NLP to transform a sentence of text and split it into the different elements or 'tokens' that compose it. For example, given the textual sentence *'recommendations for different users and items'*, the tokenisation returns a sequence of six tokens *['recommendations' 'for' 'different' 'users' 'and' 'items']*, with each single word as its own token. *Dictionary-based tokenisation* is perhaps the most common type of tokenisation used in the AI industry, this method uses a pre-defined dictionary of mapped words into tokens, typically learnt from a large set of textual sequences, which allows to tokenise every new given sentence to be processed. However, there are different tokenisation methods such as *rule-based tokenisation*, *regular expression tokenisation*, or *sub-word tokenisation* in the NLP literature that are not covered in the scope of this work.

As in this work the tokens to be processed do not correspond to real words in a pre-trained dictionary but to item identifiers, it is necessary to learn a new dictionary from the data during model training and use it to compress the sequences of purchases at prediction phase.

4.2.3 Token Embeddings

As mentioned in section 2.2.2, word embeddings are powerful methods used in natural language processing and information retrieval to overcome the high-dimensionality problem of dealing with large corpus of text by obtaining representations of words contained in documents [Bengio et al., 2003]. Embeddings are essential for transforming discrete tokens into continuous vectors that can be fed into machine learning models. Several types of embeddings can be employed for sequence-aware recommender systems and the choice of each method may have an impact in the final performance of the system. Table 4.3 summarise

multiple benefits and caveats of for three popular embedding methods with the task of encoding item ids instead of typical words, word2vec, GloVe, and BERT embeddings.

Table 4.3: Advantages and disadvantages of word embedding methods to represent items into dense vectors.

Method	Description	Main Advantages	Main Misadvantages
Word2Vec	Word2Vec [Mikolov et al., 2013a] can be applied to user-item interaction sequences by treating item ids as words. This technique learn dense representations of tokens by predicting either the context items given a target item or the target item given its context items.	• Efficient and scalable, making it suitable for large datasets. • Captures semantic relationships between items based on their co-occurrence patterns within user interaction sequences.	• Sensitive to the choice of hyperparameters • Cannot capture subword information, which may be relevant for certain types of item ids or metadata.
GloVe	GloVe [Pennington et al., 2014] can be adapted for user-item interactions. It learns embeddings by factorizing the co-occurrence matrix of items, capturing both global and local context information.	• Combines global and local context information, leading to more comprehensive item embeddings. • Can be more robust to variations in the frequency of item occurrences compared to Word2Vec.	• Requires constructing and storing a large co-occurrence matrix, which can be computationally expensive for large datasets. • Does not account for the order of items within a sequence.
BERT	BERT [Devlin et al., 2018] can be adapted for user-item interaction sequences by pretraining the model on masked item prediction tasks. Fine-tuning the pretrained BERT model on the specific recommendation task can result in powerful dense representations.	• Bidirectional context representation allows for capturing complex item relationships and dependencies. • Pretraining on large datasets can lead to powerful and expressive embeddings.	• Requires substantial computational resources and time for pretraining and fine-tuning. • May be prone to overfitting on smaller datasets, and more challenging to interpret and explain compared to simpler embedding methods.

Experiments presented in this work are based on the use of Word2Vec [Mikolov et al., 2013a] to encode sequences of item ids into dense vectors. This method offers several advantages for the task at hand and overcomes some of the limitations of other methods for the specific available data. First, Word2Vec is an efficient and scalable technique, making it suitable for handling large datasets with numerous users and items. This scalability ensures that as the dataset grows, due to the business acquiring more customers and products, the embedding process remains computationally manageable without compromising performance. Furthermore, Word2Vec can capture semantic relationships between items based on their co-occurrence patterns within user interaction sequences. By learning these relationships, the

model can effectively understand users' preferences and make more accurate recommendations.

In contrast, methods like BERT [Devlin et al., 2018] and GloVe [Pennington et al., 2014]have certain disadvantages that make them less suitable for the particular data available for these experiments, although these could be suitable for the same recommendation task with a different dataset. Both BERT and GloVe require pre-training on specific dictionaries before they can be applied to the task, which can be time-consuming and computationally expensive. Additionally, these methods are prone to overfitting when dealing with small datasets, leading to suboptimal performance and reduced generalization capabilities.

While there are some disadvantages to using Word2Vec, such as sensitivity to hyperparameter choices and lack of ability to capture item information or metadata, these drawbacks can be mitigated through careful experimentation and tuning. Additionally, incorporating other techniques or model architectures alongside Word2Vec may help address its limitations and further enhance the recommender system's performance.

4.2.4 Loss function

For model training we use binary cross-entropy as loss function and stochastic gradient descent with backpropagation for optimisation of the model architecture shown in Figure 4.1, therefore the optimisation problem can be written as minimising the loss function defined as

$$Loss = -\frac{1}{\mid U \mid} \sum_{j=1}^{|U|} \sum_{i=1}^{|I'|} y_{i,j} \cdot log(P_{u_j}(i_k))$$

where $y_{i,j}$ is the binary target variable introduced in section 4.2.1 which corresponds to the indicator function of user-item interaction over performance period, and $P_{u_j}(i_k)$ is the estimation of the probability of the neural network.

4.2.5 Ranking Mechanism

As the final goal of a recommender system is to output a list of potentially relevant items for users, a wide range of techniques incorporate ranking methods and ad-hoc ranking losses during training to stream model training and testing and mitigate serving biases [Covington et al., 2016, Zhang et al., 2019, Guy Shani, 2005]. The ranking process also allows incorporation of additional business-related information about the items without affecting

the estimation of probabilities, for example, in a real-world scenario we might be interested in recommending an item which has slightly lower probability of interaction with users, but that it is more profitable for the company that is making recommendations.

In this work we keep the ranking process separated from the overall model architecture at training, as it allows higher flexibility to react to business rules and item offers that can be promoted by companies. The process to obtain the final item list is straightforward by recommending the items which obtain the highest uplift $R(u_j, i_k)$ in terms of user-item interaction probability against the baseline probability of interactions for an item i_k.

$$R(u_j, i_k) = \frac{P_{u_j}(i_k)}{P(i_k)}$$

where $P(i_k)$ represents the probability that a random user interacts with item i_k. $R(u_j, i_k)$ can be interpreted as how many times a user u_j is more likely to interact with item i_k against a random user.

4.2.6 Training process

The training process of the neural network shown in Figure 4.1 is carried by minimising the binary cross-entropy introduced above via backpropagation. To create the input sequences, we decided to split transactions into two disjoint sets to obtain the observation and performance data with respect to an arbitrary analysis date which can be set according to the business needs and frequency of interactions. Then input sequences $Seq(u_j)$ can be created for all users by using only transactions contained in the observation period (prior to the analysis date), whereas targets are built out of transactions in performance (after the analysis date), as shown in in the data pre-processing method stated section 4.2.1. Once sequences are obtained for all users, we split further this data into 80% of the sequences for model training and 20% for validation of results. Pseudo-code included in Appendix B details the full procedure for data pre-processing and model training.

4.3 Experiments and Results

4.3.1 Dataset 1: *Company 1 - Retail*

The first dataset contains transactions made by customers of *company 1*, a retailer in the UK specialised in selling alcoholic beverages. The total volume of transactions in the sampled data consists of nearly 80 millions of purchases made by over 3 million customers between March 2016 and February 2022. Although there are 10,000 different items available for customers over the whole observation period, in average, customers only interact with 24 of these items over their whole customer life cycle, and rarely provide explicit feedback about the purchased items, which induces several challenges at training recommender systems due to the highly data sparsity, the lack of any explicit feedback, and the repeated interactions of customers with the same set of items over time.

For our off-line experiments we used 80% of the total transactional data prior to a defined analysis date (September 2021), allowing to have 6 months of customer transactions for performance, and the 100% of the data after the analysis date for evaluation purposes. Results of the tests for *company 1* are presented in Table 4.4.

Unfortunately, this dataset does not contain a detailed user-item interaction timestamp at minute and second level, but rather purchase events are stored by the company only at event day level (dd/mm/yyyy), which raise the challenge of dealing with simultaneous events stated in section 2.4.5, at not being able to identify the exact order of items in which these were added to the shopping basket or shopping session. Moreover, due to several company's discounts for bulk shopping, i.e., discounts for purchasing multiple items within the same session, 55.5% of the total sequences contain simultaneous events. To assess the effect of not having the correct granularity of data, an experiment was conducted by sorting item ids in ascending and descending order for simultaneous user-item interactions and compare recommendation performance against a recommender system trained with sequences without applying any specific order so simultaneous events. The results of this experiment, presented in table 4.6, show a slight decrease in recommendation performance at enforcing a specific order in interactions due to the lack of granular data, particularly in MAP@10. However, despite not fulfilling the assumption stated in section 4.2, which states that event times do not occur simultaneously for users, it does not necessarily result in a significant deterioration

of recommendation performance for the top ranking of the recommendation list.

4.3.2 Dataset 2: *Company 2 - Retail*

The second dataset contains information of transactions in *company 2*, a large building equipment retailer in the UK with nearly 80 million of purchases made by 1.2 million customers between June 2020 and March 2022. The dataset contains 73,000 potentially recommendable items, although on average each user interacts with only 51 different items over their life cycle.

Similarly to the first dataset, we consider only 80% of customer transactional data prior to a defined analysis date (September 2021) to allow a 6 months performance window, and the rest of transactions for model testing. Model performance for this dataset is presented in Table 4.4. Unlike he first dataset, this dataset contains information of the exact timestamp in which items were added to the shopping basket, for all purchased items, thus model training can be implemented with this timestamp rather than the purchase date in format dd/mm/yyyy.

4.3.3 Dataset 3: *Open dataset: Movielens 25M*

The MovieLens dataset [Harper and Konstan, 2015] is widely used in research and industry to benchmark recommender systems performance. The version of the data used in this work, MovieLens 25M provided by *GroupLens Research*, contains 25 million movie ratings for 62,000 movies and 162,000 users during January 1995 and November 2019. Each of the users in this data have rated at least 20 movies, which mitigates the cold-start problem of not having enough information. As mentioned previously, the data pre-processing needed does not make use of ratings, thus we only consider the interaction between users and items in the data which is stored at timestamp level and no further aggregation is required. Model performance for the MovieLens dataset is presented in Table 4.4.

4.3.4 Results

In this section we empirically demonstrate the effectiveness of our approach by comparing the ranking metrics presented previously: MAP@1, MAP@10, and NDCG for the three mentioned datasets in the retail industry against collaborative filtering and matrix factorisation, Table

4.4 include the details of this performance assessment. Additionally to the off-line evaluation, we evaluate the impact in overall revenue by conducting a live A/B experiment for *Company 1* at comparing sales made with our method against another recommender system based on matrix factorisation used by *Company 1*, further detail of this test is presented in table 4.7.

It is worth mentioning that metrics obtained in this section, MAP and NDCG are computed only from previous purchases made by customers over a period of six months, which do not take into account the fact that customers might not be aware of the existence of all the possible recommendable items in the product catalogue and have only knowledge of popular items, this might induce a bias of equal opportunity in assessing performance of user-item interaction for unpopular items in recommender systems, at these items being inherently less likely to be purchased by users even when in cases where the items are highly relevant. To evaluate our method against CF and MF techniques, while overcoming this issue, we conducted off-line recommendations by training all recommender systems with a sampled dataset which do not considers purchases of the top 10% more popular items in each dataset. Performance results of MAP@1, MAP@10, and NDCG for this experiment are presented in Table 4.5.

Table 4.4: Recommendation performance obtained from off-line experiments with 20% of unobserved customers at training in the validation dataset and allowing 6 months of user-item interactions as the performance period.

	Company 1			*Company 2*			*MovieLens*		
	MAP@1	**MAP@10**	**NDCG**	**MAP@1**	**MAP@10**	**NDCG**	**MAP@1**	**MAP@10**	**NDCG**
Sequence-aware	**0.0119**	0.0233	**0.0314**	**0.0354**	0.1003	**0.1559**	**0.0085**	**0.0124**	**0.0160**
Collaborative Filtering	0.0111	**0.0265**	0.0215	0.0139	0.1016	0.0607	0.0016	0.0046	0.0067
Matrix Factorisation	0.0014	0.0144	0.0265	0.0146	**0.1143**	0.0643	0.0021	0.0046	0.0063

Live A/B test results.

As mentioned, evaluating of recommender systems is usually carried with off-line metrics such as Mean Average Error (MAE) when information about user ratings is known, and ranking metrics like MAP@K and NDCG when there is no information available about real user preferences. Additionally to the off-line evaluation carried for MAP@K and NDCG, we conducted a live A/B test item for item recommendations included in an email marketing

Table 4.5: Recommendation performance from a subset training dataset obtained by removing the 10% most popular items at training and evaluating performance for 20% unobserved customer in the validation dataset with a 6 months of transactions in the observation window.

	Company 1			*Company 2*			*MovieLens*		
	MAP@1	**MAP@10**	**NDCG**	**MAP@1**	**MAP@10**	**NDCG**	**MAP@1**	**MAP@10**	**NDCG**
Sequence-aware	**0.0051**	0.0078	**0.0095**	**0.0050**	0.0092	0.0141	0.0010	0.0081	0.0177
Collaborative Filtering	0.0015	**0.0091**	0.0058	0.0000	**0.0186**	0.0013	**0.0210**	**0.0403**	**0.0535**
Matrix Factorisation	0.0026	0.0055	0.0069	0.0000	0.0053	**0.0162**	0.0000	0.0130	0.0090

Table 4.6: Recommendation performance for dataset 1: retail dataset at sorting simultaneous transactions in ascending and descending order.

	MAP@1	**MAP@10**	**NDCG**
Sequence-aware	**0.0194**	**0.0233**	**0.0314**
Sequence-aware ascending order	0.0165	0.0001	0.0137
Sequence-aware descending order	0.0165	0.0001	0.0137

campaign for *company 1*. In this test, we tested our method against an in-house recommender system for which we cannot provide further details other than that overall performance compares to a matrix factorisation model. This test only considered the top 1 item recommendation from a universe of 8 recommendable items that the company wanted to promote in an email marketing campaign for over 500,000 customers. Six of these items are considered priority for *company 1* and should be ranked first if possible, the other two items are considered popular items and should be sent as a default in case there is no better recommendation for customers. The top 1 recommendation for both systems, as well as the default items were presented as the main banner in the email sent to customers, as illustrated in Figure 4.2, and the rest of the email components such as the header, images, and any other items promoted in the email remained the same for both systems.

Although we cannot compare these systems at ranking level due to confidentiality constraints, we can provide the average ranking probability of purchase for each in the selection of eight items used for the live test. As shown in Figure 4.3, our recommender system could identify the best item to be recommended to customers and rank customers with 20-40 times higher precision at the top of the ranking list. Besides the overall sales results showed that the

Figure 4.2: Email template sent to customers in live A/B test. 50% of customers were selected for recommendation of system A using Matrix factorisation and the rest 50% with sequence-aware recommendations. Only customers without transaction history were selected to be included as part of the default template with popular items.

average customer revenue for users targeted with our system increased by 51% in comparison to users revenue targeted by the company system. Further detail of the A/B test revenue results is presented in Table 4.7.

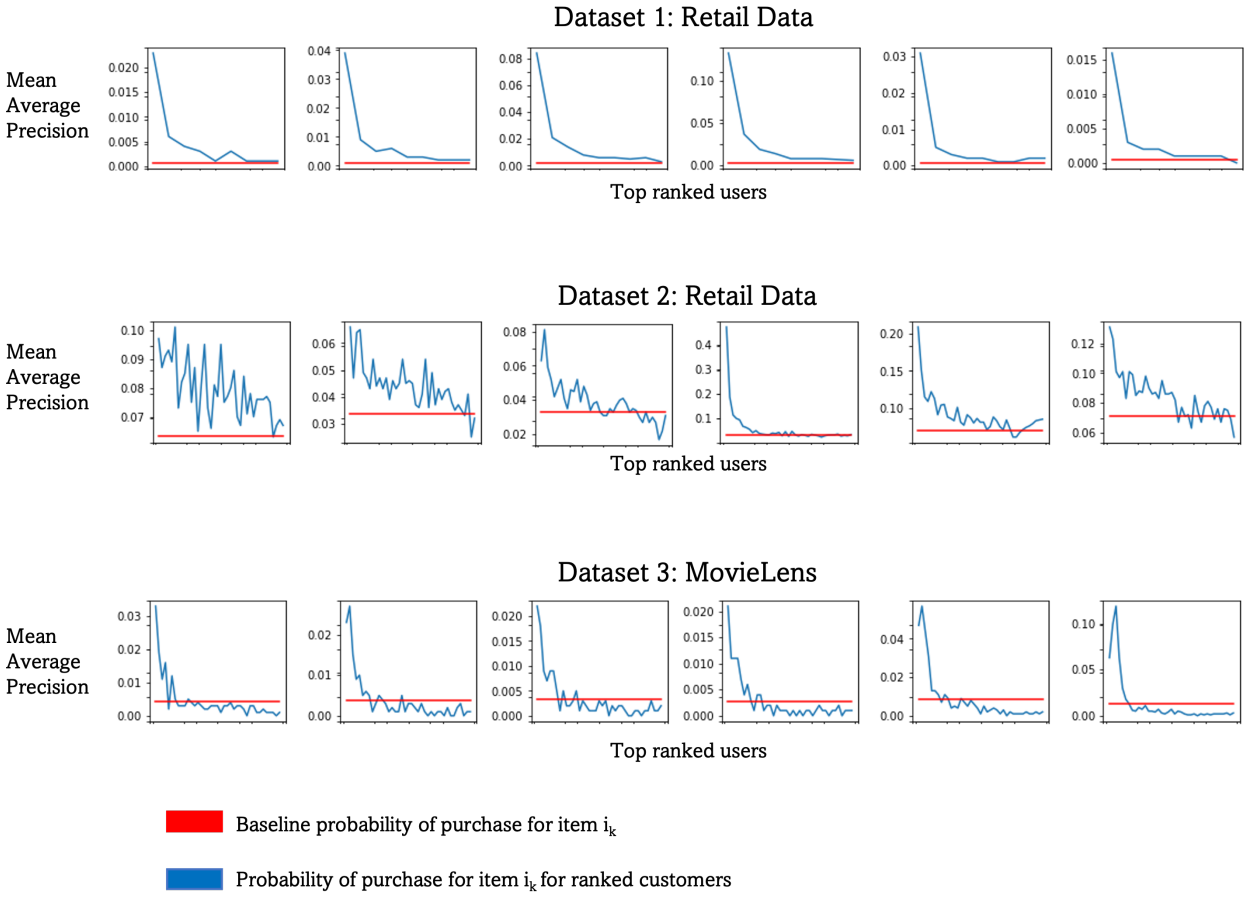

Figure 4.3: Mean Average Precision for Sequence-aware recommendations for 6 products in each dataset selected at random.

Table 4.7: Revenue results of A/B test carried for *Company 1* using our sequence-aware method to make item recommendations against an in-house recommender system based on matrix factorisation. Due to confidentiality constraints with the company providing data and settings for the experiments, sales amounts multiplied by a random number to be presented in this report and preserve confidentiality.

	System A: In-house recommender system				System B: Sequence-Aware recommender system			
Item	Volume of Users	Response	Total Revenue	Revenue p/c	Volume of Users	Response	Total Revenue	Revenue p/c
item 1	7.0%	1.75%	£13,622	£37.73	4.4%	0.99%	£9,030	£69.44
item 2	8.8%	1.17%	£13,720	£45.15	8.9%	0.95%	£20,440	£81.76
item 3	3.7%	0.88%	£2,149	£22.12	16.4%	0.88%	£24,472	£57.05
item 4	11.5%	1.81%	£35,105	£56.70	8.1%	1.02%	£21,042	£85.54
item 5	8.2%	1.49%	£17,003	£46.97	9.2%	0.88%	£20,517	£85.47
item 6	10.6%	0.88%	£14,294	£51.80	6.1%	0.99%	12,719	£70.28
item 7	9.8%	1.27%	£16,079	£43.54	38.9%	0.48%	£25,578	£46.62
item 8	40.5%	0.60%	£15,981	£22.33	7.9%	0.56%	£672	£5.11
Total	100%	1.05%	£127,953	£41.23	100%	0.73%	£134,470	£62.37

4.3.5 Comparison against sequence-aware approaches

Recommender systems have become an essential part of various online platforms, providing personalised experiences to users based on their preferences and behaviour [Quadrana et al., 2018, Baeza-Yates et al., 2015, Covington et al., 2016, Tang and Wang, 2018]. This section aims to present a comparative analysis between the proposed methodology to make sequence-aware recommendations with the use of recurrent neural networks presented in section 4.2, against other three sequence aware recommender systems proposed in the literature, Caser [Tang and Wang, 2018], SASRe [Kang and McAuley, 2018], and Bert4Rec [Sun et al., 2019]. Besides presenting only comparative metrics in terms of recommendation performance, this analysis shows the advantages and disadvantages of each approach, highlighting the importance of considering temporal dynamics and user-item interactions in developing effective recommender systems.

Caser: Convolutional Sequence Embedding Recommendation Model

Caser (Convolutional Sequence Embedding Recommendation Model) [Tang and Wang, 2018] combines both collaborative filtering and sequence-based recommendation models by using convolutional neural networks (CNN) to learn representations of the user-item interactions presented in figure 2.7. Caser takes into account all the items what a given user has interacted

with, alongside their respective interaction order to build an embedding matrix that is used to search for sequential patterns as local features with the use of convolutional filters. Then, the convolutional output feature maps are concatenated and fed into a fully connected layer to generate a representation of the entire sequence, which captures the essential information about the user-item interactions and can be used to make predictions by applying a softmax activation function to produce a probability distribution over all possible recommendable items.

Caser offers several notable advantages over matrix-completion recommendation methods, making it a highly flexible tool for several applications. Its adaptability allows it to handle different types of sequential data, including implicit feedback and explicit ratings, making it suitable for diverse use cases such as e-commerce and media consumption. With the use con convolutional filters, Caser can learn complex patterns and dependencies within the data, resulting in more accurate and personalised recommendations. Finally, its scalability allows it to manage large-scale datasets and make prediction efficiently, ensuring optimal performance even when dealing with millions of users and items.

Nevertheless, Caser also presents certain limitations at implementing it in real-world scenarios. For example, the convolutional architecture used to organise data and extract embeddings, avoids it to handle efficiently the cold-start problem that affects most recommendation models, and provide accurate recommendations for new users or items with limited interaction data. Additionally, Caser requires careful tuning of various hyperparameters, such as the number of convolutional filters used at training, filter sizes, and learning rate, which is require larger efforts and experimentation when used in real-world environments and can lead to time-consuming and computationally expensive processes.

SASRe: Self-Attention based Sequential Recommendation model

SASRe (Self-Attention based Sequential Recommendation model) [Kang and McAuley, 2018] is inspired in the Transformer architecture [Vaswani et al., 2017], which has shown outstanding performance in several NLP and sequential modelling applications. SASRe is designed to capture both short-term and long-term user preferences by analysing the sequential patterns in users' interactions history with an architecture that consists of multiple layers of self-attention blocks, followed by position-wise feed-forward networks. Each self-attention block contains multi-head attention and normalisation layers, which allows it to capture complex

patters in data and weigh the importance of different items by computing the attention scores between all pairs of items in the sequence.

SASRec offers several advantages as a sequential recommendation technique, its attention-based architecture is scalable and can be efficient at handling large datasets without suffering from the vanishing gradient problem commonly found in deep RNN-based models, besides significantly improving the training time in compared to sequential RNN-based models [Kang and McAuley, 2018]. Furthermore, item-pair attention scores obtained during SASRec training process may insights into item relationships over time, which can then be used to enhance model's interpretability.

However, the SASRe approach faces several disadvantages when implemented in production environments. SASRe self-attention mechanism leads to high memory consumption due to the storage of attention scores for all item pairs in the sequence, potentially limiting its applicability on devices with restricted memory resources. And like most recommendation systems, SASRec faces the cold-start problem, struggling to provide accurate recommendations for new users or items with limited interaction history.

BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer

BERT4Rec (Sequential Recommendation with Bidirectional Encoder Representations from Transformer) [Sun et al., 2019] is a transformer-based model built upon the BERT architecture specifically designed for sequential recommendation tasks and has achieved outstanding recommendation performance against matrix-completion and deep learning-based recommender systems [Sun et al., 2019] it uses pre-training and fine-tuning mechanisms of BERT (Bidirectional Encoder Representations from Transformers) [Devlin et al., 2018] to learn representations for items in a user's interaction sequences and capture both short-term and long-term user preferences. In Bert4Rec, each item in a sequence is represented as embeddings, which are then fed into the transformer architecture with typically additional positional encodings to enable the mechanism to add information related to users or items features.

During embedding pre-training, a percentage of items in the input sequence are masked and the model learns to predict these masked items based on the context provided by the unmasked items. Once this process is completed, the embedding model can be fine-tuned on a specific recommendation task using user-item interaction data.

Bert4Rec offers multiple advantages as a recommendation system. Its ability to capture both short-term and long-term dependencies in user-item interactions allows it to provide highly accurate and context-aware recommendations. Additionally, BERT4Rec architecture based on transformers allows to parallelisation during training, making it suitable to handle very large data. However, Bert4Rec also presents some drawbacks as a recommendation system, its architecture is particularly computational complex during the pre-training phase, which may restrict its applicability in resource-constrained settings and may require specialised hardware like GPUs for efficient training. As most sequence-aware recommender systems, Bert4Rec's performance can be sensitive to hyperparameter choices, requiring careful tuning and experimentation to achieve optimal results. Finally, like many deep learning models, Bert4Rec lacks recommendation interpretability, which might be a concern for several commercial applications where transparency is crucial.

Performance and applicability comparison

While attention-based sequence-aware methods for item recommendation presented in this section may outperform the RNN-based approach presented in section 4.2 in several applications, the RNN-based approach is a competitive alternative to several sequence-aware recommender systems, like Caser, SASRe, and BERT4Rec. It addresses the temporal dynamics of user behaviour interacting with items with the use of recurrent neural networks, provides scalability to handle large datasets efficiently, allows to incorporate additional user and item features during training, and offers higher levels of interpretability in compared to attention-based models, making it a practical solution for businesses dealing with increasing amounts of data.

Although, due to data usage limitations, it is not possible to make a direct performance comparison against attention-based recommender systems with the retail datasets presented section 4.3. Table 4.8 shows prediction performance reported in the literature of alternative sequence-aware recommender systems mentioned in this section, applied to the Movielens dataset [Tang and Wang, 2018, Sun et al., 2019]. To obtain these metrics, in compared to results presented in tables 4.4 and 4.5, where all users are considered regardless of if they made an interaction over the performance period, table 4.8 show performance metrics for user who made at least one interaction with an item of the recommendable set of items during the performance period.

Table 4.8: Recommendation performance reported in literature of sequence-aware recommender systems for the Movielens 25m dataset [Tang and Wang, 2018, Sun et al., 2019]. Performance is obtained considering users that made an interaction with a recommendable item over the performance period.

	MAP@1	**Recall@1**	**Recall@10**	**NDCG**
Caser	0.2502	0.1232	0.5427	0.3062
SASRe	0.4026	0.2544	0.7136	0.4665
BERT4Rec	**0.4785**	**0.344**	0.7473	**0.534**
Sequence-aware RNN	0.2705	0.2705	**0.87352**	0.532

Furthermore, it is essential to consider the specific requirements and constraints of each application when choosing the most suitable sequence-aware recommender system. While attention-based methods offer several advantages, they may not always be the best choice for every scenario. For instance, if computational resources or data are limited. Some relevant factors to consider in the choice of the right approach include:

Problem domain and data characteristics: The choice of the recommendation approach requires a thorough understanding of the problem domain and main data characteristics. For instance, if the dataset exhibits strong temporal patterns or if the application requires capturing the evolution of user preferences over time, RNN-based methods might be more appropriate, as this method is more flexible to adjust to quick changes in user preferences. Alternatively, if the main goal is capturing complex relationships between items or users, methods like Caser or SASRe might be more suitable, as these focus on the relationship between items embeddings.

Data availability: Attention-based models like BERT4Rec and SASRe require of extensive pre-training to capture meaningful relationships in data, which is not possible for small and medium size datasets. Alternatively, RNN-based methods and Caser require less data during training and might be more effective where data is not vast.

Computational Resources and Training Time: The availability of computational resources is a key constrain in the selection of a recommendation approach. While RNN-based methods can efficiently handle large datasets, the training time of these can typically be performed in commercial hardware. In contrast, although methods like SASRe and BERT4Rec may provide

higher accuracy, these might require specialised hardware like GPUs for efficient training, particularly in the pre-training phase of the embedding layers.

Recommendation Performance and Accuracy: While RNN-based methods can provide accurate recommendations by capturing temporal dynamics and sequential patterns, as shown in table 4.8, other methods like BERT4Rec might achieve better recommendation performance in several scenarios.

Interpretability and Transparency: RNN-based methods offer more transparent insights into user behaviours and preferences compared to attention-based models. Which might be an important consideration where transparency and explainability are key requirements in the application.

Ease of Implementation and Maintenance: While RNN-based methods can be relatively straightforward to implement and maintain, other methods like BERT4Rec or Caser might require more complex architectures and careful tuning of hyperparameters over time. However, the RNN-based approach might be more sensitive to changes in the underlying data distribution and user preferences, requiring more frequent updates and retraining.

Robustness to Cold-start Problem: While most sequence-aware recommender systems like RNN-based, and BERT4Rec alleviate the cold-start problem for new users in the system due to their architecture to process sequential information in user-item interactions. Methods like Caser might struggle more with the cold-start problem in compared to other sequential approaches, due to its structure based in convolutional filters. Furthermore, attention-based recommender might be the most flexible at handling new items in the system due to their structure to effectively capture item-to-item relationships through the pre-training process.

Flexibility and Adaptability: Sequence-aware recommender systems are inherently flexible and adaptable, making them suitable for various types of sequential data, including implicit feedback and explicit ratings. In particular, the RNN-based proposed in section 4.2 does not require the extensive pre-training process that attention-based system like BERT4Rec need. Making it suitable for faster implementations in production environments.

4.4 Conclusion

This work presents an innovative approach to make item recommendations for individual customers while considering the order than previous purchases were made, all by using

recurrent neural networks and data pre-processing methods traditionally used in the field of natural language processing to make predictions of words. Although sequence-aware recommender systems have been widely explored and used in fields where customer preferences change drastically in short periods of time, such as mobile app usage, and video games applications such as the ones mentioned in section 4.1, to the best of our knowledge there are no applications of sequence-aware item recommendations for the retail industry as the ones presented in the datasets used section 4.3.4.

The main motivation of this work is performing item recommendations in settings such as retail where matrix-completion like collaborative filtering and matrix factorisation approaches presented in section 2.4.2 do not perform well for to several different reasons, while recommendation performance is at least still similar to the mentioned and widely used methods, specially for the top of the recommendation list. For instance, user preferences might change rapidly over time, or users may interact repeatedly with specific items without providing explicit ratings.

The proposed sequence-aware recommender system in this work overcome these issues and maintain predictive performance comparable to other recommendation techniques by inherently assuming that items may be present in the purchasing sequence multiple times, and all items, including the ones previously bought by users, may be contained in the set of recommendable items. This change allows us to achieve two main goals: firstly, improve customer engagement by identifying the best products for users, regardless of whether these have been already purchased by users. Secondly, improve customer journey by identifying which items are most likely to be purchased in sequential order, and provide tailored recommendations with special offers.

Additionally, the cost of maintaining our recommender system in production environments is lower than maintaining matrix completion methods, at these need to be constantly updated to account for new user-item interactions, whereas our method only considers this change in the input data fed into the network, which is extremely convenient for businesses and machine learning engineers. Furthermore, our method generates item recommendations considerably faster than matrix completion techniques, at not needing to estimate preferences for all users and all items simultaneously, but one user at the time instead.

However, although our method has achieved great performance in compared to the baselines models outlined in section 4.3, and comparable performance against alternative

sequence-aware recommendation techniques,it still has its limitations. As the final probability for user-item interactions is estimated from the soft-maxed output layer of the recurrent neural network, our methodology is prone to suffer selection bias induced from highly popular items. This effect occurs at observing an extremely imbalanced number of interactions for a specific item during training, which might not persist at prediction phase, and therefore affecting the item ranking overall. Additionally, our methodology suffers severely from cold start problems at introducing new items for ranking, as training targets introduced in section 4.2.1 during data pre-processing are built from a specific period for each item. Finally, as shown in table 4.5, our method might not be the best performing where the size of the recommendation list for each customer is large, but rather in recommendation scenarios where the goal is finding the top 1 best item to recommend based in the most recent data.

Our methodology presented in this work could be easily extended in different ways to improve item recommendation performance, as the main focus of our approach is just encoding and capturing information of previous purchases for each customer, we are so far ignoring information from users and items characteristics, as well as potential hand-crafted features that can be easily added to the input of the neural network or to any other model that could be used to perform token prediction.

CHAPTER 5

Final Conclusions

5.1 Contributions

The first part of this thesis introduces the concepts of deep learning and representation learning, and it explores several techniques used in research and industry to represent data efficiently with the use of neural networks such as recurrent neural networks, bayesian neural networks, and autoencoders. These techniques offer several advantages in data representation with respect to hand-crafted feature engineering:

1. Reliability: These techniques can obtain reliable representations of complex high-dimensional data distributions. This by simple connecting either probabilistic or recurrent layers to the model to perform automatic feature extraction.

2. Scalable and flexible: Since the representations are obtained without human interventions, it is relatively simple to replicate these methods in multiple applications where high volumes of data are available without requiring to allocate human labour to process and analyse data, which is the case is most marketing applications.

3. Easy to implement: Representation learning techniques can be easily implemented with the use of popular deep learning libraries such as *Tensorflow* or *Pytroch* which can automatically perform the learning process with backpropagation for any differentiable architecture.

The second part of this thesis introduces two novel applications to obtain data representation for customer interactions and how these can be used to predict whether customers are likely to make a next purchase (customer churn in Chapter 3), or likely to interact with a specific item in the products catalogue (item recommendation in Chapter 4). These two chapters show how customer transactions can be seen as sequential information to be analysed with Bayesian and recurrent neural networks to produce insights about customers' future behaviour, without the need of creating additional hand-crafted attributes for customers as it is traditionally done in most marketing applications.

However, although several representation learning methods have been proposed in the literature, there are several challenges while implementing these techniques:

1. Interpretability: As almost all the methods presented in section 2.2 are based on the training of either bayesian or recurrent neural networks, it is inherently difficult to associate the distribution of network weights to particular characteristics of data, thus making it almost impossible to give interpretable results of the representation space with respect to the data itself.

2. Evaluation: There has not been enough research on how representation space methods perform against hand-crafted feature engineering, besides comparing final model performance in terms of accuracy and precision. And in general, achieving a unified framework for comparison is still not clear. Although representation learning methods make sure that data representations learnt to match the input data distribution and models trained with representation only achieve good performance, it is still not clear which method is better in general and for particular applications.

3. Experimentation: Most representation learning methods require large efforts of design and experimentation, and the choice of model hyper-parameters and/or prior distributions impacts directly the data representation obtained. Therefore, although these methods are inherently robust and allow streamlining the analysis of complex high-dimensional data, there is still some level of ambiguity and human input in the development of machine learning applications with the use of representation learning.

5.2 Open questions for future work

It is well known that every new advancement in science starts with a new set of open questions that are often harder to answer than the initial ones. This work introduces two novel methodologies to obtain customer churn and item recommendation predictions for the retail industry with the use of representation learning and deep learning techniques. Along the way in the development of this work, several questions for further research were raised:

Use of attention mechanisms to find representations:

In just a few years the attention mechanism and transformers presented by Vaswani et al. [Vaswani et al., 2017] have reshaped the way that most machine learning applications work, their flexibility in finding general data representation in NLP and CV tasks allows practitioners to implement these techniques as out-of-the-box solutions for several scenarios. It is inevitable asking if the attention mechanism could be modified to find specific data representation that led to customer insights, without necessarily having a specific application in mind.

A unified framework to assess effectiveness of hand-crafted features against data representations:

Nowadays, most of the research focuses in only one or the other approach to develop new ML applications, without a robust comparison of which method is better at training machine learning models besides comparing the final model performance of the task at hand, and under which scenarios and applications which approach is better. Having a unified framework to compare the effectiveness and efficiency of different approaches will lead to a much better understanding of the field and its potential caveats for different applications, and particularly when it is useful to combine both methods to maximise performance.

Multi-objective model predictions from data representations:

Several modern machine learning techniques aim to learn multiple objectives simultaneously during training, particularly deep learning techniques achieve this by modifying the architecture presented in section 2.1.2 and include multiple output layers in their structure and perform backpropagation with two or more loss functions simultaneously which can be combined depending the task at hand. The learning of data representation for multi-task

neural networks is not a widely explored field aside applications in NLP for document retrieval and text classification and summarising. Future work in marketing applications could potentially explore the connection of learning representation to predict multiple customer insights simultaneously, such as customer churn, customer lifetime value, and item purchase intent with the use of the same model architecture.

The use of variational autoencoder to learn data representations in marketing sequences:

As stated in section 2.2.5 the variational autoencoder is a popular unsupervised learning technique to learn data representations from complex data distribution, such as images. Particularly the variational recurrent neural network (VRNN) [Chung et al., 2015] extend the concept of the VAE to learn representations of sequential data, which could be particularly useful in the applications presented in chapters 3 and 4 which analyse customer transactions as sequential information via embeddings and neural networks. Further experimentation with VAE and VRNN could potentially replace these layers in the proposed architectures to achieve robustness in the work presented in this thesis, and their seamlessly extension to other applications domains out of marketing sciences.

APPENDIX A

Appendix 1: Pseudo-code for modelling customer churn with bayesian neural networks.

Profusion is a data consultancy company based in London, UK, that provides data and analytic services to a wide range of businesses in the retail and financial sectors across the UK. Profusion uses different statistical and machine learning methods to identify customers likely to churn and make item recommendations. The data used for the experiments in this work is provided by Profusion, and the code and detailed methodology is restricted to Profusion use only. Nevertheless, pseudo-code with all the necessary details for the methodologies presented in Chapters 3 and 4 is provided in this section.

Algorithm 1 Pseudo-code for modelling customer churn with bayesian neural networks.

Input: Customer transactional data presented in table 3.1 including *'Customer id'* and *'Purchase date'*.

Output: Recurrent Neural Network to predict next customer event time.

Split data into training, testing, and validation datasets:

- Split customer set into two disjoint training and validation sets containing 80% and 20% of customer's data respectively.
- Split training customer transactions into two disjoint training and testing datasets w.r.t. the selected analysis date.

Data processing:

1. Obtain time between subsequent purchases for each customer as $t_{k,i} = d_{k,i} - d_{k,i-1}$.
2. Obtain sequence input vectors for each customer as $T_X = \bigcup_{k \in K} \mathbf{t}_k \cup t_{k,n'_k}$
3. Pad sequences the same length with a sliding window over each customer sequence and assign the target event time as the immediate next arrival time in the sequence.

Model Training:

Repeat until convergence:

1. Initialise recurrent neural network with randomised weights and biases.
2. Apply network forward pass to the input sequences to obtain the estimated next arrival time or each customer.
3. Compute the model loss by using the real observed and censored arrival times and the estimated next arrival times.
4. Update weights and biases in the network.

Model Prediction:

1. Create input sequences for validation dataset as $T_X = \bigcup_{k \in K} \mathbf{t}_k \cup t_{k,n'_k}$
2. Apply forward pass of the recurrent model to obtain the next estimated time events.
3. Sample estimated next event time $t_{k,j+1}$ from the posterior distribution of event times for each customer to estimate $\hat{\lambda}_k$ for each customer
4. Make predictions of estimated customer survival probability by obtaining $\hat{S}_k(t) = exp(-\hat{\lambda}_k t)$

APPENDIX B

Appendix 2: Pseudo-code for modelling probability of user-item interactions in a sequential-aware framework.

Algorithm 2 Pseudo-code for modelling probability of user-item interactions in a sequential-aware framework.

Input: Customer transactional data presented in table 4.1 including *'Customer id'*, *'Purchase date'*, *'Item id'*.

Output: Item list recommendations for users.

Data Split:

- Split training w.r.t the set analysis date to get observation and performance transactional data.
- Split total data into two disjoint sets of customers for training (80%) and validation (20%).

Data Pre-processing:

- With the observation transactional data, for each user in the dataset create $Seq(u_j)$ with the time-ordered product ids of previous items purchased by user u_j as stated in the data pre-processing method in section 4.2.1.
- With performance data, for each user create the target vector

$$Y_{u_j} = [y_{u_j,i_1}, y_{u_j,i_2}, ..., y_{u_j,i_{|I|}}],$$

with user-item interactions in the performance period as stated in data pre-processing section 4.2.1.

Model Training:

- Train the sequence-aware model via backpropagation with the architecture presented in figure 4.1, the data obtained from the pre-processing step, and the binary cross-entropy outlined section 4.2 as loss function.

Model Prediction:

- For each user, obtain the probability of interaction with each potentially recommendable item $\hat{P}_u = [\hat{P}_{u_j}(i_1), ..., \hat{P}_{u_j}(i_{|I|})]$.

Ranking Process:

- For each user u_j and item $i_k \in \mid I' \mid$, obtain the estimated uplift $R(u_j, i_k)$ in probability interaction as stated in the ranking mechanism section

$$R(u_j, i_k) = \left[\frac{\hat{P}_{u_j}(i_1)}{P(i_1)}, ..., \frac{\hat{P}_{u_j}(i_{|I'|})}{P(i_{|I'|})} \right]$$

- For each user, recommend the top K items for which $R(u_j, i_k)$ is the largest.

Bibliography

[Aggarwal, 2016] Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer Publishing Company, Incorporated, Switzerland, 1st edition.

[Amico and Van Keilegom, 2018] Amico, M. and Van Keilegom, I. (2018). Cure models in survival analysis. *Annual Review of Statistics and Its Application*, 5(1):311–342.

[Baeza-Yates et al., 2015] Baeza-Yates, R., Jiang, D., Silvestri, F., and Harrison, B. (2015). Predicting the next app that you are going to use. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 285–294, New York, NY, USA. Association for Computing Machinery.

[Balabanovic and Shoham, 1997] Balabanovic, M. and Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40:66–72.

[Ballard, 1987] Ballard, D. H. (1987). Modular learning in neural networks. In *Proceedings of the sixth National conference on Artificial intelligence*, AAAI'87, pages 279–284, Seattle, Washington. American Association for Artificial Intelligence.

[Basu et al., 1998] Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98, pages 714–720, Madison, Wisconsin, USA. American Association for Artificial Intelligence.

[Bell et al., 2007] Bell, R., Koren, Y., and Volinsky, C. (2007). Modeling relationships at multiple scales to improve accuracy of large recommender systems. KDD '07, pages 95–104, New York, NY, USA. Association for Computing Machinery.

[Bender et al., 2005] Bender, R., Augustin, T., and Blettner, M. (2005). Generating survival times to simulate cox proportional hazards models. *Statistics in Medicine*, 24(11):1713–1723.

[Bengio et al., 2013] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35:1798–1828.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

[Bengio et al., 2014] Bengio, Y., Thibodeau-Laufer, E. r., Alain, G., and Yosinski, J. (2014). Deep generative stochastic networks trainable by backprop. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages 226–234. JMLR.org.

[Bennis et al., 2020] Bennis, A., Mouysset, S., and Serrurier, M. (2020). Estimation of conditional mixture weibull distribution with right-censored data using neural network for time-to-event analysis. *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020*, pages 687–698.

[Borah et al., 2020] Borah, S., Prakhya, S., and Sharma, A. (2020). Leveraging service recovery strategies to reduce customer churn in an emerging market. *Journal of the Academy of Marketing Science*, 48:848–868.

[Burke, 2002] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370.

[Camacho-Collados and Pilehvar, 2018] Camacho-Collados, J. and Pilehvar, M. T. (2018). On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP.*, pages 40–46. Association for Computational Linguistics.

[Chamberlain et al., 2017] Chamberlain, B. P., Cardoso, A., Liu, C. B., Pagliari, R., and Deisenroth, M. P. (2017). Customer lifetime value prediction using embeddings. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1753–1762, New York, NY, USA. Association for Computing Machinery.

[Chapfuwa et al., 2018] Chapfuwa, P., Tao, C., Li, C., Page, C., Goldstein, B., Carin, L., and Henao, R. (2018). Adversarial time-to-event modeling. *Proceedings of machine learning research*, 80:735–744.

[Chen et al., 2018] Chen, T., Keng, B., and Moreno, J. (2018). Multivariate arrival times with recurrent neural networks for personalized demand forecasting. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 810–819. IEEE Computer Society.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *Presented in NIPS 2014 Deep Learning and Representation Learning Workshop*, abs/1412.3555.

[Chung et al., 2015] Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2980–2988, Cambridge, MA, USA. MIT Press.

[Claypool et al., 1999] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., and Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley, California. ACM.

[Coussement and Van den Poel, 2008] Coussement, K. and Van den Poel, D. (2008). Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. *Expert Systems with Applications*, 34(1):313–327.

[Covington et al., 2016] Covington, P., Adams, J., and Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 191–198, New York, NY, USA. Association for Computing Machinery.

[Cox, 1972] Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220.

[Davenport et al., 2019] Davenport, T., Guha, A., Grewal, D., and Bressgott, T. (2019). How artificial intelligence will change the future of marketing. *Journal of the Academy of Marketing Science*, 48:1–19.

[De Souza Pereira Moreira et al., 2021] De Souza Pereira Moreira, G., Rabhi, S., Lee, J. M., Ak, R., and Oldridge, E. (2021). Transformers4rec: Bridging the gap between nlp and sequential / session-based recommendation. In *Fifteenth ACM Conference on Recommender Systems*, pages 143–153, New York, NY, USA. Association for Computing Machinery.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. cite arxiv:1810.04805Comment: 13 pages.

[Dilokthanakul et al., 2016] Dilokthanakul, N., Mediano, P. A. M., Garnelo, M., Lee, M. C. H., Salimbeni, H., Arulkumaran, K., and Shanahan, M. (2016). Deep unsupervised clustering with gaussian mixture variational autoencoders. *CoRR: a computing research repository*, abs/1611.02648.

[Doshi et al., 2011] Doshi, F., Wingate, D., Tenenbaum, J., and Roy, N. (2011). Infinite dynamic bayesian networks. pages 913–920, Washington, USA. The 28th International Conference on Machine Learning.

[Fader et al., 2005] Fader, P., Hardie, B., and Lee, K. (2005). âcounting your customersâ the easy way: An alternative to the pareto/nbd model. *Marketing Science*, 24:275–284.

[Fawcett, 2006] Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874.

[Fernandez-Tobias et al., 2012] Fernandez-Tobias, I., Cantador, I., Kaminskas, M., and Ricci, F. (2012). Cross-domain recommender systems: A survey of the state of the art. *Proceedings of the 2nd Spanish Conference on Information Retrieval*.

[Fisher and Lin, 1999] Fisher, L. D. and Lin, D. Y. (1999). Time-dependent covariates in the cox proportional-hazards regression model. *Annual Review of Public Health*, 20(1):145–157.

[Fouss and Saerens, 2008] Fouss, F. and Saerens, M. (2008). Evaluating performance of recommender systems: An experimental comparison. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 735–738.

[Gal and Ghahramani, 2015] Gal, Y. and Ghahramani, Z. (2015). Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv*, abs/1506.02158.

[Giunchiglia et al., 2018] Giunchiglia, E., Nemchenko, A., and van der Schaar, M. (2018). Rnn-surv: A deep recurrent model for survival analysis. In *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 23–32, Cham. Springer International Publishing.

[Graf et al., 1999] Graf, E., Schmoor, C., Sauerbrei, W., and Schumacher, M. (1999). Assessment and comparison of prognostic classification schemes for survival data. *Statistics in medicine*, 18(17-18):2529–2545.

[Gul et al., 2020] Gul, N., Faiz, N., Brawn, D., Kulakowski, R., Khan, Z., and Lausen, B. (2020). Optimal survival trees ensemble. *arXiv: Applications*.

[Gunawardana and Shani, 2009] Gunawardana, A. and Shani, G. (2009). A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, 10:2935–2962.

[Guy Shani, 2005] Guy Shani, David Heckerman, R. I. B. (2005). An mdp-based recommender system. *Journal of Machine Learning Research*, 6(43):1265–1295.

[Hadden et al., 2006] Hadden, J., Tiwari, A., Roy, R., and Ruta, D. (2006). Churn prediction: Does technology matter. *International Journal of Intelligent Technology*, 1:104–110.

[Hariri et al., 2012] Hariri, N., Mobasher, B., and Burke, R. (2012). Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 131–138, New York, NY, USA. Association for Computing Machinery.

[Harper and Konstan, 2015] Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4).

[Harrell et al., 1982] Harrell, F., Califf, R., Pryor, D., Lee, K., and Rosati, R. (1982). Evaluating the yield of medical tests. *JAMA*, 247(18):2543–6.

[Hashmi et al., 2013] Hashmi, N., Butt, N. A., and Iqbal, D. (2013). Customer churn prediction in telecommunication a decade review and classification. *IJCSI*, 10:271–282.

[He et al., 2017] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 173–182, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

[Heagerty et al., 2000] Heagerty, P. J., Lumley, T., and Pepe, M. S. (2000). Time-dependent roc curves for censored survival data and a diagnostic marker. *Biometrics*, 56(2):337–344.

[Heagerty and Zheng, 2005] Heagerty, P. J. and Zheng, Y. (2005). Survival model predictive accuracy and roc curves. *Biometrics*, 61(1):92–105.

[Hidasi and Karatzoglou, 2018] Hidasi, B. and Karatzoglou, A. (2018). Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 843â852, New York, NY, USA. Association for Computing Machinery.

[Hinton and Salakhutdinov, 2006] Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science (New York, N.Y.)*, 313:504–7.

[Hinton and McClelland, 1987] Hinton, G. E. and McClelland, J. L. (1987). Learning representations by recirculation. In *Proceedings of the 1987 International Conference on Neural Information Processing Systems*, NIPS'87, pages 358–366, Cambridge, MA, USA. MIT Press.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554.

[Hochreiter, 1998] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 8,9:1735–1780.

[Hofmann, 2004] Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115.

[Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558.

[Hu et al., 2008] Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272.

[Hung and Chiang, 2010] Hung, H. and Chiang, C.-T. (2010). Estimation methods for time-dependent auc models with survival data. *The Canadian Journal of Statistics / La Revue Canadienne de Statistique*, 38(1):8–26.

[Hung et al., 2006] Hung, S.-Y., Yen, D. C., and Wang, H.-Y. (2006). Applying data mining to telecom churn management. *Expert Systems with Applications*, 31(3):515–524.

[Hyvärinen et al., 2009] Hyvärinen, A., Hurri, J., and Hoyer, P. O. (2009). *Independent Component Analysis*, pages 151–175. Springer London, London.

[Jamal and Bucklin, 2006] Jamal, Z. and Bucklin, R. E. (2006). Improving the diagnosis and prediction of customer churn: A heterogeneous hazard modeling approach. *Journal of Interactive Marketing*, 20(3):16–29.

[Järvelin and Kekäläinen, 2002] Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

[Jazayeriy et al., 2018] Jazayeriy, H., Mohammadi, S., and Band, S. (2018). A fast recommender system for cold user using categorized items. *Mathematical and Computational Applications*, 23:1.

[Kalbfleisch and Prentice, 2002] Kalbfleisch, J. and Prentice, R. (2002). *The Statistical Analysis of Failure Time Data, Second Edition*, pages 247–277.

[Kamarudin et al., 2017] Kamarudin, A. N., Cox, T., and Kolamunnage-Dona, R. (2017). Time-dependent roc curve analysis in medical research: current methods and applications. *BMC Medical Research Methodology*, 17(1):53.

[Kang and McAuley, 2018] Kang, W. and McAuley, J. (2018). Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206, Los Alamitos, CA, USA. IEEE Computer Society.

[Kaplan and Meier, 1958] Kaplan, E. L. and Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282):457–481.

[Kim et al., 2022] Kim, Y., Hwangbo, H., Lee, H., and Lee, W. (2022). Sequence aware recommenders for fashion e-commerce. *Electronic Commerce Research*, pages 1–21.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv*.

[Kolen and Kremer, 2001] Kolen, J. F. and Kremer, S. C. (2001). *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pages 237–243.

[Koren et al., 2009] Koren, Y., R.Bell, and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(08):30–37.

[Kvamme et al., 2019] Kvamme, H., Borgan, Ø., and Scheel, I. (2019). Time-to-event prediction with neural networks and cox regression. *Journal of Machine Learning Research*, 20:129:1–129:30.

[Lambert and Chevret, 2016] Lambert, J. and Chevret, S. (2016). Summary measure of discrimination in survival models based on cumulative/dynamic time-dependent roc curves. *Statistical Methods in Medical Research*, 25(5):2088–2102.

[Lampinen and Vehtari, 2001] Lampinen, J. and Vehtari, A. (2001). Bayesian approach for neural networksâreview and case studies. *Neural Networks*, 14(3):257–274.

[LeBlanc and Crowley, 1993] LeBlanc, M. and Crowley, J. (1993). Survival trees by goodness of split. *Journal of the American Statistical Association*, 88(422):457–467.

[Lee et al., 2006] Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2006). Efficient sparse coding algorithms. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 801–808, Cambridge, MA, USA. MIT Press.

[Lee et al., 2008] Lee, T. Q., Park, Y., and Park, Y.-T. (2008). A time-based approach to effective recommender systems using implicit feedback. *Expert Systems with Applications*, 34(4):3055–3062.

[Linden et al., 2003] Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80.

[Liu et al., 2020] Liu, F., Zheng, L., and Zheng, J. (2020). Hienn-dwe: A hierarchical neural network with dynamic word embeddings for document level sentiment classification. *Neurocomputing*, 403:21–32.

[Liu, Wenbin et al., 2020] Liu, Wenbin, Wen, Bojian, Gao, Shang, Zheng, Jiesheng, and Zheng, Yinlong (2020). A multi-label text classification model based on elmo and attention. *MATEC Web Conf.*, 309:03015.

[Lu, 2002] Lu, J. (2002). Predicting customer churn in the telecommunications industry - an application of survival analysis modeling using sas. *SAS User Group International (SUGI27) Online Proceedings*, 114:27.

[Marchand and Marx, 2020] Marchand, A. and Marx, P. (2020). Automated product recommendations with preference-based explanations. *Journal of Retailing*, 96(3):328–343.

[Martinsson, 2017] Martinsson, E. (2017). *WTTE-RNN: Weibull Time To Event Recurrent Neural Network (Doctoral dissertation or Master's thesis)*. PhD thesis, Chalmers University Of Technology, Gothenburg, Sweden.

[Mavri and Ioannou, 2008] Mavri, M. and Ioannou, G. (2008). Customer switching behaviour in greek banking services using survival analysis. *Managerial Finance*, 34:186–197.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings*

of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, page 3111â3119, Red Hook, NY, USA. Curran Associates Inc.

[Minsky and Papert, 1969] Minsky, M. and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA.

[Murray and Adams, 2010] Murray, I. and Adams, R. P. (2010). Slice sampling covariance hyperparameters of latent gaussian models. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, page 1732â1740, Red Hook, NY, USA. Curran Associates Inc.

[Nagpal et al., 2020] Nagpal, C., Li, X., and Dubrawski, A. (2020). Deep survival machines: Fully parametric survival regression and representation learning for censored data with competing risks. *IEEE journal of biomedical and health informatics*, 25(8):3163–3175.

[Nagpal et al., 2019] Nagpal, C., Sangave, R., Chahar, A., Shah, P., Dubrawski, A., and Raj, B. (2019). Nonlinear semi-parametric models for survival analysis. *CoRR: a computing research repository*, abs/1905.05865.

[Naz et al., 2018] Naz, N., Shoaib, U., and Sarfraz, M. (2018). A review on customer churn prediction data mining modeling techniques. *Indian Journal of Science and Technology*, 11:1–7.

[Nelder and Wedderburn, 1972] Nelder, J. A. and Wedderburn, R. W. M. (1972). Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384.

[Nunez Valdez et al., 2018] Nunez Valdez, E., Quintana, D., Gonzalez Crespo, R., Isasi, P., and Herrera-Viedma, E. (2018). A recommender system based on implicit feedback for selective dissemination of ebooks. *Information Sciences*, 467:87–98.

[Paterek, 2007] Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*.

[Pearson, 1901] Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods*

in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

[Phong and Ribeiro, 2020] Phong, N. H. and Ribeiro, B. (2020). Rethinking recurrent neural networks and other improvements for image classification. *arXiv*, abs/2007.15161.

[Quadrana et al., 2018] Quadrana, M., Cremonesi, P., and Jannach, D. (2018). Sequence-aware recommender systems. *ACM Computing Surveys*, 51(4).

[Ravula et al., 2022] Ravula, P., Jha, S., and Biswas, A. (2022). Relative persuasiveness of repurchase intentions versus recommendations in online reviews. *Journal of Retailing*.

[Rawat and Wang, 2017] Rawat, W. and Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29:1–98.

[Reichheld, 1990] Reichheld, F. (1990). Zero defections-quality comes to services. *Harvard Business Review*, Sep.-Oct.:105–111.

[Ren et al., 2019] Ren, K., Qin, J., Zheng, L., Yang, Z., Zhang, W., Qiu, L., and Yu, Y. (2019). Deep recurrent survival analysis. Honolulu, Hawaii, USA. AAAI Press.

[Resnick et al., 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA. Association for Computing Machinery.

[Rezende et al., 2014] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China. PMLR.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408.

[Rothenbuehler et al., 2015] Rothenbuehler, P., Runge, J., Garcin, F., and Faltings, B. (2015). Hidden markov models for churn prediction. In *2015 SAI Intelligent Systems Conference (IntelliSys)*, pages 723–730.

[Rumelhart and McClelland, 1987] Rumelhart, D. E. and McClelland, J. L. (1987). *Learning Internal Representations by Error Propagation*, pages 318–362.

[Salakhutdinov and Mnih, 2007] Salakhutdinov, R. and Mnih, A. (2007). Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1257–1264, Red Hook, NY, USA. Curran Associates Inc.

[Schmittlein et al., 1987] Schmittlein, D. C., Morrison, D. G., and Colombo, R. (1987). Counting your customers: Who are they and what will they do next? *Management Science*, 33(1):1–24.

[Shaaban et al., 2012] Shaaban, E., Helmy, Y., Khedr, A., and Nasr, M. (2012). A proposed churn prediction model. *International Journal of Engineering Research and Applications (IJERA*, 2:693–697.

[Sharma and Panigrahi, 2011] Sharma, A. and Panigrahi, D. P. K. (2011). Article: A neural network based approach for predicting customer churn in cellular network services. *International Journal of Computer Applications*, 27(11):26–31.

[Sidana et al., 2021] Sidana, S., Trofimov, M., Horodnitskii, O., Laclau, C., Maximov, Y., and Amini, M.-R. (2021). User preference and embedding learning with implicit feedback for recommender systems. *Data Mining Knowledge Discovery*, 35:568–592.

[Smyth and Cotter, 2000] Smyth, B. and Cotter, P. (2000). A personalised tv listings service for the digital tv age. *Knowledge-Based Systems*, 13(2):53–59.

[Spanoudes and Nguyen, 2017] Spanoudes, P. and Nguyen, T. (2017). Deep learning in customer churn prediction: Unsupervised feature learning on abstract company independent feature vectors. *CoRR: a computing research repository*, abs/1703.03869.

[Srivastava and Tanna, 2007] Srivastava, R. and Tanna, V. (2007). Double stage shrinkage testimator of the scale parameter of an exponential life model under general entropy loss function. *Communications in Statistics - Theory and Methods*, 36(2):283–295.

[Su and Khoshgoftaar, 2009] Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[Sun et al., 2019] Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. (2019). Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1441â1450, New York, NY, USA. Association for Computing Machinery.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

[Szegedy et al., 2014] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014). Intriguing properties of neural networks. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

[Tamaddoni et al., 2010] Tamaddoni, A., Sepehri, M., Teimourpour, B., and Choobdar, S. (2010). Modeling customer churn in a non-contractual setting: The case of telecommunications service providers. *Journal of Strategic Marketing*, 18:587–598.

[Tang and Wang, 2018] Tang, J. and Wang, K. (2018). Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, page 565â573, New York, NY, USA. Association for Computing Machinery.

[Teh et al., 2005] Teh, Y. W., Seeger, M., and Jordan, M. I. (2005). Semiparametric latent factor models. In Cowell, R. G. and Ghahramani, Z., editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, pages 333–340. PMLR. Reissued by PMLR on 30 March 2021.

[Torralba et al., 2008] Torralba, A., Fergus, R., and Weiss, Y. (2008). Small codes and large image databases for recognition. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[Ungar and Foster, 1998] Ungar, L. H. and Foster, D. P. (1998). Clustering methods for collaborative filtering. In *Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence (AAAI'98)*, pages 112–125, Madison, Wisconsin, USA. AAAI Press.

[Van den Poel and Lariviere, 2004] Van den Poel, D. and Lariviere, B. (2004). Customer attrition analysis for financial services using proportional hazard models. *European Journal of Operational Research*, 157(1):196–217.

[Varian, 1975] Varian, H. (1975). A bayesian approach to real estate assessment. *Studies in Bayesian Econometrics and Statistics in Honor of Leonard J. Savage*, pages 195–208.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

[Vincent et al., 2008] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103, New York, NY, USA. Association for Computing Machinery.

[Volkovs et al., 2017] Volkovs, M., Yu, G., and Poutanen, T. (2017). Dropoutnet: Addressing cold start in recommender systems. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[Wang and Zhang, 2013] Wang, J. and Zhang, Y. (2013). Opportunity model for e-commerce recommendation: Right product; right time. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 303–312, New York, NY, USA. Association for Computing Machinery.

[Werbos, 1974] Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, Massachusetts, USA.

[Wong, 2011] Wong, K. K. (2011). Using cox regression to model customer time to churn in the wireless telecommunications industry. *Journal of Targeting, Measurement and Analysis for Marketing*, 19:37–43.

[Wu et al., 2017] Wu, C.-Y., Ahmed, A., Beutel, A., Smola, A. J., and Jing, H. (2017). Recurrent recommender networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 495–503, New York, USA. Association for Computing Machinery.

[Zellner, 1986] Zellner, A. (1986). Bayesian estimation and prediction using asymmetric loss functions. *Journal of the American Statistical Association*, 81(394):446–451.

[Zhang et al., 2019] Zhang, S., Yao, L., Sun, A., and Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys*, 52(1).

[Zhao et al., 2014] Zhao, G., Lee, M. L., and Wynne, H. (2014). Utilizing purchase intervals in latent clusters for product recommendation. In *Proceedings of the 8th Workshop on Social Network Mining and Analysis*, SNAKDD'14, New York, NY, USA. Association for Computing Machinery.

[Zhou et al., 2020] Zhou, K., Wang, H., Zhao, W. X., Zhu, Y., Wang, S., Zhang, F., Wang, Z., and Wen, J.-R. (2020). S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information &amp Knowledge Management*. ACM.

[Zhou et al., 2015] Zhou, X., He, J., Huang, G., and Zhang, Y. (2015). Svd-based incremental approaches for recommender systems. *Journal of Computer and System Sciences*, 81(4):717–733.