EXAMENSARBETE INOM DATATEKNIK,
AVANCERAD NIVÅ, 30 HP
STOCKHOLM, SVERIGE 2017

Predicting Customer Churn Using Recurrent Neural Networks

JESPER LJUNGEHED

KTH
SKOLAN FÖR DATAVETENSKAP OCH KOMMUNIKATION

Predicting Customer Churn Using Recurrent Neural Networks

JESPER LJUNGEHED

Master in Computer Science
Date: June 6, 2017
Supervisor: Kevin Smith
Examiner: Viggo Kann
Swedish title: Predikera kundbeteende genom användning av återkommande neurala nätverk
School of Computer Science and Communication

Abstract

Churn prediction is used to identify customers that are becoming less loyal and is an important tool for companies that want to stay competitive in a rapidly growing market. In retail, a dynamic definition of churn is needed to identify churners correctly. Customer Lifetime Value (CLV) is the monetary value of a customer relationship. No change in CLV for a given customer indicates a decrease in loyalty.

This thesis proposes a novel approach to churn prediction. The proposed model uses a Recurrent Neural Network to identify churners based on Customer Lifetime Value time series regression. The results show that the model performs better than random. This thesis also investigated the use of the K-means algorithm as a replacement to a rule-extraction algorithm. The K-means algorithm contributed to a more comprehensive analytical context regarding the churn prediction of the proposed model.

Sammanfattning

Illojalitet predikering används för att identifiera kunder som är påväg att bli mindre lojala och är ett hjälpsamt verktyg för att ett företag ska kunna driva en konkurrenskraftig verksamhet. I detaljhandel behöves en dynamisk definition of illojalitet för att korrekt kunna identifera illojala kunder. Kundens livstidsvärde är ett mått på monetärt värde av en kundrelation. En avstannad förändring av detta värde indikerar en minskning av kundens lojalitet.

Denna rapport föreslår en ny metod för att utföra illojalitet predikering. Den föreslagna metoden består av ett återkommande neuralt nätverk som används för att identifiera illojalitet hos kunder genom att predikera kunders livstidsvärde. Resultaten visar att den föreslagna modellen presterar bättre jämfört med slumpmässig metod. Rapporten undersöker också användningen av en k-medelvärdesalgoritm som ett substitut för en regelextraktionsalgoritm. K-medelsalgoritm bidrog till en mer omfattande analys av illojalitet predikeringen.

Contents

Glossary

Churner For a company, a churner is a customer that is becoming less loyal. The customer is considered lost.

Churn Prediction Churn prediction is a process to identify customers that are prone to churn.

CLV Customer Lifetime Value (CLV) is a concept used in marketing to describe the future value of a customer.

CRM Customer Relationship Management (CRM) is an data analysis driven approach for managing customer interaction.

FNN Feedforward Neural Network (FNN) is the most basic neural network.

Recency, Frequency, Monetary Recency, Frequency, Monetary (RFM) is a method used by businesses to analyse customer behaviour and value, by analysing recency of purchases, frequency of purchases and amount of purchases.

RNN Recurrent Neural Network (RNN) is a type of neural network which contain loops to persist information.

ROC Receiver Operating Characteristic (ROC) curve is a graph that plot the true positive rate against the false postitive rate in regards to a varying discrimination threshold.

Chapter 1

Introduction

The most valuable asset to a company is their customer base [1]. Consequently, Customer Relationship Management (CRM) is an important task conducted by companies [2]. CRM is a strategic approach to customer relationship management that aims to create profitable, long-term customer relationships through the use of information technology [3, 4]. One practice of CRM is customer retention. By identifying and understanding valuable customer segments, the appropriate marketing strategies can be adopted to boost customer satisfaction and maintain customer loyalty, increasing the company's retention rate [5]. With a high retention rate, the business is more likely to prosper. The growing interest for big data mining techniques and the fact that companies often manage a large transaction database with millions of data points has led to increased attention to use churn prediction as a means of increasing the customer retention from companies that seek to gain a competitive advantage. In marketing analytics, the concept of churn prediction is to use machine learning and predictive modeling to detect customers who are most likely to cancel a subscription or stop their purchases in favour for another retailer. These customers are referred to as churners.

A problem that arises when performing churn prediction is the definition of a churner. Typically, the definition of churn is based on product usage and some customer independent explicit threshold. Furthermore, the definition of a churner is usually domain specific, often even company specific. In a subscription based service, defining a churner is relatively easy. For example, when performing churn prediction for a mobile phone service company, a churner is typically defined as a customer with a cancelled subscription. Van den Poel and Lariviere [6] defined a churner as a bank customer with all accounts closed. However, the use of an explicit threshold is difficult to apply to a non-contractual setting. In retailing, customers usually have different behaviours and therefore, each customer should have a different definition of churn. For retailers, a customer independent definition of churn does not reflect reality well.

In light of this, several papers have investigated the use of Customer Lifetime Value (CLV) as metric for churn. CLV is the monetary value of a customer relationship, based on the cash inflow and outflow of that customer relationship. CLV is calculated individually for each customer, which offers a more dynamic approach to the definition of churn. Neslin et al. [7] provide a comparison of several churn predictors that use change in CLV as the definition of churn.

Different machine learning algorithms applied to churn prediction involve classification trees [8], neural networks [9], SVM [10] and sequential patterns [11]. One rather

unexplored approach to churn prediction is the use of Recurrent Neural Network (RNN). RNN is a type of neural network that, simply put, has memory capacity. Characteristics of RNNs which makes them applicable for time series prediction are that RNNs tend to be robust to temporal noise and are suitable for sequential input [12, 13]. One drawback of neural networks is that they are considered black box models. The black box issue makes it difficult to gain any further insight of the function that is being approximated by a neural network by studying the structure and weights of the network. The black box nature becomes critical in the context of analysis, as it is difficult to determine how each of the input variables contribute to the output of the network.

1.1 Research Question

This paper contributes to existing literature by exploring the potential of using RNN for predicting churn based on CLV time series. For marketing analysts, it is vital to understand their customers. In churn prediction, one important aspect is to identify the key drivers of churn. To overcome the black box drawback of RNNs, the model was used in combination with the K-means algorithm. The K-means was used to perform customer segmentation. The result from the churn prediction was then analysed in the context of the clustering labels. If the customer segmentation is meaningful (each segment represents a customer group, like "High spenders"), more conclusions may be drawn from the predictions, which provide a more comprehensive analytical view of the outcome.

The following research question is investigated throughout this paper: *How does RNN in combination with CLV time series perform when classifying potential churners in a non-contractual setting? Specifically, how does it perform when compared to randomness?*

This paper also investigates if clustering customers into segments helps to identify customer groups that are more likely to churn.

1.2 Limitations

The purpose of this paper was not to compare different ways of modeling CLV, or to provide a new definition of CLV.

Only one data set was used for the experiments, therefore, conclusions drawn might only be applicable to the particular data set.

K-means was used to cluster customers based on RFM attributes. Other clustering algorithms were considered. However, due to a limited time frame, implementing and testing more than one algorithm was considered out of scope. Only a limited number of RFM attributes were used for the clustering.

Only one type of RNN was investigated, the LSTM. The time spent optimising hyperparameters of the RNN was limited and therefore the RNN might not have been optimal.

1.3 Structure of the Paper

The rest of this paper is organised as follows: in Chapter 2, the theoretical framework of this thesis is presented. Relevant theory is presented in this chapter. In Chapter 3, the chosen method and implementation details are described. In Chapter 4, the result of the

conducted case study is presented, and in Chapter 5 conclusions drawn from the result is presented. This thesis concludes in Chapter 5 with a review of the findings and contributions. Suggestions for future work as well as a critical review of the thesis is presented in this chapter.

Chapter 2

Theoretical Framework

This chapter presents the relevant theory and intends to motivate why certain methods and concepts were chosen in favour for others, with respect to limitations, the data set, and the proposed model.

2.1 Terminology

A data point refers to a vector that may contain many different attributes.
An attribute, or feature, refers to a field in the vector, and for a customer, an attribute may be age or days since last purchase.
The dimension of a data point is equal to the number of attributes in the vector.

2.2 Definition of Churn

Churn prediction is a process that identifies customers that are prone to churn. To perform churn prediction, a churned customer has to be defined. When performing churn prediction in, for example, a mobile phone service context, there exist subscriptions, often called a contractual context. In a contractual context, it is relatively easy to define a churner; a customer that has cancelled their subscription. For retailers, there exists no subscription, making it a non-contractual context. This section discusses the difficulty of defining a churner in a non-contractual context.

Typically, when performing churn prediction in a retail setting, the data set consists of transaction history and no subscriptions. The challenge with a non-contractual context is that the predictor needs to differentiate between a customer who has completely stopped and a customer who is just in the midst of a long gap between transactions. This becomes troublesome if the definition of churn is explicitly defined, such as "if customer X has not bought anything for 30 days, the customer is churned". A scenario where this could go wrong is if customer A usually has 10 purchase occasions a year, and customer B has 100 purchase occasions a year. If a churner is defined as someone with less than 10 purchase occasions a year, customer A will be defined as a churner if his purchase occasions drop with 1, while customer B is still considered as a loyal customer if his purchase occasions go down to 11. Clearly, the loyalty of customer B has dropped significantly, as opposed to customer A.

To overcome the problem of an explicit definition of churn, Baesens et al. [14] proposed a more dynamic approach using a Bayesian network to estimate the slope of the

customer life cycle in a non-contractual setting. The lifecycle was based on the past product purchases of each customer, and a predicted negative slope represented a decrease in future customer spending. Thus, each customer had a relative definition of churn. Glady et al. [15] built upon this work. They used CLV as the definition of churn.

2.3 Customer Lifetime Value

CLV is a metric used to evaluate customers. For each customer, CLV is the monetary value based on the cash flow associated with that customer relationship. This section explains the difficulties regarding the definition of CLV and how it can be calculated.

There has been much research on how to properly define CLV, especially the meaning of the word "value" [16]. In their overview, Pfeifer et al. [16] begins by defining the term cash flow. The cash flow of a customer relationship is the accumulated value of cash inflow and outflow related to that specific customer relationship. The inflow of a customer relationship could be the money the customer spends in a particular department, and the outflow could be the total cost of marketing campaigns needed to retain the customer. By letting the user specify what constitute to the cash flow of a customer relationship, the definition of CLV can be more relaxed.

A problem that arises is that the CLV is a current value calculated from the past cash flow of that particular customer. Due to this, CLV has to be properly defined to be in compliance with the fundamental principles of finance, such as the discount rate. The discount rate is the interest rate used in discounted cash flow [17]. Pfeifer et al. [16] defined CLV as ".. the present value of future cash flows attributed to the customer relationship". They argue that by doing so, they are consistent with CLV definitions that account for "the time value of money".

Usually, CLV is modeled as a function of the future contribution and the future cost, such as retention actions and marketing costs. The function can simply be described as follows:

$$CLV_i = \sum_{t=1}^{h} \frac{\text{future contribution}_{it} - \text{future cost}_{it}}{(1+r)^t} \tag{2.1}$$

where

i = customer index,
t = time index,
h = time horizon,
r = discount rate.

However, in the literature, there exists several different methods and approaches for modeling CLV.

Berger and Nasr [18] proposed an equation for CLV. This equation was based on various assumptions, such as (1) one sale occasion per year and (2) the cash flow per year of each customer is constant. These assumptions are too restrictive for the setting in this thesis since members in the data set have several purchase occasions per month, some even per week. In the same paper, more complex equations were presented, which relax assumption (1) by considering periods other than one year. However, this model does not include the acquisition cost of a member.

Gupta et al. [19] presented a function for modeling CLV that included the acquisition cost of a member, estimating it as the total marketing costs divided by the number of

acquired customers for that period.

Glady et al. [15] proposed an approach to modeling CLV, with application to churn prediction, which considered the profits rather than the total cash flow associated with the customer.

Finally, for a more detailed overview, Gupta et al. [20] described six approaches for modeling CLV.

In this thesis, CLV will be a modeled as described by Glady et al. [15], defined by Equation 2.2. Thus, the CLV of the customer i over the time horizon h, for the period t, is the sum of the cash flow $CF_{i,j,t+k}$, yielded by the transaction of item j among the q items.

$$CLV_{i,t} = \sum_{k=1}^{h} \sum_{j=1}^{q} \frac{1}{(1+r)^k} CF_{i,j,t+k} \tag{2.2}$$

Glady et al. [15] defined cash flow to account for the cost of retention actions in a non-contractual setting. In so doing, they argue that their model consider customers that are more profitable to act on. The approach to cash flow in this thesis considers the total cash inflow, rather than the profits. However, Pfeifer et al. [16] argue that if "value in CLV is to match the value in present value, and if the definition of CLV is to be consistent with the fundamental principles of finance, then CLV should be the present value of the cash inflows and outflows" and not profits.

The cash flow is defined as a function of the gross price π_j and the item usage of item j of customer i during period i, $\phi_{i,j,t}$:

$$CF_{i,j,t} = \pi_j \phi_{i,j,t}. \tag{2.3}$$

By making the gross price π_j the actual price that the member paid for an item (item discount included), the cash flow is considered to include retention costs to some extent

Using Equation 2.2 and Equation 2.3, the final definition of the CLV for the customer i at time t over the q item is:

$$CLV_{i,t} = \sum_{k=1}^{h} \sum_{j=1}^{q} \frac{1}{(1+r)^k} \pi_j \phi_{i,j,t+k}. \tag{2.4}$$

Note that the CLV function does not consider the acquisition cost of each member, as this cost is not present in the data set. However, including the acquisition cost in the function implicitly models CLV of an as-yet-to-be-acquired customer [20]. Thus, not including the acquisition cost of a customer is not considered a limitation, since this thesis aims to predict churn among existing customers.

2.4 Customer Segmentation by Recency, Frequency, Monetary Attributes

The RFM model is a popular approach to customer segmentation that is based on past behaviour [21]. The attributes of RFM model can be used to find valuable customer segments, to find trend in spending and to predict future customer behaviour [22]. The segmentation based on RFM attributes can also be used by marketers to create marketing campaigns directed to identified segments.

In retail, the three components of RFM are usually defined as (1) Recency - the time passed since last purchase occasion; (2) Frequency - the number of purchase occasions; (3) Monetary - total amount spent on purchase occasions [23]. There exists slightly different definitions of these components as well as extensions. For example, Marcus [24] defines monetary as the average purchase amount per transaction. A common extension is the number of the purchases per product category, which effectively is the same as grouping the customers according to what they purchase.

Alternatives to RFM analysis for customer segmentation are CHAID and logic regression [25]. Both methods can be used to create marketing lists, but require customer interaction. The marketing lists are created based upon sampling campaigns, where a few customers are selected and based on their response, necessary steps are done to construct the respective algorithms.

2.5 Clustering Techniques for Customer Segmentation

This section aims to introduce common data mining techniques used for customer segmentation. The intention of segmenting customers was to overcome the drawback of RNN being a "black box" model. By finding meaningful clusters, the intention was that analysts might be able to find patterns among churners and nonchurners.

2.5.1 Distance Functions

Clustering involves calculating the similarity of two vectors. The similarity between two vectors, $\vec{x}$ and $\vec{y}$, is calculated using a distance function $d(\vec{x}, \vec{y})$. In literature, several different distance functions have been proposed [26]. This section give a brief introduction to common distance functions.

For continuous numerical values, the most commonly used distance function is the Euclidean distance, defined in Equation 2.5, which is an application of Minkowski distance. Manhattan distance, is defined as seen in Equation 2.6 and is another application of the Minkowski distance.

$$d_{Euclid}(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (\vec{x}_i - \vec{y}_i)^2} \tag{2.5}$$

$$d_{Manhattan}(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |\vec{x}_i - \vec{y}_i| \tag{2.6}$$

For categorical attributes, the most used similarity measure is the Jaccard index [27]. Another distance function is the Mahalanobis distance [28], as defined by Equation 2.7. This distance function has been modified to work with mixed variables [29]. The Mahalanobis distance is more computationally extensive than the Euclidean distance [30].

$$d_{Mahalanobis}(\hat{x}, \hat{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})} \tag{2.7}$$

where S is the covariance matrix between data points.

The similarity, or the dissimilarity, between two vectors depend on the variance between features in the vectors. Features with larger values impose a higher variance. Normalisation methods are used to ensure that different variables are in the same scale prior

to clustering, to minimise that some indicators may have too much impact on the clustering result. Two of the most common methods for normalisation is the z-score and the min-max value conversion. For attribute x, the min-max approach is defined the following way:

$$\hat{x}_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{2.8}$$

where $\hat{x}_i$ is the normalised value, ranging between 0 and 1, of attribute x_i. The calculation of the z-score of a attribute is defined as follows:

$$\hat{x}_i = \frac{x_i - \text{mean}(x)}{\sigma(x)} \tag{2.9}$$

where the resulting $\hat{x}_i$ have a mean of 0 and a standard deviation of 1.

2.5.2 Clustering Models

The goal of a clustering model is, given a set of objects, to find groups in a way such that each object in the same group has a high measure of similarity, and the similarity between objects in different groups are low.

Clustering models are one of the most used tools for understanding and interpreting data when no prior knowledge of the structure or underlying distribution is present. Clustering models ability to uncover hidden structures in big data, broad range of application and empirical success are the main reasons for their popularity [31]. Clustering is widely used in areas such as image segmentation, information retrieval, bioinformatics and customer segmentation [32]. Clustering has been applied as a method for organising data, creating topical hierarchies used to describe and access the data [33, 34], as well as to discover the natural structure of data [35].

Clustering models may be divided into two broad categories; partitioning and hierarchical. Hierarchical clustering recursively constructs the clusters in either a top-down or a bottom-up way [36]. The top-down approach is also known as divisive hierarchical clustering. In this method, all data points belong to one cluster, and then the algorithm recursively splits each cluster into smaller clusters. The bottom-up approach, called agglomerative hierarchical clustering, starts with one cluster for each data point and then recursively merges pairs of clusters. The decision of which pairs of clusters to merge or split is based on a linkage criterion. Single-link and complete-link are the two most popular linkage criterion [32]. For single-link, the distance between two clusters is defined as the minimum distance between data points in each cluster. The complete-link criterion defines the distance between two clusters as the maximum distance between data points in each cluster. The splitting and merges of the clusters impose a dendrogram like hierarchy of the clusters (as seen in Figure 2.1), creating multiple partitions. If the final dendrogram is persisted some way, the clusters can be manipulated. For example, if one cluster is considered meaningless or too small, traverse the dendrogram and choose a different partitioning. In a best case scenario, the time complexity of a hierarchical model is $O(n^2)$ where n is the number of data points, making it impractical for large data sets [36].

Partitioning methods start from an initial partitioning. The data points are then relocated between clusters to minimise some error criterion. In contrast to hierarchical clustering, partitioning methods only produce one partitioning of the data points, and usually require the user to specify the number of clusters. The most commonly used parti-

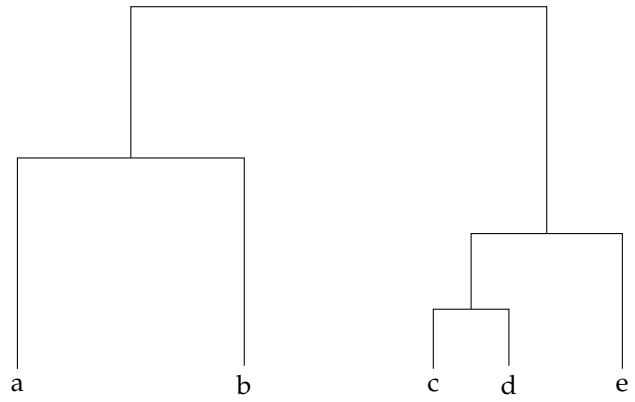

Figure 2.1: Dendrogram representing a hierarchical clustering. All data points started in one cluster, indicating a top-down approach has been utilised. In this example, applying a cut after the lowest row of the dendrogram would yield clusters {a,b} {c,d} {e}. Performing a cut after the first row would yield clusters {a,b} {c,d,e}.

tioning algorithm is K-means [37]. The popularity towards K-means can be explained by its simplicity, its simple implementation and relative efficiency.

2.5.3 Clustering with K-means

The K-means algorithm [38–40] is a simple clustering algorithm. As the name implies, the goal of the algorithm is to find K subsets, clusters, of a data set. The clusters are represented by K centroids, which is the mean or weighted average of the data points in the cluster. The problem is to partition the data points in such a way that the squared error between the centroid and the data points in the cluster are optimised. Thus, the ultimate goal of K-means is to minimise the sum of squared errors over all K clusters, which has been shown to be NP-hard [41]. As such, implementations of K-means converge to local optima. This local search consists of the four steps shown in Figure 2.2. For each iter-

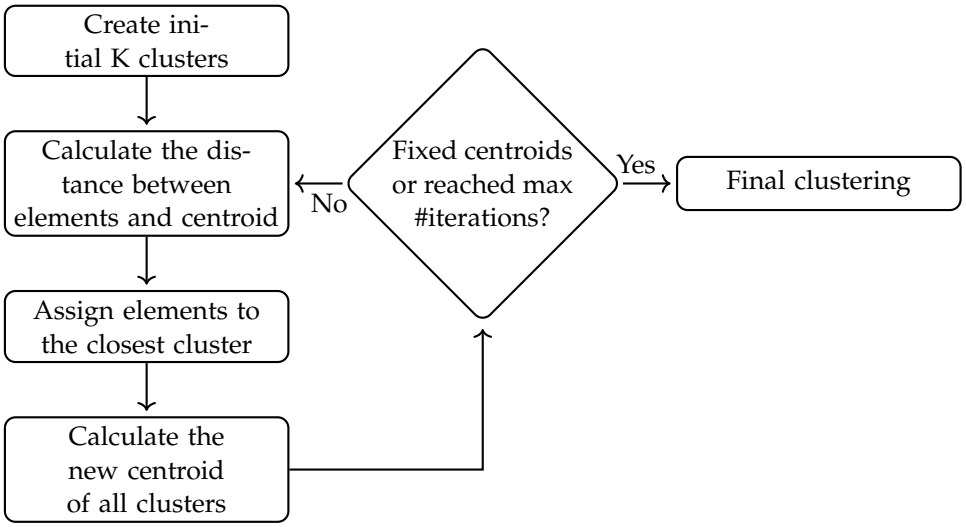

Figure 2.2: Flowchart of the K-means algorithm

ation, $K \cdot n$ comparisons are needed, where n is the number of data points. The number of iterations needed until the algorithm converge (i.e. no element change cluster) is

unknown, but typically a maximum of number iterations, t, is specified. Thus, the time complexity of the K-means algorithm is O(knt).

The K-means algorithm requires the user to specify three parameters. These are the initialisation of the clusters, the number K and distance function. The initialisation of clusters could either be fixed or randomly generated. Different initialisations have the potential to reach different local optima. The most critical parameter is the choice of K. There exist various methods for choosing K, the simplest one being a guess by the user. Other methods employ heuristics [42], while some methods dynamically adjust K during execution to find the optimal K [43, 44].

One drawback of the K-means algorithm is that it does not work with categorical values. Typically, the distance function used to calculate the similarity between data points assumes numerical values. Categorical values can be seen as enumerations rather than numerical values, for which the similarity calculation is undefined. For example, the distance between "cat" and "dog" or "man" and "woman" is numerically undefined. It is possible to use binary indicators for categorical values, which is called indicator coding. For n categorical values, $n - 1$ dummy variables are introduced. The dummy variable that corresponds to a category is set to 1 for the records belonging to that category, and the rest is set to 0. The last category is represented by setting all dummy variables to 0. This method introduces several attributes which increases the dimensionality of each record, impacting the performance of the K-means algorithm. Also, categorical variables tend to dominate the structure of the final clustering solution, yielding biased solutions that overlook patterns in continuous variables, which can be explained by the high variance introduced when using binary indicators [45]. In the literature, there exist several other extensions to the K-means algorithm for handling categorical values, for example by changing distance function [46].

2.5.4 Clustering Evaluation

Cluster solutions should be examined and evaluated to assert meaningful clusters, which is a necessity for a meaningful analysis. The number of clusters and objects assigned to each cluster should be examined. Furthermore, a good cluster solution should have highly separated clusters where each object is concentrated around their assigned centroid. In literature, there exist several methods for this purpose, some of which will be discussed in the following sections.

A general measure for clustering evaluation is the sum of squared errors (SSE) as shown in Equation 2.10:

$$SSE = \sum_{t=1}^{n} \sum_{t=1}^{n} d_{Euclidean}(c_i, x)^2, \tag{2.10}$$

where c_i is the centroid of cluster i, and x is data point of cluster i. A low total SSE indicate that objects are close to their assigned centroid.

The compactness of clusters can be examined by calculating the standard deviation of each clustering field for each cluster. Typically, all clustering fields should have low standard deviation which indicates similar clusters with low dispersion. Furthermore, the maximum Euclidean distance from each centroid represents the cluster radius of each cluster.

The separation of clusters can be evaluated by identifying the minimum distance between cluster centroids. Pair of cluster centroids with a significantly lower distance may indicate similar clusters that may be merged.

In retail, clusters may be examined through profiling, by for example attributing the profile "High spenders" to one cluster, and the "Favours discounted items" profile to another cluster. This is done by calculating the means of the clustering fields for each cluster. The mean values are then compared to the overall population means and clustering fields, in an attempt to find clustering fields with mean values of a significant difference to the overall mean. Deviation from the overall mean may indicate a meaningful cluster. Further investigation may be done through the Student's t-test, to assure that the differences are statistically significant.

2.6 Feedforward Neural Network

The Feedforward Neural Network (FNN) is the most basic artificial neural network. The network consists of layers of neurons, where each neuron is a node with a complex, non-linear mapping function. The layers are organised as seen in Figure 2.3: the input layer, one or more hidden layers, and an output layer. A neural network can be many-to-many, one-to-many or many-to-one. The input layer contains the input, often called predictor variables in the context of classification.

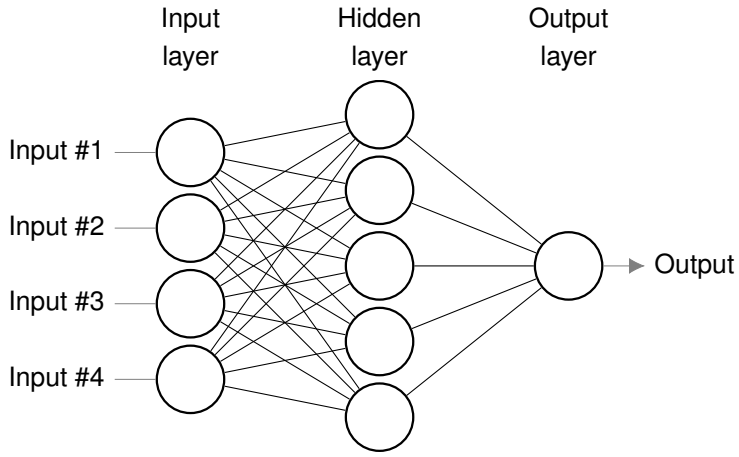

Figure 2.3: ANN with one hidden layer

The network is trained in an iterative manner to estimate the weights of the mapping functions in each node that connect the predictor variables to the output layer. The network is presented with the input series that contain the observed predictor variables, which are propagated forward in the network to the output layer where the network produces an output. To get a more accurate prediction, a gradient-based technique called backpropagation is used. The observed errors, calculated using a loss function that compares the output of the network with the expected output, are propagated backward in the network and used to adjust and optimise the weights of the mapping function of each neuron. This way, the network learns which predictor variables contribute to the correct output.

Neural networks are considered as black box models as they do not provide any direct explanation of the predictions made. In literature, several rule-extraction algorithms

have been developed to uncover the decisions made by a model [47]. Rule-extraction algorithms are only capable representing decisions made in terms of the input features to the neural network. Thus, all the features of significance have to be included as input to the neural network, increasing the computational needs.

2.7 Recurrent Neural Network

For years, RNN was considered difficult to train [13]. However, in the past few years, progress has been made, and RNNs have proven to be suitable for classification and generating sequences in domains such as natural language processing [48], speech recognition [49] and computer vision [50].

The RNN is a method of applying FNN. The RNN differs from the FNN in the aspect that the RNN contains loops. Whereas the FNN only consider the input at every timestep, the loops in the RNN allow information to be passed from one timestep to the next. This allows information to persist.

The loop in the RNN can be seen as a sequence of multiple copies of the same feed-forward neural network, unfolded through time, as seen in Figure 2.4. Each neural network represents a timestep of the input. Because of this, RNNs are especially applicable to sequences and time series.

The training procedure of RNN is called Backpropagation Through Time (BPTT) [51]. This gradient-based technique is an extension of the backpropagation used for FNNs, adjusted to account for the addition of timesteps. However, early adoptions of the RNN suffered from the vanishing gradient problem. The weights of the mapping function in each node is updated based on the gradient. Because of the many stages of multiplications that happen in a sequence of multiple neural networks, the derivatives calculated by the BPTT can be very small. This can lead to vanishing gradients too small for the RNN to learn.

The Long Short-Term Memory variant of the RNN was proposed by Hochreiter and Schmidhuber [52] as a solution to the vanishing gradient problem. The LSTM is the most used variant of the RNN. Most notable applications of RNNs are achieved using the LSTM [53]. The LSTM includes a memory cell, which allows the network to remember through timesteps. The state of the cell is regulated by three gates, each with its set of weights. The input pass through each of the gates. The input gate decides if information should be written to the cell. The output gate decides if information should be read. The forget gate allows the cell to reset its state. The weights of each gate are adjusted by the BPTT algorithm. To preserve a constant error, which is essential for the BPTT to avoid vanishing gradients, the new cell state is decided by adding the current state with the new input instead of using multiplication. The output is then calculated based on the new cell state.

For more detailed discussions on the complexity of training and using RNNs, check the recommended papers [54, 13].

2.8 Techniques for Churn Prediction

Upon deciding CLV as the metric for churn, two approaches can be applied for identifying churners. This section gives a brief introduction to various techniques for CLV prediction. In both approaches,

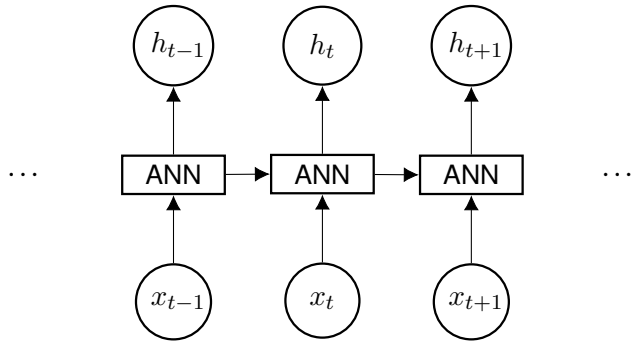

Figure 2.4: RNN represented as multiple FNNs

the input to the classifier is the same, namely CLV. The input may be be extended with other attributes, such as the RFM attributes.

2.8.1 Classification Model

The first approach is to train the predictive model as a classification model. Either a binary classifier or a probabilistic classifier can be used. For a given customer, the output of a binary classifier is whether that customer will stay or not, and for a probabilistic classifier, the output is the probability of whether that customer will stay. A probabilistic classifier can be converted into a binary classifier by introducing a threshold. One drawback of using a classification model for churn prediction is the class imbalance problem [55]. When one class is dominating the data set, which is the case when performing churn prediction, the training of the classification model becomes troublesome. The model is easily overfitted and almost all examples is classified as belonging to the dominating class.

2.8.2 Regression Model

The second approach is to let the model predict future CLV for each customer, referred to as a regression model. In this approach, the model is trained to predict a future continuous value based on previous values. Then, the continuous output can be handled the following way to perform churn prediction: Given the predicted CLV value at time t+1, divide it with the actual CLV value at time t, which gives the predicted change in CLV. The customers with no predicted change in CLV are then classified as churners. This way, the regression model is converted into a binary classifier. The performance of the model can be evaluated by comparing the predicted churners to the actual churners. In literature, there exist several different models for predicting future CLV.

Etzion et al. [56] proposed a model, based on Markov Chains, for predicting CLV. Specifically, the predictions were based on RFM attributes. The paper described a case study which showed that the proposed model performs well when predicting future CLV.

In contrast, Malthouse and Blattberg [57] developed regression and neural network models for predicting CLV based on RFM attributes. Their result was not satisfactory, and they claim that "historical value is not a very accurate predictor of future value". They argue that of the top 20% customers, 55% will be misclassified and not retrieve special treatment.

RNNs has been applied in many areas to perform time series prediction, such as financial forecasting [58] and electricity demand forecasting [59]. A time series is a sequence of data points ordered in time. As CLV is easily converted to time series, RNN is applicable in the context of churn prediction.

In the literature, RNN has not been used in combination with CLV time series.

2.8.3 Evaluating Binary Classifiers

If the accuracy of the predictions is low, conclusions drawn from the analysis is rendered useless. Several metrics can be used to evaluate the accuracy of a classifier. The rest of this section give a brief introduction to metrics commonly used to evaluate binary classifiers in a churn prediction context [8, 10, 60]. One class is labelled as positive (a churner) and the other one is labelled as negative (a non churner).

Precision

Precision is the fraction of actual churning customers among the customers classified as such.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} \cap \text{false positives}} \tag{2.11}$$

Recall

Recall, also referred to as sensitivity in a classification context, measures the fraction of the churning customers that are correctly identified as such.

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} \cap \text{false negatives}} \tag{2.12}$$

Receiver Operating Characteristic Curve

Receiver Operating Characteristic (ROC) is a plot that represents the performance of a probabilistic or a threshold based classifier turned into a binary classifier for a varying discrimination threshold. The ROC curve is defined by plotting the precision against the recall at different discrimination thresholds. This is useful when evaluating a classifier in the context of churn. If the result of the classifier is used to compile lists for targeted marketing campaigns, inspection of the graph will reveal how many of the potential churners that would be included for a given threshold. By increasing the threshold (assuming a non-perfect classifier was used to assign the probability to churn, where a perfect classifier would assign the highest probability to all actual churners), more potential churners will be targeted but will also lead to more false positives. Thus, the graph can be used to choose an appropriate threshold.

The area under the ROC curve is also commonly used to compare the performance of different classifiers. It represents the likelihood that the probability assigned to a randomly selected churner is higher than the probability assigned to a randomly selected non-churner.

The cumulative gain and cumulative lift table (gain and lift table) can be derived from the area under the ROC curve. To construct a gain and lift table, the customers

are split into deciles. The 10% of the customers with the highest churn propensities is present in decile 1. Decile 2 has the next 10% of the customers ordered by the churn propensities in descending order. The cumulative gain represents the percentage of churners that are included up to a given decile i:

$$\text{cumulative gain} = 100 * \frac{\sum_i c_i}{\text{total number of churners}} \tag{2.13}$$

where

c_i = number of churners in decile i.

It can be used to determine how many of the total customers a campaign has to target to include a certain percentage of the churners.

The cumulative lift, Equation 2.14, is calculated as the ratio between how many churners a classifier includes compared to a random sampling:

$$\text{cumulative lift} = \frac{\frac{\sum_i c_i}{\sum_i n_i}}{\text{C/N}} \tag{2.14}$$

where

c_i = number of churners in decile i,

n_i = total number of observations in decile i,

C = total number of churners,

N = total number of observations.

A random sampling of the customers is, given that 10% of the customers are targeted at random, it will contain 10% of the customers.

Chapter 3

Method

In this chapter, the chosen method is described. The data set used for the experiments is also described.

3.1 Data Set

The data set used in this study consisted of member data provided by a retail company and was collected through a customer loyalty program in a non-contractual setting. All the data was encrypted and anonymised by a third party, Omegapoint. The data set contained purchase history and the demographic information of each member. Each purchase occasion was either made online or in a physical store. Specifically, the data of 2,097,323 members were extracted from a three-year-long period spanning from January 2013 to January 2016. For a detailed description of the data set, see Appendix A.

3.2 Proposed model

The proposed prediction model was a Recurrent Neural Network, which was trained to predict CLV time series. Prior to prediction, the preprocessed data set was clustered using K-means. The process was divided into three steps: data preparation, clustering, and churn prediction.

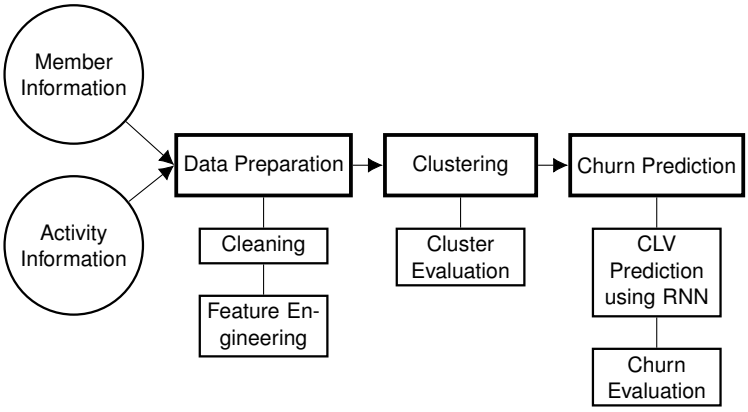

Figure 3.1: Flowchart of the proposed model

3.3 Data Preparation

The data preparation phase was done in two steps. In the cleaning step, some members were disregarded as unfit for the proposed method. The feature engineering step computed RFM features for the remaining members.

3.3.1 Cleaning

Members were removed according to two criteria. The first criterion removed members with less than six purchase occasions over the period spanned by the data set. These members are considered outliers. Outliers could potentially contain valuable information and should be investigated carefully. Investigations conducted by the company owning the data set has shown that their behaviour could be explained by the fact that some members only make purchases before Christmas or during Black Friday. This behaviour was not considered loyal in the first place, and as such was considered to be out of the scope of this analysis. Besides, the input for the RNN of these members would consist of too few data points, which would add no valuable information. Thus, these outliers were disregarded as data noise. The second criterion removed members that registered after the first six months of the data set, as this implied a new member, and only existing members were considered.

The cleaning was done for two reasons, the first being from an analytical aspect as explained above. Also, the complexity of the clustering and the RNN is linear to the number of members. After the cleaning had been applied, roughly 65% of the members were disabled, which reduced the computational need for the upcoming, more computational expensive, steps.

3.3.2 Feature Engineering

Using the purchase history, seven continuous RFM attributes were computed for each member (Table 3.1). Additionally, the days since registration for each member was extracted from the demographic information. Purchases per category and per location were disregarded. The addition of these attributes were considered unfeasible, as this would increase the dimensionality of the feature space, thus increasing the execution time and memory usage of the application.

Table 3.1: Table of RFM attributes calculated for each member

RFM Attribute	Mean	Standard Deviation
Days Since Last Purchase	0.1007	0.1162
Average Time Between Purchases	0.4458	0.2707
Total Amount	0.0286	0.0143
Total Discount	0.0149	0.0192
Total Amount Spent In Web Shop	0.0193	0.0685
Days Since Registration	0.8133	0.1622
Total Orders	0.0214	0.0258

3.4 Clustering

In this thesis, the goal of the clustering was to group a large set of customers according to similar behaviour. Due to hierarchical model's inability to scale well and K-means be-

ing more feasible and easier to implement, the chosen clustering algorithm was K-means.

The clustering was done on the previously computed RFM attributes shown in Table 3.1. The RFM attributes were normalised using the min-max method (Equation 2.8). The Euclidean distance was the chosen distance function, which is most common for K-means. Thus, the clustering was restricted to continuous values, and the discrete variables in the data set was disregarded. It should be noted that there were few categorical variables present in the data set that was considered useful for analysis. The most obvious one was gender. However, the variance of gender in the data set was low, and would add no further information for the clustering process.

The K-means process was repeated until it converged to local optima. Then, each member was labelled as belonging to one of the K clusters and persisted to the database. By randomising the centroid starts, the clustering was done in 10 iterations. This was done to overcome K-means being a local search algorithm. The values 5, 10, 15 and 20 for K were investigated. For each K, the cluster solution with the lowest sum of squared errors were further investigated. The cluster solutions were evaluated by constructing profiling tables.

The reason for the clustering was to uncover the black box issue of the RNN without the need of a rule-extraction algorithm. A rule-extraction algorithm would require the input to the RNN to be all of the features of interest, namely the RFM attributes, increasing the computational resources needed for the RNN. Thus, the intention of using the K-means was to be able to use the clustering labels from the clustering combined with the output of the RNN to analyse the members without adding significant computation.

3.5 Churn Prediction

3.5.1 CLV Calculation and Churn Labelling

CLV was computed for each member, according to the CLV definition (Equation 2.4) in Section 2.2. Discount rate r was set according to the Swedish Riksbank reference rate, which was 0 in the end of 2016. Therefore, the discount rate was set to the interest rate of SEK, even though purchases had been made in Euro and NOK. However, a more complex discount rate was considered out of scope for this thesis. The period t was set to each month during the time horizon h of three years. Thus, the CLV calculation resulted in CLV time series of length 36 for each member. Based on the last six months, a member was labelled as either a churner or a nonchurner. If the CLV at time step 30 was equal to the CLV at timestep 36, the member was considered churned. 29,597 members were labelled as actual churners, which corresponded to 4% of all the members.

Table 3.2: Customer Lifetime Value distribution

	Mean	**Standard deviation**
CLV	5,594	6,408

3.5.2 CLV Time Series Prediction using RNN

The input sequence to the RNN was the CLV time series of each member. The output sequence was the same length shifted forward by one time step. Thus, the RNN was

trained to predict the CLV value for the next time step. The time series were divided into three sets. The training set consisted of time steps 1 to 24. The validate set consisted of time steps 24 to 30 and was used to evaluate the training. The last six time steps were never presented as input to the RNN. These time steps are included in the test set and were used to evaluate the predictions.

The model was used to predict the CLV of each customer every 30 days for 180 days in the future. It should be noted that prior to applying a certain configuration of the RNN to the full data set, sanity checks were performed. The sanity check procedure was to overfit the network on a tiny subset of the data of 100 members.

3.5.3 Churn Evaluation

The output of the RNN was the predicted CLV $\hat{x}_t$ of each member for the time steps 30 to 36. To identify predicted churners, the predicted value $\hat{x}_{t=36}$ was compared with the actual CLV value at time $t = 30$. The ratio $\frac{\hat{x}_{t=36}}{x_{t=30}}$ was then treated as the predicted change in CLV. A discrimination threshold was used to label members as either churners or nonchurners. Members with a predicted CLV change lower than the threshold was labelled as a churner.

The RNN was evaluated according to two aspects. The first aspect is how well it performed in labelling churners. This was evaluated by the gain and lift table. The second aspect was measure of the spread of the predicted values. Specifically, the mean and the standard deviation of the predicted values were examined, as well as the maximum value and the minimum value. The measure of spread of the predicted values should be close to the original values.

3.6 Technical Implementation Details

3.6.1 Application Overview

The application was implemented using Java. The main framework was SpringBoot [61]. SpringBoot was used to create a stand-alone web app, used to trigger the different components in the model. The Hibernate API [62] was used for creating the database architecture, which allowed for easy expansion of member info, such as adding or removing one RFM attribute. It also handled the connection and queries to the database that stores the member data. A PostgreSQL database was used.

3.6.2 K-means algorithm

The K-means algorithm was a modified version of the one present in the open-source data mining library called SPMF [63]. Trove4J [64] was used to improve the performance of the K-means algorithm.

3.6.3 Recurrent Neural Network

The RNN was implemented in Java using DeepLearning4J, an open-source library that provides distributed deep learning for Java [65]. Due to a limited time frame, the initial hyperparameters were based on recommendations in [13, 54]. The final hyperparameter configuration that was used is shown in Table 3.3.

Table 3.3: Hyperparameter configuration proposed Model

Hyperparameter	Value
Updater	Nesterovs, momentum 0.8
Learning rate	0.001
Activation Function	ReLU
Recurrent Connections	15
Stacked RNNs	1

Chapter 4

Results

The results of the experiments are presented in this chapter. First, the result of the clustering is presented. Then, the result of the churn prediction is shown. Finally, the churn prediction is evaluated in regards to a clustering solution.

4.1 Clustering

In Table 4.1, average run time and sum of squared errors (SSE) statistics is shown when the K-means algorithm was performed using different K.

Table 4.1: Table over K-means runs. 9 iterations for each K was done.

K	Average Run Time (min)	SSE Mean	SSE Best
5	19	28,434	24,189
10	24	21,218	16,695
15	31	17,859	14,511
20	36	16,758	12,744

The clustering solutions with the lowest SSE (SSE best in Table 4.1 for each K was further evaluated. Table 4.2 shows the mean of the RFM attributes for each cluster when compared to the overall mean for K=10. In this table, a high value for days since registration indicates a new member. A lower value for average time between purchases indicates less days between purchases. A lower value for days since last purchases indicates less days since last purchase. Observing the table, some behaviour groups can be identified. For example, members of cluster 2 is frequent online shoppers, while members of cluster 8 spend around the same total amount, but in comparison, is more frequent in physical stores. Clusters 7 and 8 consist of very similar RFM attributes. However, members of cluster 7 was registered more recently.

The profiling tables for other K values is found in Appendix C.1.

Table 4.2: Table over the clustering fields compared to overall mean, K=10

	Members	Total Amount	Total Discount	Total Orders	Days Since Last Purchase	Days Since Registration	Average Time Between Purchases	Total Amount Web Shop
Cluster 1	100,738	71%	67%	8%	97%	112%	192%	48%
Cluster 2	15,023	158%	192%	181%	51%	103%	53%	1,950%
Cluster 3	28,611	69%	67%	8%	162%	29%	190%	29%
Cluster 4	117,215	79%	73%	35%	103%	111%	120%	50%
Cluster 5	0	0	0	0	0	0	0	0
Cluster 6	140,193	94%	86%	82%	84%	110%	74%	53%
Cluster 7	88,645	134%	148%	213%	42%	68%	42%	92%
Cluster 8	138,910	142%	147%	228%	36%	107%	37%	88%
Cluster 9	48,598	70%	69%	12%	396%	110%	181%	43%
Cluster 10	56,522	81%	78%	50%	116%	70%	103%	46%
Overall	734,490	0.0286	0.0149	0.0214	0.1007	0.8133	0.4458	0.0193

4.2 Recurrent Neural Network Performance

In this section, the result of the churn prediction is presented. Specifically, the performance of the RNN that performed best according to the evaluation criterion is presented. The evaluation of the other configurations is found in Appendix D.2.

In Table 4.3, the gain and lift for each decile are shown. One decile comprises of 73,490 members. In the top decile, the predictor identified 223% more churners when compared to random. Specifically, 25,840 of the churners were identified in the top decile.

In Table 4.4, the statistical results of the regression are shown, calculated at month 36 over the 734,490 members. Note that the absolute mean and standard deviation is close to the original distribution of the CLV (Table 3.2). Other configurations managed to get better prediction accuracy (gain and lift), but less spread in the predicted values.

Table 4.3: Gain and lift table

Decile	Gain (%)	Lift (%)
1	22.26	223
2	39.33	197
3	54.28	181
4	66.93	167
5	77.08	154
6	85.13	142
7	90.99	130
8	95.08	119
9	97.85	109
10	100.00	100

Table 4.4: Prediction statistics

Max	Min	Mean	σ
225,748	-6,152	-816	4,975

In Figure 4.1, a graphical plot representing the gain when compared to random is shown. The closer the predicted curve is to the upper left corner of the graph, the better.

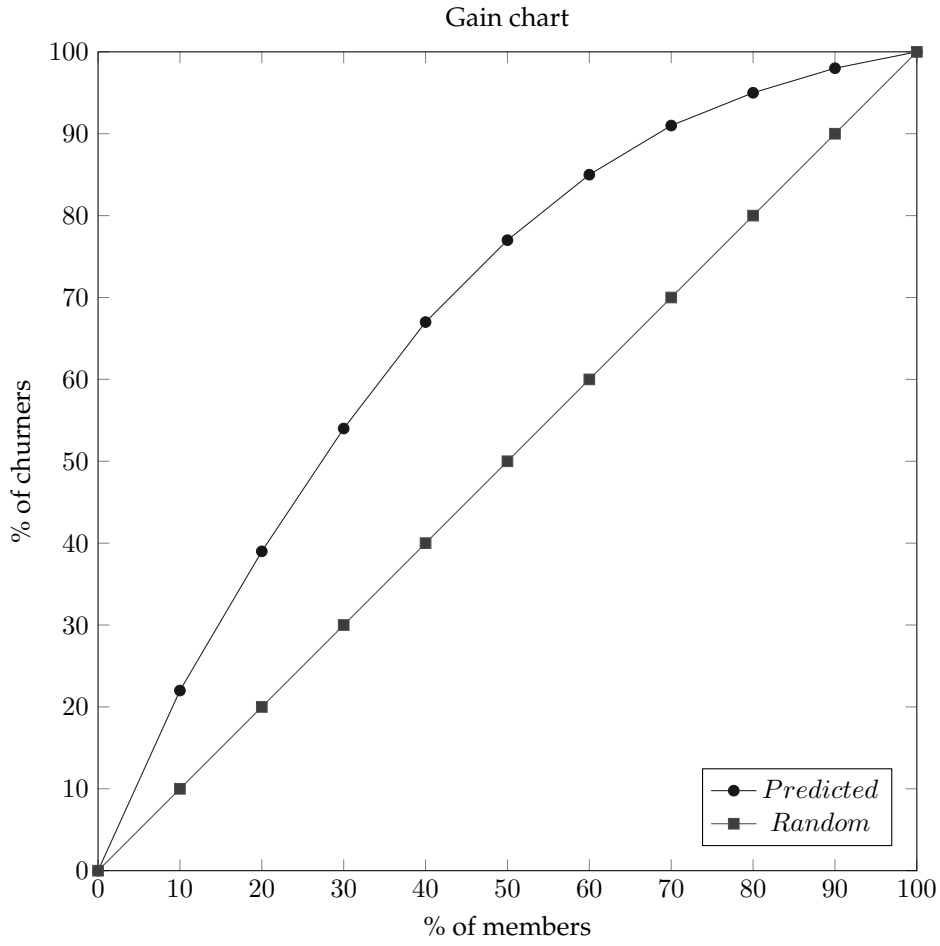

Figure 4.1: Gain chart for the proposed model

4.3 Combined Clustering and Churn Prediction

This section presents the results when the K-means algorithm, for K=10, was combined with the results of the churn prediction. K=10 was chosen after comparing profiling tables for different K values. In Table 4.5 and Table 4.6, cluster 5 is excluded because no members were assigned to this cluster by the K-means algorithm.

Table 4.5: Identified actual churners per decile for each cluster

Decile	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 6	Cluster 7	Cluster 8	Cluster 9	Cluster 10
1	4,806	0	4,252	1,660	123	2	0	14,197	800
2	2,735	0	2,315	2,810	442	5	0	10,112	1,391
3	1,773	2	1,405	3,290	942	26	6	8,199	1,708
4	1,153	1	844	3,118	1,522	54	7	6,133	1,860
5	664	10	435	2,668	1,920	72	24	4,322	1,648
6	350	29	225	2,153	2,241	175	56	2,736	1,378
7	172	81	104	1,484	2,174	232	99	1,494	963
8	90	103	51	876	1,825	241	173	825	568
9	24	144	28	433	1,453	215	235	367	311
10	11	193	10	178	1,048	233	460	213	152

In Table 4.5, the actual churners per decile are shown, grouped by assigned cluster and ordered by highest churn propensity according to the output of the RNN. Mem-

Table 4.6: Nonchurners per decile for each cluster

Decile	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 6	Cluster 7	Cluster 8	Cluster 9	Cluster 10
1	28,906	2	7,616	7,682	739	46	19	0	2,599
2	22,643	2	5,039	16,002	3,722	298	87	0	5,846
3	15,974	26	2,955	19,127	9,042	949	398	0	7,627
4	10,237	99	1,777	18,356	16,028	2,464	1,526	0	8,270
5	6,034	363	878	15,353	21,793	4,970	4,499	0	7,781
6	3,057	847	407	10,930	24,074	8,363	10,250	0	6,178
7	1,360	1,673	173	6,432	22,227	12,312	18,378	0	4,091
8	530	2,505	67	3,152	16,281	15,936	28,050	0	2,176
9	179	3,701	23	1,187	9,248	19,351	35,662	0	888
10	52	5,242	9	324	3,349	22,707	38,981	0	287

bers of clusters with low total orders mean and more days since last purchase (clusters 1, 3 and 9) tend to dominate the first number of deciles. The RNN failed to identify actual churners from clusters 2, 7 and 8. These clusters are characterised by a high total orders mean, a high total amount and discount mean, and a low average time between purchases.

In Table 4.6, the number of nonchurners per decile is shown, grouped by assigned cluster and ordered by highest churn propensity. The trends found in this table are consistent with the trends found in Table 4.5. The members of cluster 1 and 3 tend to dominate the first deciles, while members of clusters 2, 7, 8 tend to dominate the later deciles.

Chapter 5

Discussion

In this section, the thesis is discussed from several perspectives. The expected and unexpected consequences of working with a large data set are discussed in this chapter. The chosen method is reviewed, and finally, the ethical aspects is discussed.

5.1 Working with Large Data Sets

The data set included over 2 million members and tens of millions transactions. Thus, it was expected to be a time-consuming and compute-intensive task to perform the experiments. The first apparent obstacle was the amount of RAM required to keep all members in memory at the same time. Therefore, all components were designed to process batches. The size of the batches was set to 30,000, mainly because of a limit of the PostgreSQL JDBC driver, which restrict the number of parameters in a query to a 2-byte integer (32,767). The rest of this chapter discuss component-specific issues and solutions.

The reason for the optimisations was to achieve reasonable execution time when the experiments were performed, not to create a state-of-the-art application. Without the optimisations, the experiments were considered unfeasible due to the time frame of this thesis. Therefore, the optimisations might not have been optimal.

5.1.1 PostgreSQL Database

All the pre-processing and the preparation of the RFM attributes were calculated issuing database queries. The first naive approach was to use the PostgresSQL database as setup by the Amazon RDS. Initially, calculating the number of total orders for members took 37 hours. Note that this time was for one batch of 30,000 members on a server with 16GB of RAM and four cores. Also, persisting members were a very time-consuming task. Several optimisations were introduced iteratively to reduce this time:

- Appropriate indexes
- Optimised queries
- PostgreSQL configuration parameters (see Appendix B.1)
- Bulk Updates
- Disabled Hibernate Cache

In the final solution, the pre-processing of all the 2 million members were done in under 20 minutes.

5.1.2 K-means Algorithm

The K-means algorithm was modified to be efficient when applied to a large data set. The clustering algorithm was designed to process 30,000 members at a time. Another major difference was that the cluster class was optimised not to store the full member instances. Instead, it just stored the member keys of the members in the cluster, lowering the amount of RAM required. Trove4J's THashSet was used for this purpose. Using this approach, the algorithm would be required to iterate over all member keys in the cluster and fetch the rows containing the RFM attribute when recomputing the cluster mean. However, the SPMF implementation of the K-means algorithm was already modified to efficiently recompute the mean for each cluster. In addition to the HashSet storing the member keys of the members present in the cluster, an array of doubles was added. Upon adding a new member vector to a cluster, the RFM attributes were added to the array. When recomputing the mean, each of the values in the vector was divided by the number of members assigned to the cluster.

5.1.3 Training and Tuning a Recurrent Neural Network

DeepLearning4J offers highly modifiable neural networks. The RNN used the CUDA backend provided by the library. The training and testing were done on an Amazon EC2 p2.xlarge instance. The p2.xlarge instance offers one Tesla K80, four virtual CPUs and 62GB of RAM. Despite this, running the experiments took several hours. Therefore, most of the settings of the RNN were set according to recommendations in [13, 54]. The following hyperparameters were experimented with:

- Learning rate
- Stochastic Gradient Descent updater
- Number of hidden nodes
- Activation function
- Depth of the network (number of stacked LSTMs)

The optimisation of the hyperparameters was a time-consuming task. Configurations that showed promising validation performance were trained for 48 to 72 hours. It should be noted that the RNN used in the experiments might not have been optimal. See Appendix D.1 for different configurations that was investigated.

5.2 Discussion of Churn Prediction and Churn Labelling

The testing was designed to be analogue with a real world situation. That is, the predictor was trained on all members for a given time period to predict the following months. Thus, cross validation could not be applied. However, to really evaluate a churn predictor it should be tested in a real world situation, why this was the chosen evaluation method. Also, an RNN is more applicable to sequences, in this case time series. When

presented with a time series, the internal state of the RNN is stored for this particular member, and the next months are predicted based on the stored state. This characteristic of RNN would not be fully utilised if cross validation was used.

Furthermore, it should be noted that the training of an RNN and using it for prediction is time-consuming tasks. Therefore, this approach is only applicable in a real world situation where a marketing campaign list has to be compiled every 15 or 30 days and is less applicable in a real-time application. However, the state of the RNN can be saved, which would allow for on-demand predictions and further training when new data is available.

In retail, the shopping behaviour is regarded as recurrent on a yearly basis, influenced by seasonal events, such as clearance sales. Some members in the data set might only purchase before Christmas or during some particular sale. Therefore, it might be argued that a member should be labelled as either a churner or nonchurner based on one year of purchase history. However, as members who had fewer than six purchase occasions were disregarded in this thesis, the expected average time between purchases is less or equal to six months. Furthermore, it should be noted that the half year of data used for churn labelling included Christmas.

5.3 Discussion of Churn Prediction Evaluation

The churn prediction was evaluated according to two aspects, namely accuracy in churn prediction and the measure of spread of the predicted regression values. As the actual CLV had a deviation of 6,408, a model with significantly lower deviation was disregarded regardless of accuracy in churn prediction. The regression performance of such a model was considered unreliable. The accuracy of the churn prediction, the percentage of correctly classified churners for a given threshold, could be high even if the measure of spread of the predicted values was low. The problem with a low spread is if all members have a predicted CLV value close to x, and this particular value x results in a CLV ratio for members such that the labelled churners get the lowest CLV ratio, the model seems to perform well when in fact it fails to identify the trend. Therefore, an RNN configuration that results in a higher spread of the predicted regression values is more preferable. If the measure of spread is high, i.e. close to the actual deviation and mean, and the accuracy is high, it indicates the RNN managed to identify the trend among members.

5.4 Discussion of Combined Clustering and Churn Prediction

The combination of the clustering and the churn prediction showed that the proposed model was able to identify the different trends of the members. However, what can be derived from Table 4.5 and Table 4.6 is that the RNN failed to identify members in a certain cluster that deviate from the particular trend. This could be explained by the fact that the members with high RFM attributes are considered loyal members and only a few of them were found to be churners. Therefore, members with similar trend are treated the same way by the RNN. The dominating number of nonchurners of these particular clusters suppress the signals of the actual churners. This is problematic because it leads to a class imbalance problem for a specific cluster with a dominating trend, which from now on is referred to as a trend group. If this trend group is considered to be a loyal group, it is expected to consists of a low amount of churners. To further enhance

the performance of the proposed model, the deviations from the trend should be carefully considered. This could be done by training a model for a particular trend group with the class imbalance problem taken into consideration, e.g. performing sampling of the churners in the data preparation phase.

5.5 Ethical Aspects

There are several ethical aspects that should be considered when performing churn prediction. Churn prediction depends on companies collecting customer data. The data is then used to perform prediction that can help companies to launch targeted marketing campaigns or identify risk groups. This data is valuable for companies, but it is also valuable for surveillance apparatuses. These apparatuses could use the data to track individuals, potentially threatening civil liberty. It is therefore important that companies prevent the data from falling into the wrong hands. The data set used in this study was encrypted with SHA-256 and therefore anonymised, which limit the risk of potential harm. Also, scientists publishing papers should be aware that a robust prediction algorithm could be used by such apparatuses.

Furthermore, it could be argued that churn prediction can harm an economically and ecologically sustainable development, as the ultimate goal is to ensure that a company keeps customers, whom in turn will keep purchasing products. Thus, it can be seen as if churn prediction encourage to a growing consumer economy, leading to more waste that potentially harm ecological sustainability. This is true to some extent, but as churn prediction is a tool to prevent customers from changing point of purchase, i.e. purchase products from a competitor, it does not increase the amount of consuming, rather it tries to concentrate customers to a specific company. However, an already established company with a large budget could potentially spend more resources on churn prediction, which could make it difficult for new companies to enter the market, discouraging a free market.

Chapter 6

Conclusion

Churn prediction is an important tool for companies to stay competitive in a rapidly growing market. In retail, a dynamic definition of churn is needed to identify churners correctly.

This thesis investigated the possibility of combining RNN with CLV time series prediction, a novel approach to churn prediction. The experimental results show that the proposed model performed better than random when predicting churners. It was found that even though the regression accuracy was low, the RNN was capable of identifying trend among all members, which is key when predicting churn. These results indicate the RNN trained to perform CLV time series prediction is a promising approach for churn prediction. The hyperparameter optimisation procedure was not exhaustive, and further optimisation may be worthwhile as it could potentially improve the performance of the RNN. However, it should be noted that while the proposed model managed to identify trends, it struggled to identify members that are prone to deviate from the particular trend of its assigned trend group. This concludes that the proposed model was sensitive to the class imbalance problem.

It was also proposed that the K-means algorithm could be used to conduct a more comprehensive analysis of the predicted churners. The K-means algorithm gave promising and meaningful clusters, which concludes that there was patterns to be distinguished in the data set. Due to a limited time frame, only one clustering solution was investigated. The results of the evaluation show that K-means is applicable for further investigating the output of the churn prediction without the need of a rule-extraction algorithm, resulting in a shorter execution time and a more simple implementation.

6.1 Future work

There are several potential approaches of how to apply RNN in combination with the CLV based definition of churn.

Classification. Rather than using the RNN to predict future CLV values and then indirectly classify churners, it would be interesting to investigate the potential of using the RNN to perform classification directly. This approach would not restrict the RNN the same way regression did, namely one-to-one or many-to-many prediction, which would allow for more input data and less uncertainty. One possible setup for this kind of experiment could be: given x months of data, use $x - t$ months for the training set and validation set, where t is the prediction time frame. For example, use $t = 3$ to train the

RNN to predict churners 3 months ahead. Based on the CLV definition of churn, use the last t months to label members as churners or nonchurners, which will be the target for the training and validation set.

One predictor cluster. The K-means algorithm analysis conducted in this thesis indicated that the RNN was capable of identifying trends, but failed to predict deviations from the trend. This was discussed, and it was suggested that the cause of this was class imbalance problems for certain trend groups. Therefore, it may be worthwhile to investigate the potential of training one RNN predictor per cluster, rather than training a one-size fit all model. The data preparation should also consider the class imbalance problem.

Ensemble learning. Given a meaningful clustering solution of K clusters, it could be worthwhile to investigate the performance of averaging the output of K different RNNs trained separately on each cluster. Ensemble learning could provide for easier training of the K distinct networks and better generalisation. This approach could be applied for both time series prediction and classification.

Bibliography

[1] Antreas D Athanassopoulos. Customer satisfaction cues to support market segmentation and explain switching behavior. *Journal of business research*, 47(3):191–207, 2000.

[2] Raymond Ling and David C Yen. Customer relationship management: An analysis framework and implementation strategies. *Journal of computer information systems*, 41 (3):82–97, 2001.

[3] Adrian Payne and Pennie Frow. A strategic framework for customer relationship management. *Journal of marketing*, 69(4):167–176, 2005.

[4] Injazz J Chen and Karen Popovich. Understanding customer relationship management (crm) people, process and technology. *Business process management journal*, 9(5): 672–688, 2003.

[5] Frederick F Reichheld and Thomas Teal. *The loyalty effect: The hidden force behind growth, profits, and lasting value*. Harvard Business Press, 2001.

[6] Dirk Van den Poel and Bart Lariviere. Customer attrition analysis for financial services using proportional hazard models. *European journal of operational research*, 157(1): 196–217, 2004.

[7] Scott A Neslin, Sunil Gupta, Wagner Kamakura, Junxiang Lu, and Charlotte H Mason. Defection detection: Measuring and understanding the predictive accuracy of customer churn models. *Journal of marketing research*, 43(2):204–211, 2006.

[8] Aurélie Lemmens and Christophe Croux. Bagging and boosting classification trees to predict churn. *Journal of Marketing Research*, 43(2):276–286, 2006.

[9] Parag C Pendharkar. Genetic algorithm based neural network approaches for predicting churn in cellular wireless network services. *Expert Systems with Applications*, 36(3):6714–6720, 2009.

[10] Kristof Coussement and Dirk Van den Poel. 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, 2008.

[11] Ding-An Chiang, Yi-Fan Wang, Shao-Lun Lee, and Cheng-Jung Lin. Goal-oriented sequential pattern for network banking churn analysis. *Expert Systems with Applications*, 25(3):293–302, 2003.

[12] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 369–376, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143891. URL `http://doi.acm.org.focus.lib.kth.se/10.1145/1143844.1143891`.

[13] Ilya Sutskever. *Training recurrent neural networks*. PhD thesis, University of Toronto, 2013.

[14] Bart Baesens, Geert Verstraeten, Dirk Van den Poel, Michael Egmont-Petersen, Patrick Van Kenhove, and Jan Vanthienen. Bayesian network classifiers for identifying the slope of the customer lifecycle of long-life customers. *European Journal of Operational Research*, 156(2):508–523, 2004.

[15] Nicolas Glady, Bart Baesens, and Christophe Croux. Modeling churn using customer lifetime value. *European Journal of Operational Research*, 197(1):402–411, 2009.

[16] Phillip E Pfeifer, Mark E Haskins, and Robert M Conroy. Customer lifetime value, customer profitability, and the treatment of acquisition spending. *Journal of managerial issues*, pages 11–25, 2005.

[17] Stewart C Myers. Finance theory and financial strategy. *Interfaces*, 14(1):126–137, 1984.

[18] Paul D Berger and Nada I Nasr. Customer lifetime value: Marketing models and applications. *Journal of interactive marketing*, 12(1):17–30, 1998.

[19] Sunil Gupta, Donald R Lehmann, and Jennifer Ames Stuart. Valuing customers. *Journal of marketing research*, 41(1):7–18, 2004.

[20] Sunil Gupta, Dominique Hanssens, Bruce Hardie, Wiliam Kahn, V Kumar, Nathaniel Lin, Nalini Ravishanker, and S Sriram. Modeling customer lifetime value. *Journal of service research*, 9(2):139–155, 2006.

[21] Terence P Hughes et al. Catastrophes, phase shifts, and large-scale degradation of a caribbean coral reef. *Science-AAAS-Weekly Paper Edition*, 265(5178):1547–1551, 1994.

[22] Kim Bartel Sheehan. E-mail survey response rates: A review. *Journal of Computer-Mediated Communication*, 6(2):0–0, 2001.

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

[24] Claudio Marcus. A practical yet meaningful approach to customer segmentation. *Journal of consumer marketing*, 15(5):494–504, 1998.

[25] John A McCarty and Manoj Hastak. Segmentation approaches in data-mining: A comparison of rfm, chaid, and logistic regression. *Journal of business research*, 60(6): 656–662, 2007.

[26] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[27] H John B Birks. Recent methodological developments in quantitative descriptive biogeography. In *Annales zoologici fennici*, pages 165–177. JSTOR, 1987.

[28] Kantilal V Mardia, John T Kent, and John M Bibby. Multivariate analysis (probability and mathematical statistics), 1980.

[29] Edward J Bedrick, Jodi Lapidus, and Joseph F Powell. Estimating the mahalanobis distance from mixed continuous and discrete data. *Biometrics*, 56(2):394–401, 2000.

[30] Jianchang Mao and Anil K Jain. A self-organizing network for hyperellipsoidal clustering (hec). *Ieee transactions on neural networks*, 7(1):16–29, 1996.

[31] Andrea Baraldi and Palma Blonda. A survey of fuzzy clustering algorithms for pattern recognition. i. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(6):778–785, 1999.

[32] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31 (8):651–666, 2010.

[33] Inderjit S Dhillon and Dharmendra S Modha. Concept decompositions for large sparse text data using clustering. *Machine learning*, 42(1-2):143–175, 2001.

[34] Krishna Kummamuru, Rohit Lotlikar, Shourya Roy, Karan Singal, and Raghu Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *Proceedings of the 13th international conference on World Wide Web*, pages 658–665. ACM, 2004.

[35] Xiaoli Zhang Fern and Carla E Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *ICML*, volume 3, pages 186–193, 2003.

[36] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[37] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[38] James MacQueen et al. Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1(14):281–297, 1967.

[39] John A Hartigan and JA Hartigan. *Clustering algorithms*, volume 209. Wiley New York, 1975.

[40] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[41] Petros Drineas, Alan M Frieze, Ravi Kannan, Santosh Vempala, and V Vinay. Clustering in large graphs and matrices. In *SODA*, volume 99, pages 291–299. Citeseer, 1999.

[42] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[43] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, volume 1, pages 727–734, 2000.

[44] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[45] Brendan McCane and Michael Albert. Distance functions for categorical and mixed variables. *Pattern Recognition Letters*, 29(7):986–993, 2008.

[46] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*, 2(3):283–304, 1998.

[47] Bart Baesens, Rudy Setiono, Christophe Mues, and Jan Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management science*, 49(3):312–329, 2003.

[48] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[49] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[50] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.

[51] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

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

[53] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 2016.

[54] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[55] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[56] Opher Etzion, Amit Fisher, and Segev Wasserkrug. e-clv: A modeling approach for customer lifetime evaluation in e-commerce domains, with an application and case study for online auction. *Information Systems Frontiers*, 7(4):421–434, 2005. ISSN 1572-9419. doi: 10.1007/s10796-005-4812-6. URL `http://dx.doi.org/10.1007/s10796-005-4812-6`.

[57] Edward C Malthouse and Robert C Blattberg. Can we predict customer lifetime value? *Journal of interactive marketing*, 19(1):2–16, 2005.

[58] C Lee Giles, Steve Lawrence, and Ah Chung Tsoi. Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine learning*, 44(1):161–183, 2001.

[59] Thanasis G Barbounis, John B Theocharis, Minas C Alexiadis, and Petros S Dokopoulos. Long-term wind speed and power forecasting using local recurrent neural network models. *IEEE Transactions on Energy Conversion*, 21(1):273–284, 2006.

[60] Yaya Xie, Xiu Li, EWT Ngai, and Weiyun Ying. Customer churn prediction using improved balanced random forests. *Expert Systems with Applications*, 36(3):5445–5449, 2009.

[61] Pivotal. Springboot: used to create stand-alone, production-grade spring based applications, 2017. URL `http://projects.spring.io/spring-boot/`. [Online; accessed 22-March-2017].

[62] Red Hat. Hibernate: framework for mapping object-oriented domain model to relational database, 2017. URL `http://hibernate.org/`. [Online; accessed 22-March-2017].

[63] Philippe Fournier-Viger. Spmf: an open-source data mining mining library, 2017. URL `http://www.philippe-fournier-viger.com/spmf/index.php`. [Online; accessed 22-March-2017].

[64] Eric Friedman, Rob Eden, and Jeff Randal. Trove4j - high performance collections for java, 2017. URL `http://trove4j.sourceforge.net/html/overview.html`. [Online; accessed 05-May-2017].

[65] Skymind. Deeplearning4j, 2017. URL `https://deeplearning4j.org/about`. [Online; accessed 22-May-2017].

[66] The PostgreSQL Global Development Group. Postgresql 9.6.2 documentation, 2017. URL `https://www.postgresql.org/docs/9.6/static/index.html`. [Online; accessed 05-May-2017].

Appendix A

Data Set

The data persisted in the database.

Members were disabled during the cleaning step of the proposed model. Members disabled due to registered later than 2013 was 1,128,947. 233,886 of the remaining members had fewer than six purchase occasions in total and were disabled. Thus, the experiments were done on 734,490 members, about 35% of the initial amount of members.

A.1 Member Related Information

Table A.1: Field name and description of the member data

Field name	Field description
member_ID	Unique identifier of the member
gender	The member's gender
age	The member's age
country	Home country
region	Home region
registrationDate	The date of when member registered
cluster	The number of the cluster assigned to this member

A.2 Transaction History Related Information

The currency was different for some transactions, with payments made in SEK, NOK, and EURO. The currency was converted to the same (SEK), which was done by using the rounded average value of the respective currencies (1 EURO = 9 SEK. 1 NOK = 1.00 SEK) during the period for transactions. NOK was considered the same value as SEK because of minimal difference in currency.

Table A.2: Field name and description of the transaction history data

Field name	Field description
transaction_ID	Unique identifier for each transaction
businessDate	Timestamp of when the transaction occurred
member_ID	Unique identifier of the member associated with the transaction
item_code	The universal product code of the purchased item
store_code	identifier of the customer associated with the transaction
discount	The amount discounted

Appendix B

Experimental Setup

B.1 PostgreSQL Database Configuration Parameters

Several of the parameters in the database were altered, presented in Table B.1.

Table B.1: The PostgreSQL parameters using Amazon RDS instance db.m4.xlarge

Parameter	Value
maintenance_work_mem	500MB
work_mem	32MB
autovacuum_max_workers	20
autovacuum_analyze_threshold	5,000
autovacuum_analyze_scale_factor	0
autovacuum_vacuum_threshold	5,000
autovacuum_vacuum_scale_factor	0
max_parallel_workers_per_gather	10
default_statistic_target	2,000ms
random_page_cost	2

In addition to these settings, all tables were created using the UNLOGGED flag, disabling writes to the write-ahead log. For more information about the parameters, check the PostgreSQL manual [66].

Appendix C

K-means Results

C.1 K-means Profiling Tables

This section includes the K-means profiling tables that were computed. Specifically, it includes the profiling tables for the best clustering solution for K=5, K=15 and K=20 respectively.

Table C.1: Profiling table over the clustering fields compared to overall mean, K=5

	Members	Total Amount	Total Discount	Total Orders	Days Since Last Purchase	Days Since Registration	Average Time Between Purchases	Total Amount Web Shop
Cluster 1	142,282	74%	70%	9%	159%	119%	174%	58%
Cluster 2	231,636	133%	135%	207%	42%	116%	43%	199%
Cluster 3	188,064	86%	80%	51%	107%	118%	98%	85%
Cluster 4	52,432	76%	73%	24%	144%	74%	139%	46%
Cluster 5	120,076	131%	142%	209%	49%	73%	46%	151%
Overall	734,490	0.0286	0.0149	0.0214	0.1007	0.8133	0.4458	0.0193

In Table C.1, two distinct behaviour groups can be observed. Clusters 1, 3, and 4 are quite similar, with cluster 4 having more recently registered members. Clusters 2 and 5 is made out of members with similar RFM attributes. Note that cluster 3 consists of newer members that spend more money in physical stores than the older members of cluster 2.

When comparing the profiling table for K=15 (Table C.2), with the profiling table for K=10, one particularly interesting observation can be done. Cluster 14 consists of the highest average spending members. These members purchases discounted items and preferably online. When comparing the average time between purchases and total orders for cluster 3, 5 and 14, it can be observed that there exist groups of members that on average have the same total orders and frequency, while members of cluster 3 and 5 prefer physical stores.

Table C.2: Table over the clustering fields compared to overall mean, K=15

	Members	Total Amount	Total Discount	Total Orders	Days Since Last Purchase	Days Since Registration	Average Time Between Purchases	Total Amount Web Shop
Cluster 1	27,996	69%	67%	9%	169%	69%	187%	29%
Cluster 2	0	0	0	0	0	0	0	0
Cluster 3	149,475	139%	143%	220%	39%	107%	41%	87%
Cluster 4	0	0	0	0	0	0	0	0
Cluster 5	87,399	134%	149%	215%	68%	74%	41%	88%
Cluster 6	36,514	81%	78%	48%	338%	108%	108%	39%
Cluster 7	40,406	70%	67%	7%	381%	110%	196%	41%
Cluster 8	11,480	100%	100%	64%	76%	111%	92%	1,217%
Cluster 9	137,248	91%	84%	75%	66%	110%	79%	27%
Cluster 10	110,808	76%	71%	27%	82%	111%	136%	29%
Cluster 11	0	0	0	0	0	0	0	0
Cluster 12	0	0	0	0	0	0	0	0
Cluster 13	67,888	70%	66%	4%	96%	112%	207%	46%
Cluster 14	10,760	179%	228%	231%	40%	99%	40%	2,085%
Cluster 15	54,446	81%	78%	51%	100%	70%	102%	44%
Overall	734,490	0.0286	0.0149	0.0214	0.1007	0.8133	0.4458	0.0193

Table C.3: Table over the clustering fields compared to overall mean, K=20

	Members	Total Amount	Total Discount	Total Orders	Days Since Last Purchase	Days Since Registration	Average Time Between Purchases	Total Amount Web Shop
Cluster 1	69,370	130%	138%	207%	44%	90%	42%	58%
Cluster 2	47,503	86%	82%	67%	86%	70%	86%	50%
Cluster 3	0	0	0	0	0	0	0	0
Cluster 4	7,259	94%	94%	40%	92%	112%	116%	1,386%
Cluster 5	0	0	0	0	0	0	0	0
Cluster 6	39,967	70%	68%	7%	382%	111%	196%	67%
Cluster 7	122,276	90%	82%	71%	68%	111%	82%	33%
Cluster 8	110,400	137%	136%	209%	39%	112%	40%	39%
Cluster 9	0	0	0	0	0	0	0	0
Cluster 10	108,557	76%	71%	27%	82%	112%	136%	22%
Cluster 11	6,445	188%	240%	214%	44%	95%	44%	2,533%
Cluster 12	0	0	0	0	0	0	0	0
Cluster 13	19,017	67%	66%	4%	168%	70%	206%	25%
Cluster 14	35,489	81%	78%	48%	340%	109%	108%	37%
Cluster 15	0	0	0	0	0	0	0	0
Cluster 16	28,609	73%	70%	25%	144%	70%	140%	35%
Cluster 17	17,640	141%	165%	202%	42%	109%	44%	987%
Cluster 18	67,058	70%	66%	39%	96%	112%	206%	41%
Cluster 19	54,830	143%	162%	239%	37%	62%	37%	99%
Cluster 20	0	0	0	0	0	0	0	0
Overall	734,490	0.0286	0.0149	0.0214	0.1007	0.8133	0.4458	0.0193

Appendix D

RNN Related Appendix

D.1 RNN Hyperparameter Configuration

Xavier's weight initialisation was used for all RNNs. Stochastic gradient descent was used as optimisation algorithm for all RNNs. Different updaters were investigated, like RMSProp and Adam. Best performance was achieved with the Nesterov updater.Early stopping was used without regularisation. Regularisation and dropout were briefly investigated but showed no significant improvement performance wise. A batch size of 13,000 was used.

Different loss functions, such as mean absolute error (MAE) and mean squared logarithmic error (MSLE), was investigated. The CLV of the members differ in magnitude, therefore MSLE was expected to perform well, as it measures the ratio between the actual value and predicted. However, MAE and MSLE performed worse than mean squared error (MSE).

It was found that while using a larger network does improve the training set accuracy, it did not necessarily improve the validate set accuracy. As only one predictor variable was used as input, the function approximated is regarded as simple. A more complex and large network is more prone to overfitting, which was confirmed early in the experimenting.

Hyperparameter configurations that showed promising validation set accuracy was trained for 48-72 hours.

D.2 RNN Hyperparameter Configuration Results

The configurations and belonging statistics are presented in this section.

Table D.1: Hyper Parameter configuration Model A

Hyperparameter	Value
Updater	Nesterovs, momentum 0.1
Learning rate	0.005
Activation Function	Sigmoid
Recurrent Connections	10
Stacked RNNs	1

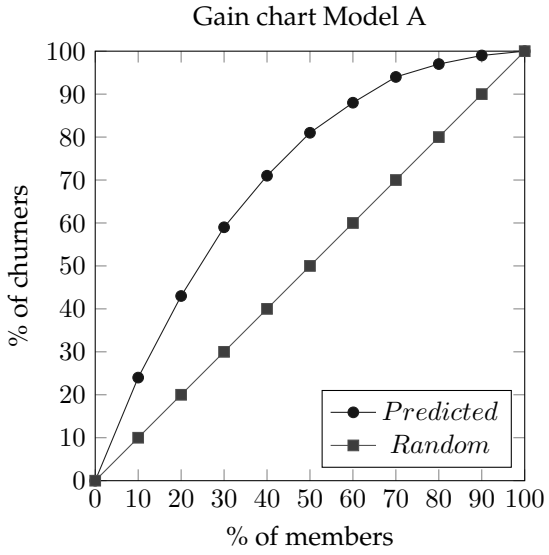

Table D.2: Gain and lift table model A

Decile	Gain (%)	Lift (%)
1	24.26	2.43
2	42.81	2.14
3	58.56	1.95
4	71.21	1.78
5	81.19	1.62
6	88.46	1.47
7	93.74	1.34
8	97.21	1.22
9	99.12	1.10
10	100.00	1.00

Table D.3: Prediction statistics for model A

Max	Min	Mean	σ
24,519	-10,127	-9,284	352

Table D.4: Hyper Parameter configuration Model B

Hyperparameter	Value
Updater	Nesterovs, momentum 0.8
Learning rate	0.0001
Activation Function	Tanh
Recurrent Connections	10
Stacked RNNs	1

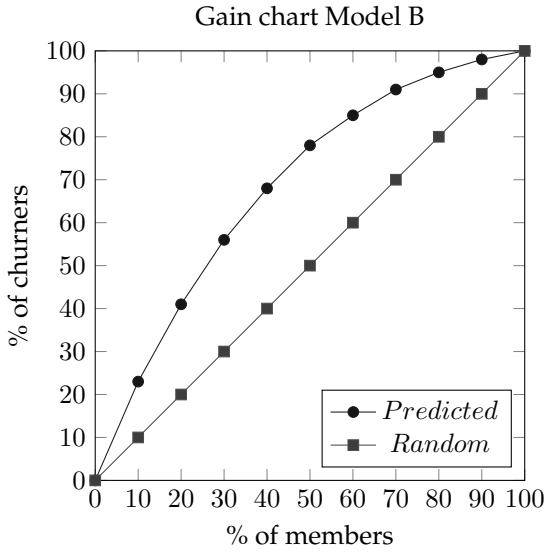

Table D.5: Gain and lift table model B

Decile	Gain (%)	Lift (%)
1	23.21	2.32
2	40.74	2.04
3	55.59	1.85
4	67.83	1.70
5	77.63	1.55
6	85.21	1.42
7	90.87	1.30
8	95.13	1.19
9	98.01	1.09
10	100.00	1.00

Table D.6: Prediction statistics model B

Max	Min	Mean	σ
65,445	-24,120	-5,833	5,082

Table D.7: Hyper Parameter configuration Model C

Hyperparameter	Value
Updater	Nesterovs, momentum 0.8
Learning rate	0.0001
Activation Function	ReLU
Recurrent Connections	15
Stacked RNNs	1

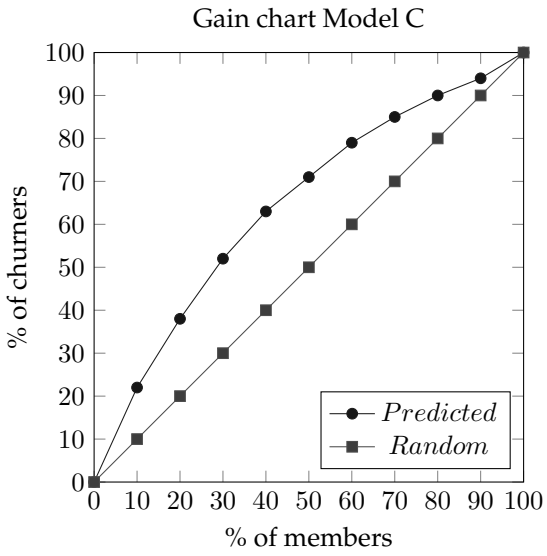

Table D.8: Gain and lift table model C

Decile	Gain (%)	Lift (%)
1	22.23	2.22
2	38.40	1.92
3	51.53	1.72
4	62.57	1.56
5	71.38	1.43
6	78.58	1.31
7	84.59	1.21
8	89.79	1.12
9	94.22	1.05
10	100.00	1.00

Table D.9: Prediction statistics model C

Max	Min	Mean	σ
77,975	-3,073	462	1,875

Table D.10: Hyper Parameter configuration Model D

Hyperparameter	Value
Updater	Nesterovs, momentum 0.8
Learning rate	0.00001
Activation Function	ReLU
Recurrent Connections 1st layer	7
Recurrent Connections 2nd layer	5
Stacked RNNs	2

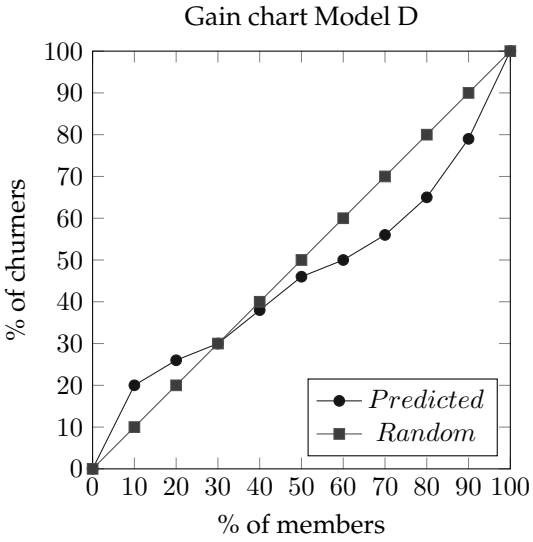

Table D.11: Gain and lift table model D

Decile	Gain (%)	Lift (%)
1	20.15	2.01
2	25.75	1.29
3	30.31	1.01
4	37.64	0.94
5	45.92	0.92
6	50.27	0.84
7	55.75	0.80
8	64.73	0.81
9	78.84	0.88
10	100.00	1.00

Table D.12: Prediction statistics model D

Max	Min	Mean	σ
1,160	-63,384	276	789

Table D.13: Hyper Parameter configuration Model E

Hyperparameter	Value
Updater	Nesterovs, momentum 0.8
Learning rate	0.0001
Activation Function	ReLU
Recurrent Connections	16
Stacked RNNs	1

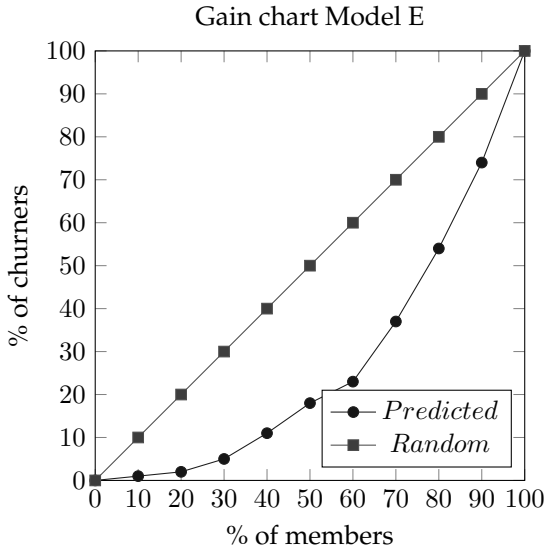

Table D.14: Gain and lift table model E

Decile	Gain (%)	Lift (%)
1	0.58	0.06
2	2.16	0.11
3	5.41	0.18
4	10.69	0.27
5	18.21	0.36
6	22.81	0.38
7	36.51	0.52
8	53.55	0.67
9	73.76	0.82
10	100.00	1.00

Table D.15: Prediction statistics model E

Max	Min	Mean	σ
1,509,902	12,480	38,818	8,479

Table D.16: Hyper Parameter configuration Model F

Hyperparameter	Value
Updater	Nesterovs, momentum 0.8
Learning rate	0.0001
Activation Function	Sigmoid
Recurrent Connections	16
Stacked RNNs	1

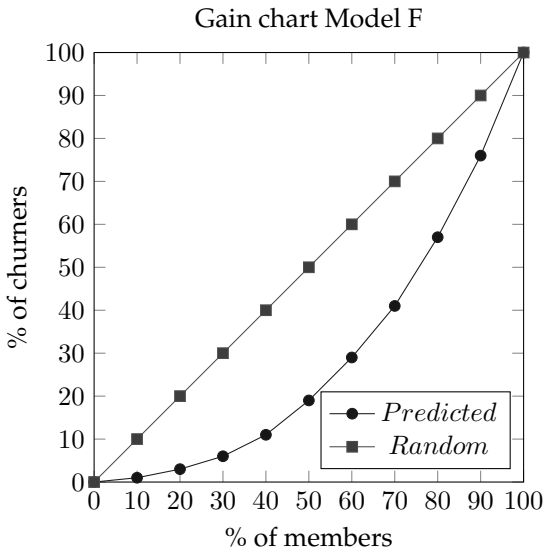

Table D.17: Gain and lift table model F

Decile	Gain (%)	Lift (%)
1	0.83	0.08
2	2.68	0.13
3	6.08	0.20
4	11.33	0.28
5	18.57	0.37
6	28.58	0.48
7	41.21	0.59
8	57.03	0.71
9	75.65	0.84
10	100.00	1.00

Table D.18: Prediction statistics model F

Max	Min	Mean	σ
13,714	7,071	7,256	96