UNIVERSITY OF AGDER

Non-contractual churn prediction using Hierarchical Temporal Memory

Jone K. Bakkevig, and Magnus Methi

SUPERVISORS
Morten Goodwin
Jahn Thomas Fidje

Master's Thesis
University of Agder, 2018
Faculty of Engineering and Science
Department of ICT

UiA
University of Agder
Master's thesis

Faculty of Engineering and Science
Department of ICT

Abstract

As markets become more saturated and industry leaders compete over the existing customer base, competitors look for ways to improve customer retention with their customers. It is considered much more expensive to gain a new customer than retaining one, so the industry leaders look for ways in which churn in a customer can be predicted and potentially be avoided by incentivizing the customer to stay. Several of the previously proposed approaches struggle with combining the linearity with the non-linearity that exist within churn analysis and prediction. This emphasizes the need for research into state-of-the-art algorithms that furthers the knowledge regarding churn analysis that utilize the temporal structure of the data in a prediction based model. As contributions to this end, this thesis examines a Hierarchical Temporal Memory (HTM) approach to predict the future purchase events of customers in a non-contractual setting. The thesis compare the results of the HTM to the potential of existing state-of-the-art in the same context. The research shows HTMs potential through documenting performance with increasing data availability. The robustness of the implementation remains in question as complexity issues arise in conceptualizing a good definition for churn. HTM proves to be a viable churn detection algorithm, but has weaknesses in terms of churn prediction. The robustness of HTM increases with available data and the multimodality of that data.

Table of Contents

Glossary

ANN Artifical Neural Network. 10

ARIMA Autoregressive Integrated Moving Average. 10, 14

CART Classification and Regression Trees. 8

CDF Cumulative Distribution Function. 12

CHF Cumulative Hazard Function. 12

CLV Customer Lifetime Value. 3, 9, 11

CMF Cumulative Mass Function. xi, 13, 42, 43, 46

CRM Customer Relationship Management. 3

HF Hazard Function. 12

HTM Hierarchical Temporal Memory. iii, xi, xiii, 4, 5, 11, 14, 16, 23–26, 31, 33–37, 39, 40, 45–47

IoT Internet of Things. 14

ISP Internet Service Provider. 8, 9

LSTM Long Short-Term Memory. 4, 5, 9–11, 14

MAPE Mean Absolute Percentage Error. 25

MSE Mean Square Error. 25

NuPIC Numenta Platform for Intelligence Computing. 14, 23, 25, 26

OPF Online Prediction Framework. 14

PDF Probability Density Function. 12

PMF Probability Mass Function. 13

RFM Recency, Frequency, Monetary. 3

RMSE Root Mean Square Error. xiii, 25, 38

RNN Recurrent Neural Network. 4, 5, 9–14, 28

SDR Sparse Distributed Representation. 14, 16, 17, 23, 24, 33

SP Spatial Pooler. 14, 16, 17, 23, 24, 33

SVM Support Vector Machine. 10

TTE Time To Event. xi, 13, 24, 25, 28, 33–40, 45–47

WTTE Weibull Time To Event. 11

WTTE-RNN Weibull Time To Event Recurrent Neural Network. xi, 11, 24, 26, 28, 29, 41, 42, 45–47

List of Figures

List of Tables

Part I

Research Overview

Chapter 1

Introduction

1.1 Introduction

As industry leaders compete over a finite number of customers, the market saturation increases. [24] Customers are looking for value-added relationships with their vendors in order to remain as loyal customers. [41] Businesses consequently look for ways to incentivize customers and provide the required value-added for their customers. In order to improve upon this process these aforementioned businesses look for ways to recognize patterns in the customers behaviour that indicates a decline in the relationship between the customer and the business. This is established as Customer Relationship Management (CRM), CRM is a strategy used by businesses to effectively allocate their resources to maintain and improve customer relationship. [21] [29]

The process, in essence, attempts to define and detect the churn rate of customers. Churning is defined as either a decline in use of a service or the termination of that service by a customer, a churner is defined as a customer that churns. In order for the businesses to establish these value-added relationships and avoid churning, several models have been developed to predict the customer behaviour to determine when a customer should receive an incentive or at which rate these incentives should be presented for the relationship to remain as profitable as possible. Several methods such as Customer Lifetime Value (CLV), Recency, Frequency, Monetary (RFM) and more predictive models using Machine Learning have been used to determine customer churn. CRM is a strategy used for businesses to effectively

allocate their resources, they are interested in determining the physical churn that occurs in their customer base. [21] [38] [34]

There exists several proposed methods of determining churn, differing between the contractual and non-contractual nature of the business with their corresponding available data structure. Depending on the data, the existing state-of-the-art methods use very detailed transaction history with churn labelling in order to determine churn for its uses.

The goal in this thesis is to establish from a transaction history a likelihood for churn for a given customer based on the customers existing transaction pattern. The state-of-the-art approaches for this subject such as Recurrent Neural Network (RNN) using Long Short-Term Memory (LSTM) create a fixed model for the prediction. We explore, as the most prevalent state-of-the-art, an approach proposed by E.Martinsson using a Weibull distribution in conjunction with a RNN to determine the statistical properties of an occurring event for churn modelling. [35] Differentiating from that, this thesis proposes utilization of HTM instead, promising a data-stream type solution that gives way for a more practical approach to be utilized by vendors that wish to compute and analyze churn.

The thesis hypothesizes an increase in usefulness from using a temporal memory approach to the established churn prediction problems. The term usefulness is hereby defined as a consolidated metric pertaining to efficiency in computation, increase in measured prediction accuracy and ease of implementation and understanding.

1.2 Motivation

The topic of churn prediction is chosen in motivation to further the research of machine learning use cases in applications previously completed by deterministic approaches. In addition, establishing a more practical approach to data analysis via streaming data with applicable algorithms.

1.3 Goal

The goal of this thesis is to explore the practicality of HTM to perform churn prediction in a non-contractual setting.

1.3.1 Field of research

This thesis entails Machine Learning use within a financial scope application.

1.4 Statement of the Problem

HTM is a more practical method of establishing churn with comparable results to existing state of the art RNN using LSTM. By gauging HTM's effectiveness through practical use cases derived from a real transaction history data set and a generated data set, the proposed state-of-the-art algorithm will be compared to the existing state-of-the-art algorithm. The comparison will determine the validity of the proposed state-of-the-art in a variety of metrics such as accuracy, practicality and efficiency.

1.5 Contributions

The contributions in this thesis are twofold; The thesis will contribute insights into handling analysis of non-contractual data in order to determine churn. The thesis will also contribute to a practical implementation of the HTM implementation on the provided and generated data sets.

1.6 Target audience

This thesis aims at increasing the scope of knowledge regarding non-contractual churn prediction available to banks and businesses with interest in obtaining prediction models for non-contractual churn in their consumer data. Specific businesses are interested in resource allocation in order to avoid customers churning

to separate businesses that fulfill the same service while banks are interested in this application for anomaly detection in behaviour leading to potentially detect fraudulent transactions.

1.7 Thesis Organization

The thesis is structured in the following manner. Chapters 1 and 2 detail field of study with the problem statement and relevant theoretical framework related to the problem. Chapter 3 explains the proposed state-of-the-art implementation with practical insights to the implementation, the comparable existing state-of-the-art is also detailed. Chapter 4 shows testing of the algorithm with parameters measured to perceive the viability of the implementation. Chapter 5 discusses the findings and the knowledge gained through the research conducted within the thesis.

Chapter 2

Literature Review

This chapter reviews the existing literature that creates the theoretical framework upon which this thesis establishes its contribution efforts. The theoretical framework consists of the previously proposed solutions and state-of-the-art approaches to the applicable field of study. As the field of study covers a set of specialized algorithms used in an established context, the state-of-the-art section covers the concepts of and the functionality of these algorithms and the pertinent underlying mathematical formulas used.

2.1 Churn prediction

Churn prediction is one of the most common and complex machine learning problems, and the results of certain churn prediction algorithms can vary on a wide spectrum. [37] The reason for such varying methods with results lie within the terminology defined during the project scope. In order to have the features contribute to a real and achievable result it is imperative to not define terminology that either restricts the algorithm interpretation or have a too ambiguous of an output. The most important features to define are customer, churn and churn probability. These are particularly important because of hidden restrictions in the data set. In order for a customer to be a valid customer considered for churn prediction purposes, a certain amount of transactions must have been completed, otherwise there will occur an overfitting of the data distribution. This number of transactions must be considered on the entire size of the data set and whether or not a customer exists

within the time-frames allotted by the algorithm. Additionally, churn per definition is the likelihood of a customer replacing service providers i.e. swapping ISPs or lowering the frequency of purchases at store chain of which to do their grocery shopping. Churn should not be defined as the end of transaction history for the customer as the data set can create false positives.

The initial established methods of churn prediction were highly deterministic and based on classification. The Classification and Regression Trees (CART) algorithm by L. Breiman [15] made the foundation for such churn analysis. As he built upon his work by introducing bootstrap aggregators or *bagging* in. [16] The *bagging* is a way to create an aggregated predictor by bootstrapping replicates of the learning set and using it as additional new learning sets to predictors. [16] The predictors are then aggregated. Random forests introduced in [17] by L. Breiman builds upon his previous work in [16] by adding an additional layer of randomness to the *bagging*. This is done by by selecting a random subset of descriptors to grow trees and each of those trees are grown on a bootstrap sample of the training set. One of the detriments of random forests is its inaccuracy on unbalanced data. A customer churn data set is usually quite unbalanced as the percentage of customers who churn are sparsely labelled or relatively unknown. This detriment was sought to be improved by weighting random forests and balanced random forests. Weighting random forests is based on cost-sensitive learning while balanced random forests is based on a sampling technique. [46] As the approaches move from classification to predictive measures the technology and researched advanced. The neural network classification implementation in [36] shows increased promise of a neural network approach in classification, but highlights the ambiguity of churn definition.

In modern research there are two main types of churn prediction, contractual and non-contractual. The different churn prediction areas contain different approaches and challenges. Several machine learning techniques and algorithms have been utilized across both paradigms, but the approaches can vary vastly due to the nature of the data structures.

2.1.1 Contractual churn

As the name suggests, contractual churn relates to all contractual business models where the customer enters a “contract” with a business, establishing a rule set to govern the transactions made between the two parties. In most cases these scenarios consist of a monetary exchange for a service done by the business for

the customer. Contractual churn prediction envelops any business-customer relationship that is based on any variation of the subscription model such as gym memberships, Internet Service Provider (ISP)s or telecommunication carriers to name a few. The nature of this relationships is represented in the data available for contractual churn prediction. In addition to the amount paid periodically by the customer, additional, very descriptive features are present. This is information such as length of contract, membership tier, previous relationship with the vendor and potential inquiries made. Contractual churn is much more immediate and binary than non-contractual churn, as when the contract is terminated, per definition the customer has churned. The churn prediction component lies within determining patterns that lead up to the churning of a customer in this scenario so that businesses may provide incentives for the customer to stay or reconsider its upcoming termination. Several methods have been proposed and established in order to solve the contractual churn problem. In addition to just higher orders of rule sets, maneuvering into the machine intelligence realm several classification methods have been proposed of decision trees, random forests with improved versions and eventually moving into neural network models. The primary concern of the initial proposed methods have been related to the sequential nature of the data, moving toward memory based approaches have helped with these problems. The most prevalent state-of-the-art within contractual churn uses a Recurrent Neural Network with Long Short-Term Memory. Genetic algorithms and evolutionary approaches show promise but lack the ability to produce the likelihood of the predictions made [14] [39] [12] [31] [36] [11].

2.1.2 Non-contractual churn

As churners in a non-contractual setting are more abstractly defined, the neural network classification methods become less viable implementations to consider. Another criticism of the neural network includes the fact that the implementation produces an output vector to further base decisions on - this makes it less easy to understand [14].

Previously proposed non-contractual churn prediction has been deterministic, with established rules that are enforced by decision trees or random forests [43]. One problem with decision trees is that some of the leaves in the decision tree might contain the same probability and consequently create noise [14]. In addition, [27] proposed a CLV modeling approach in which customers churn rate is estimated based on decreasing CLV values. This approach utilizes both neural networks and decision trees in its churn prediction analysis. The problem with the determinis-

tic approaches lies within flexibility, within a dynamic environment the models become too static and invariable to change. The existing state-of-the-art non-contractual churn prediction looks at a customers behavioural pattern over time and attempts to extrapolate a likelihood for churn based on a reduced frequency in the pattern. The non-contractual churn prediction may also be described by more anomalous behaviour that may be reflected in the pattern but by less obvious details. The nature of the data when performing non-contractual churn analysis and prediction lends itself to a time-series forecasting and sequence prediction approach. As shown in [34] a RNN with LSTM is used to predict the customer churn based on purchasing amount and frequency [19] [46] [45].

2.2 Time series prediction

Essential to understanding and creating solutions to challenges presented with any temporal data structure comes the concept of time series prediction, otherwise known as time series forecasting. Different algorithms and methods have been utilized in determining a future prediction or forecast of events based on existing data that depict the event history of the respective events. Support Vector Machine (SVM) has been utilized extensively to this task, especially in stock market predictions where the predictions have been performed on the next days daily stock market price. The implementation attempts to predict whether the stock market price is higher or lower than the current, and trains on this methodology. The accuracy and validity of the approach is then measured. This implementation is subject to a granularity and a bias that occurs within the predictions. As each prediction is on a day per day basis, over a longer period of time the integrity of the data may disperse. [32] [44].

As stated by M. Pratama et al. streaming solutions are becoming more prevalent as the data collection rate increases and the demand for online real-time strategies is required to prevent loss of accuracy and to avoid system instability. [40] [25]

One of the challenges in time series prediction is the combination of non-linearity combined with linearity. In attempt to solve this problem, G. P. Zhang proposed in [47] combining the linear Autoregressive Integrated Moving Average (ARIMA) model with a nonlinear Artifical Neural Network (ANN). The proposed solution proved benefits in error and deviation metrics [47].

2.2.1 RNN and LSTM

LSTM keeps an important feature as a memory and carries the memory to be used further. This is an improvement from the standard RNN and consequently lends itself to sequential data [31]. RNN with LSTM is utilized for many different things such as language and acoustic models for speech recognition, facial emotion recognition and other problems that require sequential modelling. [18] [28]. In speech recognition the RNN is trained to correspond sequences of acoustic and phonetics. It returns a sequence of transponded signals [28]. Continually proving that RNN is useful in temporal contexts, the literature shows RNNs proficiency in modelling time series data over the previously proposed methods such as feedforward networks. [20] [33] details the evolution of RNN with LSTM especially its potential within sequential modelling and with creative new architectures for the tasks at hand. In [34] RNN is used in conjunction with LSTM to determine churners based on existing metrics such as how many purchases and time since last purchase from customers in an online-store. Comparable to the thesis topic, that project scope is not specifically defined as non-contractual churn as there is a semi-contractual nature to an online store.

2.2.2 WTTE-RNN

The results reported by J. Ljungehed et al.[34] indicate that predicting a CLV could result in an incorrect prediction of a churner. As proposed by E. Martinsson [35] a Weibull distribution in conjunction with a RNN creates a Weibull Time To Event RNN that predicts the estimated time to next event and can be utilized based on the predicted time to determine the churning potential of a customer. The goal is to present the potential for this previous state-of-the-art as to give a context to which the proposed state-of-the-art is compared. In [5] the author details an approach to churn modelling very thoroughly but in a generalized case. The approach is adopted and performed on the same data that is used for the Hierarchical Temporal Memory.

The Weibull Distribution

The proposed approach utilizes a Weibull Distribution in conjunction with a RNN in order to predict a likelihood distribution of the next event occurring. In each step of the WTTE-RNN, α and β are the outputs of the RNN. The potential struc-

tures for this RNN can be varying, so for implementation purposes a model is constructed equivalent to examples given by the author of the proposed method. The parametrization used for the Weibull distribution has the following Probability Density Function (PDF), Cumulative Distribution Function (CDF), Cumulative Hazard Function (CHF) and Hazard Function (HF).

$$PDF : F(x) = 1 - exp[-(\frac{t}{\alpha})^{\beta}] \tag{2.1}$$

$$CDF : f(x) = \frac{\beta}{\alpha}(\frac{t}{\alpha})^{\beta-1} exp[-(\frac{t}{\alpha})^{\beta}] \tag{2.2}$$

$$CHF : \Lambda(x) = (\frac{t}{\alpha})^{\beta} \tag{2.3}$$

$$HF : \lambda(x) = (\frac{t}{\alpha})^{\beta-1} \cdot \frac{\beta}{\alpha} \tag{2.4}$$

With corresponding variables:

$$t \in [0, \infty] \tag{2.5}$$

$$\alpha \in (0, \infty) \tag{2.6}$$

$$\beta \in (0, \infty) \tag{2.7}$$

$$T \sim Weibull(\alpha, \beta) \tag{2.8}$$

Log-Likelihood

The loss calculation used in [35] describes the calculation of loss as an evaluation of log of the survival function. The evaluation is completed as follows:

$$\log(\mathcal{L}) = u \cdot [\beta \cdot \log(\frac{t}{\alpha}) + \log(\beta)] - (\frac{t}{\alpha})^{\beta} \tag{2.9}$$

This functions as the loss function for the WTTE-RNN to maximize, as long as the distribution is discrete. [35]

In order to utilize the α and β values produced by the RNN, the following functions, Probability Mass Function (PMF) and Cumulative Mass Function (CMF) are used:

$$PMF : CDF(t + 1.0, \alpha, \beta) - CDF(t, \alpha, \beta) \tag{2.10}$$

$$CMF : CDF(t + 1.0, \alpha, \beta) \tag{2.11}$$

t in this case is the query of which the next event will be predicted to. To determine that from the last event will occur on *t*, the PMF is used. For a probability of an event occurring from the last event within *t* days, the CMF is used. [5]

TTE-RNN

The TTE-RNN is defined in [35] as:

1. Let *y* be an observed time
2. Let *u* be an indicator s.t *u* = 0 (if *y* is uncensored, *y* = 1)
3. $R(\theta, y) = \Lambda(y)$ be some RCHF parametrized by θ
4. θ be the output of a RNN

WTTE-RNN

The Weibull TTE RNN is a special case of TTE-RNN. For the purposes in this thesis the discrete case is used. When $\Lambda(y) = (\frac{y}{\alpha})^\beta$ the model is transformed to the proposed WTTE-RNN [35].

2.3 Hierarchical Temporal Memory (HTM)

HTM is a biologically constrained theory of machine intelligence. It is based on the neuroanatomy and neurophysiology of the neocortex and aims to only rely on principles that may be implemented in biological tissue [30]. The literature

suggests that HTM performs robustly on traditional machine learning tasks such as image recognition[26], and excels at problems with an inherent spatio-temporal structure. It is less robust whith time-series of spatial arrangements, but extending on traditional HTM has proven to mitigate the issue [42].

Numenta Platform for Intelligence Computing (NuPIC) is an open source implementation of HTM. It includes an API called Online Prediction Framework (OPF) that is tuned towards experimentation with prediction and anomaly detection. The implementation utilizes online learning with streaming data [8][7]. Online real-time strategies are becoming increasingly popular, compared to the batch alternative, as it allows for reliable accuracy with O(1) complexity when handling large amounts of data[25] gathered by sources such as smart sensors, information technology and Internet of Things (IoT) [13].

While the theory and implementation of HTM is under ongoing development by Numenta, the key aspects are Sparse Distributed Representation (SDR), the Spatial Pooler (SP) and Temporal Memory. A swarming algorithm is implemented to decide on components and parameters according to the input data.

2.3.1 Prior Application

HTM has previously been utilized to solve problems involving temporal patterns and sequence predictions [42][22][2].

Yuwei Cui et al. compared HTM to other sequence learning algorithms, exploring accuracy and robustness in different scenarios. LSTM represented the state-of-the-art RNN model for sequence learning tasks. They state that HTM is advantageous for continuous learning from streaming data as it learns from each data point using unsupervised learning instead of needing "[...]to store a batch of data as the 'training set'" [22]. Figure 2.1 shows the results from their experiment on a real-life scenario. They found HTM to outperform LSTM and ARIMA among others. Though it is worth noting that the methods compared to HTM were converted to online learning algorithms instead of using batches[22]. They also found HTM to be quite robust as it performed comparable to LSTM on a variety of different problems while using the same set of parameters[22].

Building upon sequence prediction, HTM has been applied to anomaly detection[10] where the prediction is done to establish the regular, expected patterns in the data. Numenta has developed some example applications using HTM for anomaly de-

tection [3][1].

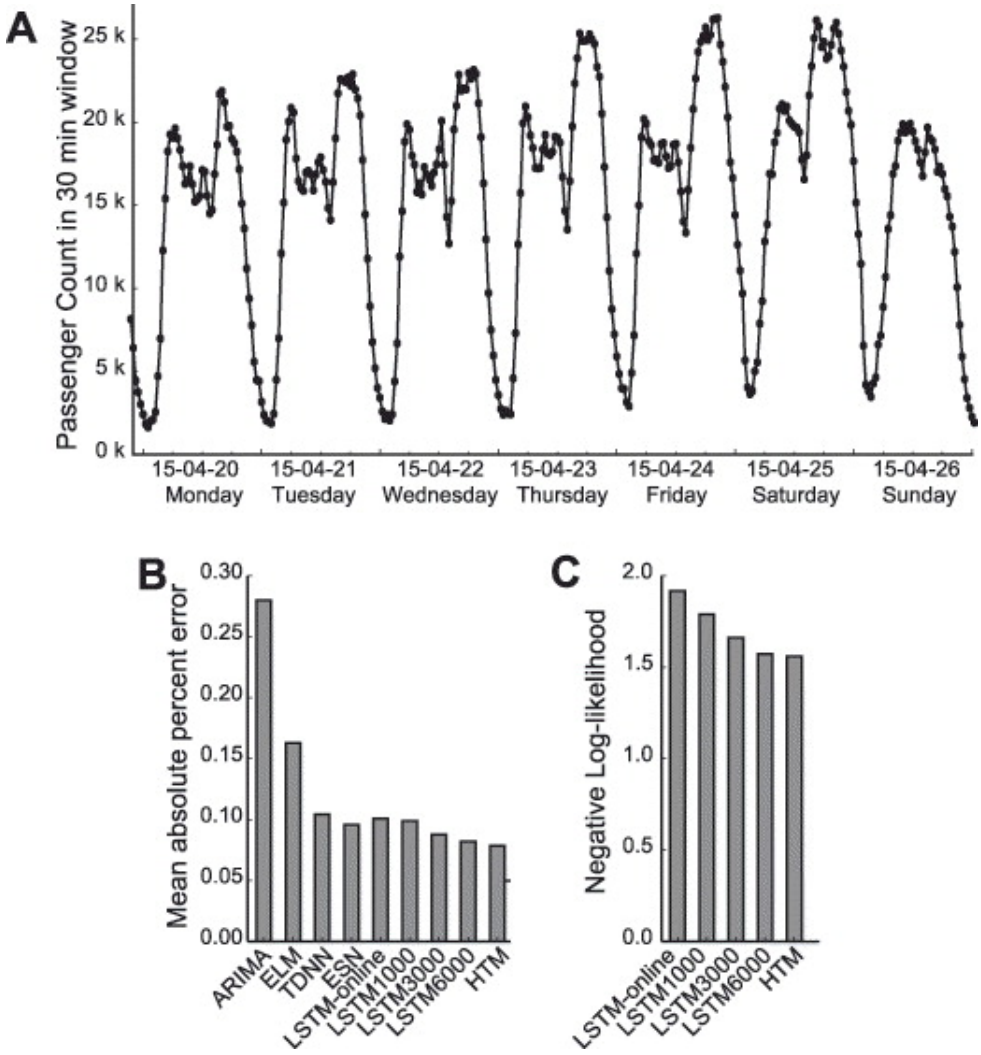


Figure 2.1: "Prediction of the New York City taxi passenger data. (A) Example portion of taxi passenger data (aggregated at 30 min intervals). The data have rich temporal patterns at both daily and weekly timescales. (B, C) Prediction error of different sequence prediction algorithms using two metrics: mean absolute percentage error (B), and negative log likelihood (C)."[22]

2.3.2 Sparse Distributed Representation (SDR)

In order to simulate the sensory input of the neocortex, HTM uses encoders to convert input types to a SDR. SDR is a collection of bits where only a small percentage of them are activated. The size of the collection can vary, but usually contains thousands of bits. This is supposed to preserve relationships and similarities between inputs by matching active bits. For instance, two time stamps representing weekend days will have a number of active bits that are located in the same positions. It is important that an encoder output the same SDR for the same input. [30]

2.3.3 Spatial Pooler (SP)

The Spatial Pooler is a learning algorithm in HTM that takes input and creates sparse patterns. It contains a collection of columns of cells. The input space is a SDR encoding and each column has a number of connections to the input space. When a connection overlaps with an active bit in the input space it contributes to the score of the column. The highest scoring columns will be activated. Figure 2.2 visualizes the relation between a column in the SP and the input space. If the column has no prior state, then all its cells will be activated as well. The sparse patterns created by the SP is then used to recognize and predict sequences of input. [30]

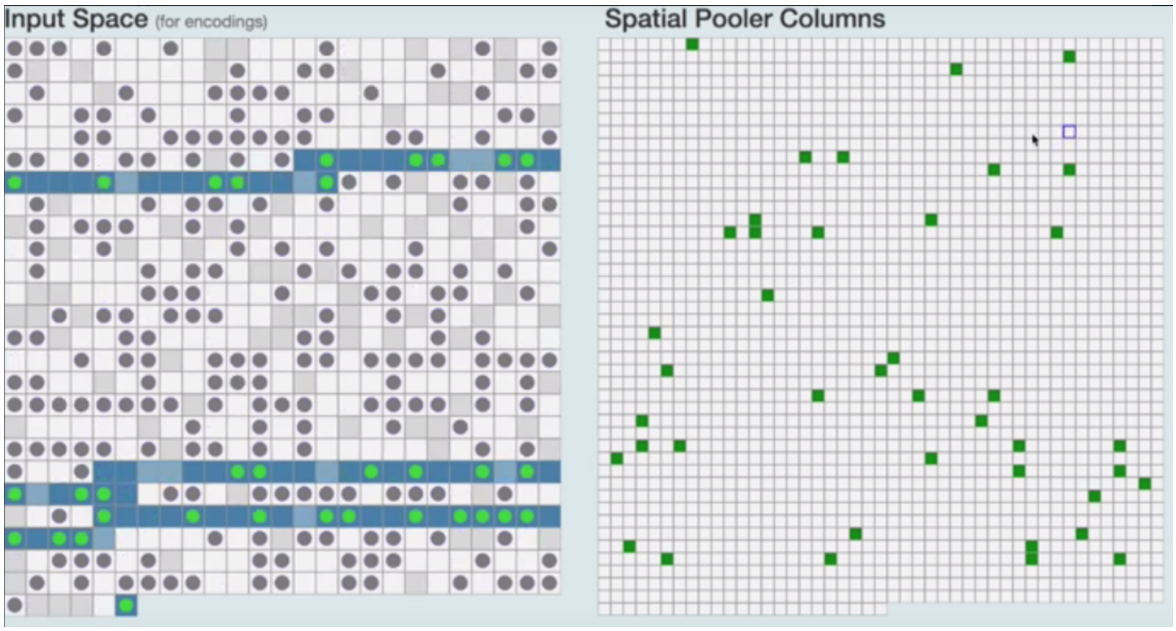


Figure 2.2: A visualization of a SP. Dots in the input space represent connections with a column. The blue squares are active bits in the encoding and the greyed out squares are bits the selected column can never connect with. The green squares are active columns, while the green dots are connections that overlap with the active bits in the input space. Image from [4]

2.3.4 Temporal Memory

The Temporal Memory is the component that learns sequences of SDRs. It takes the sparse patterns created by the SP and finds the temporal context. Then it attempts to make predictions based on the current input in context to the previous input.

While the SP learns connections between columns and the input encoding, the Temporal Memory algorithm learns connections between cells in the same layer. When a cell is activated it will establish connections with the previously active cells. This allows it to enter a predictive state when the other cells activate again. Figure 2.3 visualizes a segment with active and predictive cells. If a column contains no cells in a predictive state the Temporal Memory algorithm will activate all the cells as it has no context to infer from. Conversely if a column contains cells in a predictive state, then only those cells will be activated. [30][22]

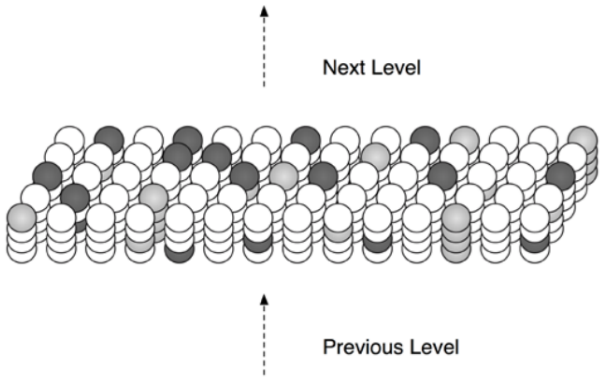


Figure 2.3: Example segment of a layer where light-grey cells are active and dark-grey cells are in a predictive state. Image from [30]

2.3.5 Swarming

A swarming algorithm is used to to distribute the workload when creating a model matching the data. Instead of fine-tuning hyperparameters, a swarm description is made. The swarm description defines what the input data looks like (i.e. types and headers), what to predict and what kind of prediction (i.e. temporal or spatial, multistep or single step). It can also include performance metric and min/max values, but the swarm will infer some values if not specified.[9]

Using the swarm description N workers will build and test possible models in a distributed manor storing the results in a shared database. The output will be the best scoring model in the form of an Online Prediction Framework. [9]

The swarm is divided into mini swarms and when finding scalar parameter values each mini swarm utilizes a Particle Swarm Optimization to find the optimal values. The search space is dynamic and will try to narrow it down to parameters yielding the best result.[9]

2.4 Data Set Considerations

The provided data set used in this thesis was provided by a bank and is used in accordance with a signed Non-Disclosure Agreement. Details regarding any contents of the data is hereby restricted and will only be expressed in general terms.

Throughout the research process it became evident that even with the number of transactions existing in the provided data set, it proved insufficient for testing purposes. As a response there was generated a data set.

2.4.1 Structure

The provided data set contains anonymous transaction records over a four year period. The records had numerous details but the only pertinent information for use in this paper are date, amount and which store the purchase was made. One of the contributions made are pertaining to churn prediction. A specific use-case is how major grocery chain stores cycle through discounts and sales in order to retain a good customer relationship value. With this angle in mind, the data set was filtered on five big Norwegian grocery chains (Coop, Rema, Kiwi, Bunnpris, Meny). Only transactions made to either one of those are retained in the customers transaction history. Information such as specific regions within Norway or internationally and site location of the stores are features that are currently not considered for practical considerations, but could be utilized in future developments of this approach.

The generated data set contains transactions occurring every 3rd and 4th day, alternating. Additionally for each transaction the amount varies between 400 and 600, functioning as arbitrary transaction amounts. This was done in order to create a set with enough transactions for testing purposes. The generated data set has the added benefit of consisting of a larger number of transactions so performance regarding data size was also gauged using this data set.

The individual approaches detailed in Chapter 3 perform additional alterations to the resulting data set in order to fit the respective solutions. Further details regarding those considerations are made in separate subsections related to preprocessing.

Part II

Contributions

Chapter 3

Proposed Solution

3.1 Proposed solution / algorithm

The Hierarchical Temporal Memory solution proposed in this research was implemented using the Numenta Platform for Intelligence Computing (NuPIC) API developed by Numenta[8][7], and using their Hot Gym prediction example[2] as a foundation.

3.1.1 The basic algorithm

1. The swarm description is written, outlining what fields of the data should be used, what to predict and how many steps into the future shall be predicted.

2. Then the swarm description is used to establish the model parameters. Different permutations of the possible components and variables are tested, and the best result is returned for further use.

3. The model parameters from the swarming are used to create an HTM model with the appropriate encoders and scalar parameters.

4. The input data is sent to the model where they are encoded to SDR and passed on to the SP. The SP makes connections to the input space and updates the columns in its layer. The active columns form a SDR which is passed on to the Temproal Memory to establish the temporal context of the

input. In the proposed solution the input data is transaction amount, days since last transaction and a timestamp.

5. The Temporal Memory updates the cells in the columns from the SP by looking for cells in a predictive state. Activated cells will form a connection to previously active cells in order to enter a predictive state the next time those cells are activated. Cells activated from a predictive state constitutes a prediction of what is likely to happen next. The output is the predicted next SDR to enter the input space. In the proposed solution the prediction is the predicted number of days until next transaction.

Figure 3.1: Diagram of the HTM architecture [23]. HTM Sequence Memory is synonymous with Temporal Memory in this case, and the Classifier decodes the SDR into a normal representation.

3.1.2 Discussion of design issues

An important factor in time series forecasting and sequence prediction is how many steps into the future the model should predict. A weather forecast that can only tell the weather of next day is not as useful as one that can predict the next week. And a highly inaccurate forecast for the next month will likely not be very helpful. The same can be said for churn prediction. A model might be highly accurate at predicting if a purchase will happen the next day, but will not account for vacations or other temporary changes in the transaction pattern. Aggregating the data into larger time frames could alleviate the problem to some degree, but would also remove some of the information. Predicting whether a transaction will happen within a given time frame would yield higher confidence predictions, than predicting specifically when the transaction will happen. If the data only contains transactions the model can predict the amount of time until the next purchase, instead of whether or not a purchase happens a given time frame. While this Time To Event method prevents predictions until an event has happened, it was chosen for its similarity to WTTE-RNN and its potential flexibility among different markets.

While the current NuPIC implementation of HTM can take several input variables, the predicted output may only consist of a single variable. This is one of the limiting factors as the predicted Time To Event (TTE) then has to be a single value, not a range or tuple.

When measuring performance there are a few different metrics to consider. While Mean Square Error (MSE) or Root Mean Square Error (RMSE) are commonly used. The HTM implementation uses Mean Absolute Percentage Error (MAPE) internally. MAPE is a popular choice as a measure of prediction accuracy when forecasting, but the relative results could serve as an abstraction and RMSE was chosen when measuring the overall performance of the proposed solution. Absolute error is used in the figures when visualizing how the predictions compare to the ground truth.

3.1.3 Algorithmic Enhancements

There are two operations done during pre-processing that significantly modifies the data.

Firstly, it is noticeably common for the store naming conventions to change and differ throughout the data set. In order to avoid identifying one store as multiple different stores a SequenceMatcher was used. It compares two strings and output a similarity ratio between 0.0 and 1.0. For instance *"abba"* and *"abca"* would return with a similarity ratio of 0.75. Certain changes (i.e. shortening part of the address) were more prevalent than others, and a difference in amount of characters would affect the similarity ratio of shorter strings more than longer strings. However, different thresholds were tested and a similarity ratio over 0.7 appeared to cover most of the occurrences.

Secondly, each record in the data set is originally a transaction, and as such a customer may have multiple records for a single day, at a single store. In order to make the processing easier each day is an aggregation of all the transactions a customer performed that day.

3.1.4 Discussion of the Parameter Space

There are a couple of parameters to consider when making the swarm description; *metricWindow* and *swarmSize*. The first defines the sample size when calculating

the error metric. It is usually set to 1000. The latter is how exhaustive the swarming is. It can either be *small*, *medium* or *large*. A small swarm will run quickly and is best suited for early testing. A large swarm can take a lot of time, but should provide the best model.

3.2 Prototype

Initial testing of HTM consisted of a simple scenario in order to provide an idea of what to expect from HTM sequence prediction and familiarity with the NuPIC API [7][8] developed by Numenta. The model received dates and predicted a float representing the transaction amount of the purchases that day.

3.3 Valuation of Contribution

By predicting the number of days until next purchase, the HTM could in theory learn patterns indicative of a decreasing frequency of transactions. Individual stores could combine that information with their own thresholds to proactively deploy measures to prevent churn.

3.4 WTTE-RNN

Throughout the literature review it became apparent that when trying to choose which of the state-of-the-art approaches to use as a comparison metric it was not feasible to find something mathematically equivalent so the decision was to find something conceptually comparable. The Weibull Time To Event Recurrent Neural Network (WTTE-RNN) returns α (alpha) and β (beta) values for a Weibull distribution that can be used to find the probability of an event occurring within a number of time steps [35]. Using this probability, the analyst needs to assert the mathematical correlation between the probability of an event and the churn potential of a customer.

The WTTE-RNN requires a purposeful structure of the training and test data set partitions. The goal is to train on data that has been labeled with time to the next event in order to train the model to recognize patterns within the customers

transaction history. As mentioned previously multiple transactions occurring per day are consolidated as one transaction for that day.

	amount	date
0	400.0	2010-01-07
1	600.0	2010-01-11
2	400.0	2010-01-14
3	600.0	2010-01-18
4	400.0	2010-01-21

Figure 3.2: The image shows the initial DataFrame extrapolated from the generated data set

To set up a given transaction history, one customers transaction history for a given store, i.e. “Coop” is extrapolated from the data set into a Pandas DataFrame. For reference, in Figure 3.2 an excerpt of the DataFrame from the generated data set is shown. The presented records show a timestamp paired with a purchase amount. Within the provided set, only completed transactions are recorded, so for a given day with no transactions there are no records. In order to effectively get a time-to-event labelled data set, days with no records need to be appended. As shown in Figure 3.3, the DataFrame is reindexed on the first and last days of the existing record to create a complete record.

	amount
2010-07-01	400.0
2010-07-02	0.0
2010-07-03	0.0
2010-07-04	0.0
2010-07-05	600.0

Figure 3.3: The image shows the reindexed DataFrame from the generated data set

The WTTE-RNN trains on labelled data, so the now reindexed dataframe has every amount greater than zero replaced with a binary one to represent an event occurring, while each amount representing no purchases remains as zero to represent a non-event , this is shown in Figure 3.4.

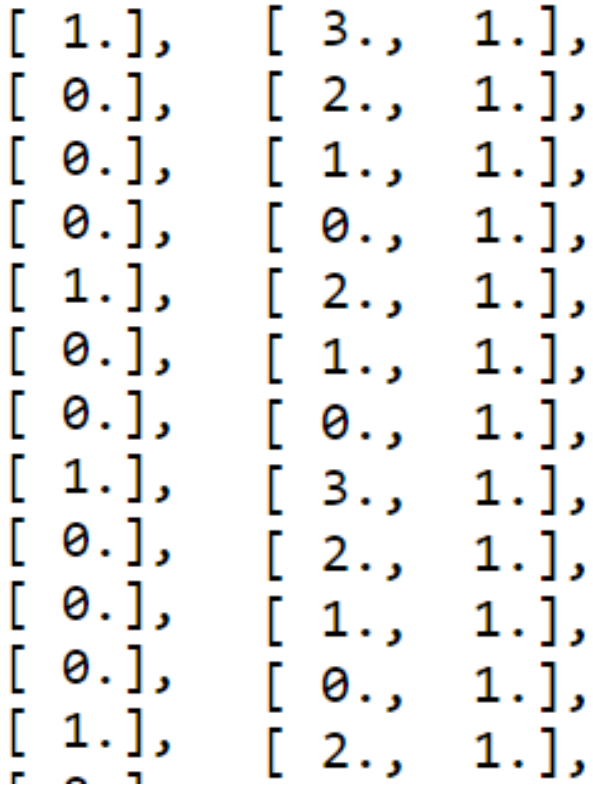
2010-07-01	1.0
2010-07-02	0.0
2010-07-03	0.0
2010-07-04	0.0
2010-07-05	1.0

Figure 3.4: The image shows the DataFrame from the generated data set with events instead of amounts

As shown in Figure 3.5b, comparatively to Figure 3.5a the data is displayed in a TTE manner where if the event is observed in the sequence it is shown. This illustrates a right-censoring in the temporal data generated.

The generated data consists of 34559 total transactions after reindexing. This is split into 80/20 training/testing partitions pre-processed according to the aforementioned requisitions.

The RNN structure is set up according to practical examples specified in [6] by the author of the proposed WTTE-RNN [35]. For all experiments performed the same model architecture was maintained.



(a) x values (b) y values

Figure 3.5: The above figures display the right-censored data structure of the input to the WTTE-RNN

Chapter 4

Experiments

The experiments were conducted for single customers, for single stores in order to simulate the amount of data a store manager or company might have access to.

4.1 HTM

4.1.1 Early Testing

The initial HTM model was defined for a more simplistic scenario where the model would receive consistent input.

The input data was sequential dates, and the model was defined to predict the sum of the transaction amounts for each day. Days without a transaction was included as 0.0 NOK. It was set to predict five days into the future.

Figure 4.1 shows a graph of the results after 1600 lines of input. It is still not very proficient at predicting the correct transaction amount. However, it appears to have learned some of the transaction patterns as the frequency of the predicted transactions partially matches with the frequency of the actual transactions.

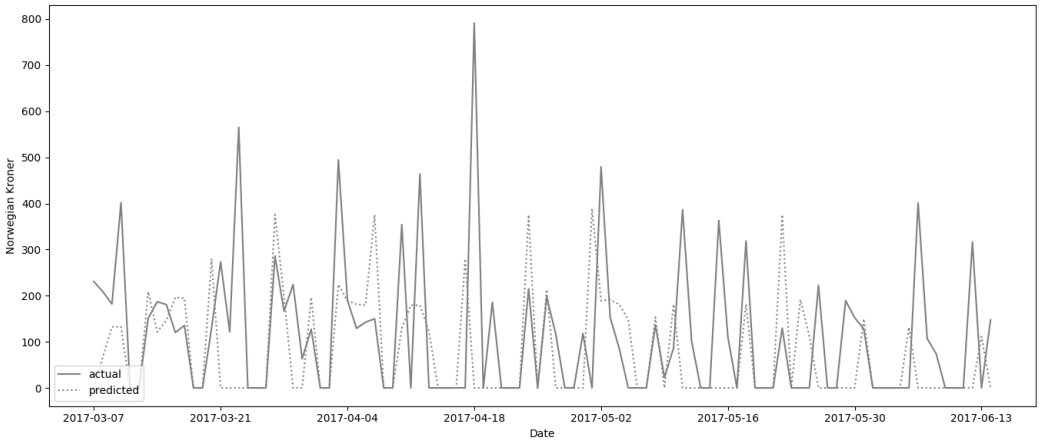


Figure 4.1: Graph of the early HTM results for predicting transaction amount. The prediction is five days into the future.

4.1.2 Time To Event (TTE) Prediction

Every data point sent to the model contains a timestamp, transaction amount and the number of days since last transaction, and the objective of the model is to predict the number of days until next transaction. The SP should establish the spatial context when constructing SDRs from the encoded; date, transaction amount and days since last transaction. And the Temporal Memory should learn the temporal context between the SDR patterns it receives from the SP.

The HTM model starts by copying the input it receives because it is yet to establish any temporal context, but over time it starts outputting predictions of its own. In theory if given enough data, it should eventually learn the spatio-temporal purchasing patterns of a customer and be able to predict the number of days until next transaction.

The following figures are all illustrating predictions of real transactions for one specific customer. The swarming was done on the data from the customer, and thus the model parameters were extrapolated from details such as minimum and maximum values of the data it will run on.

Figure 4.2 compares the prediction to the actual TTE after 300 lines of input. It is already starting to pick up on weekends and correctly predicts three days until next transaction multiple times. It is visible by the right-shifted dotted line that the model is still occasionally copying the input, which indicates that more input is necessary to make contextual predictions.

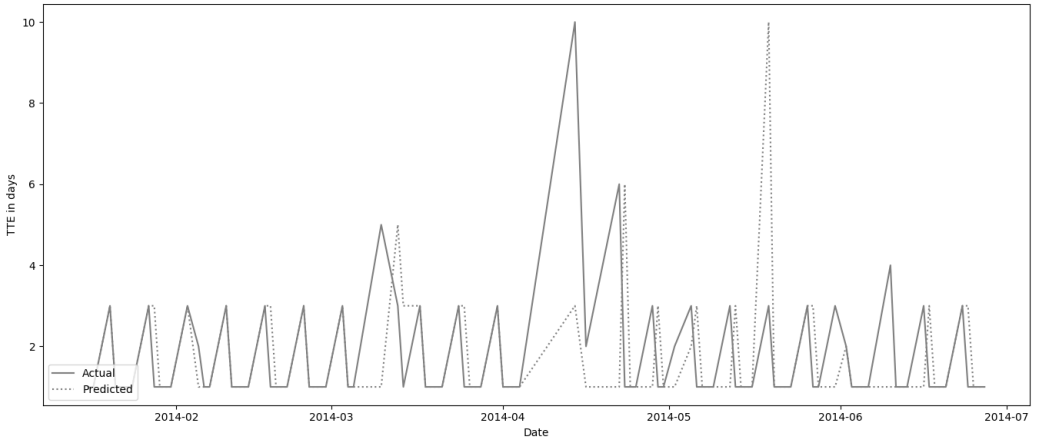


Figure 4.2: Graph displays HTM prediction results of TTE after 300 lines of input.

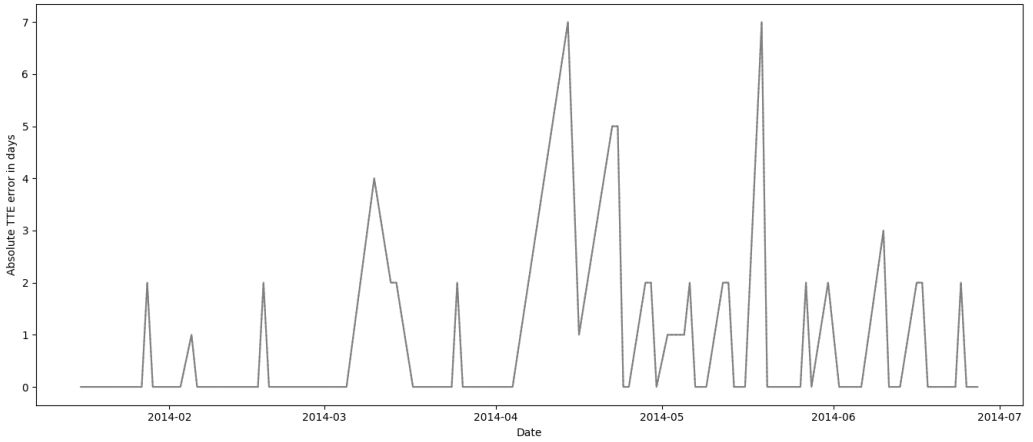


Figure 4.3: Graph displays absolute error from the HTM prediction results of TTE after 300 lines of input.

After 600 lines of input the model has become more proficient at predicting daily transactions with a regular pause over the weekend. It is however not learning the more irregular longer pauses or the two-day pauses in the transactions. Feeding the model data for common vacation periods might make it able to predict breaks in the transactions like the eight-day pause shown in Figure 4.4.

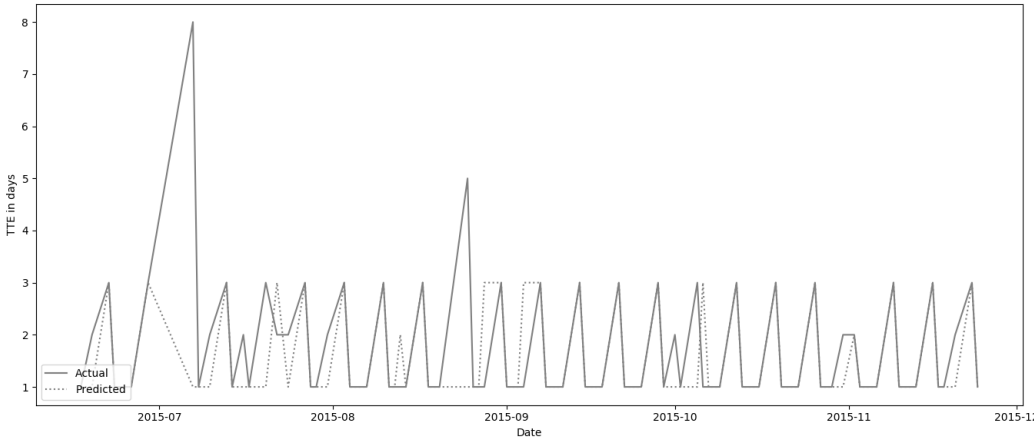


Figure 4.4: Graph displays HTM prediction results of TTE after 600 lines of input.

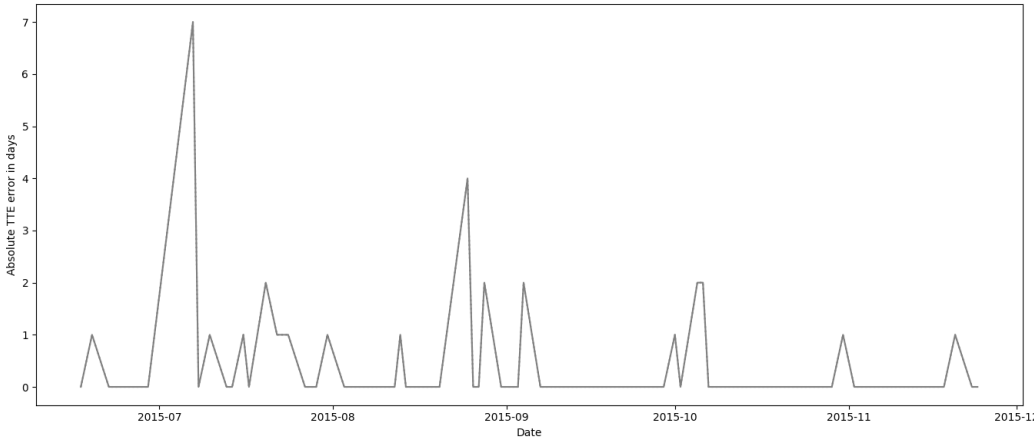


Figure 4.5: Graph displays absolute error from the HTM prediction results of TTE after 600 lines of input.

Figure 4.6 displays how the HTM is predicting against the actual TTE at the end of the available transactions for the customer. After 900 lines of input the model is still copying some of the input, or predicting incorrectly. However, it has started to predict outside of the regular pattern, daily transactions interrupted by the week-end.

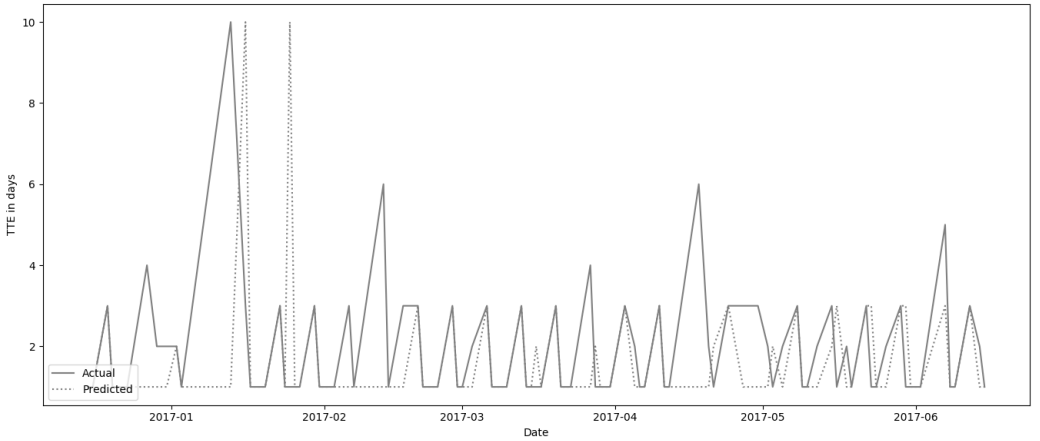


Figure 4.6: Graph displays HTM prediction results of TTE after 900 lines of input.

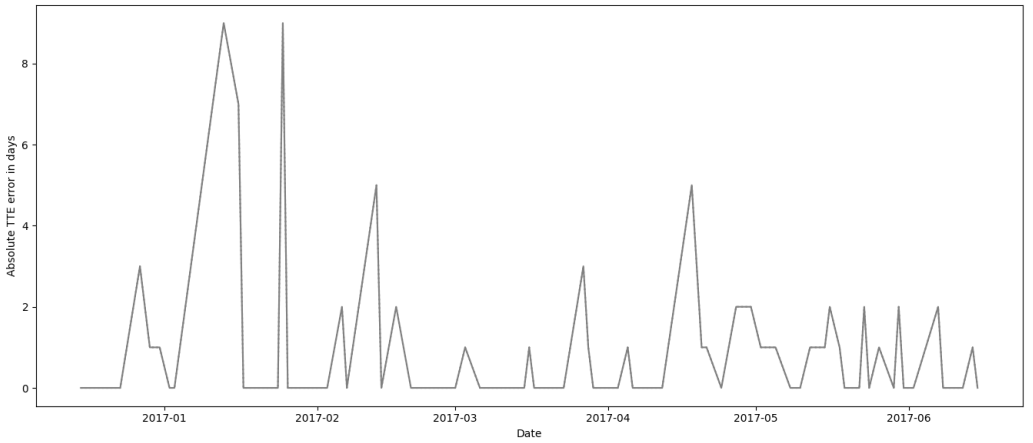


Figure 4.7: Graph displays absolute error from the HTM prediction results of TTE after 900 lines of input.

The previous figures illustrated the results when the swarming was performed on the input data, therefore the model parameters were chosen for that specific customer. Figure 4.8 shows how a model performs when the swarming is done on unrelated data. In this case the data is from another customer with less transaction days and larger breaks in the purchasing patterns. It is apparent that the model still learns about the daily transactions with weekend breaks.

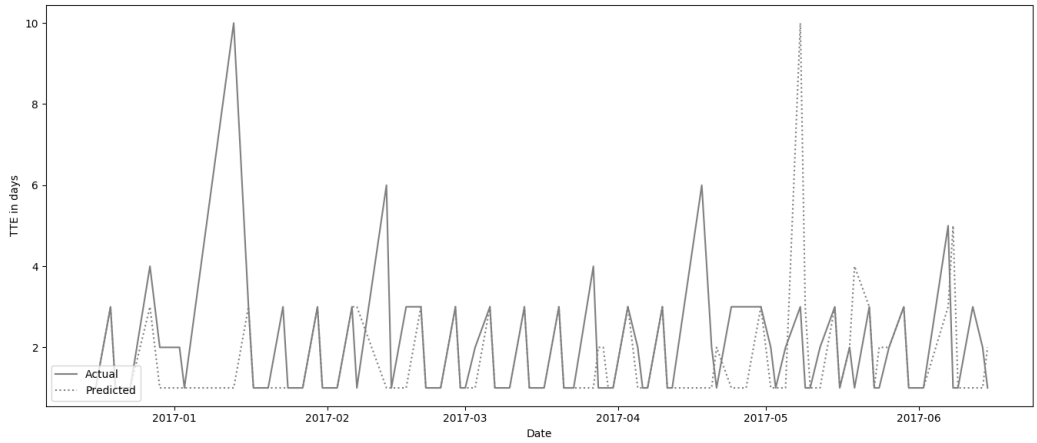


Figure 4.8: Graph displays HTM prediction results of TTE after 900 lines of input. Swarmed on different customer

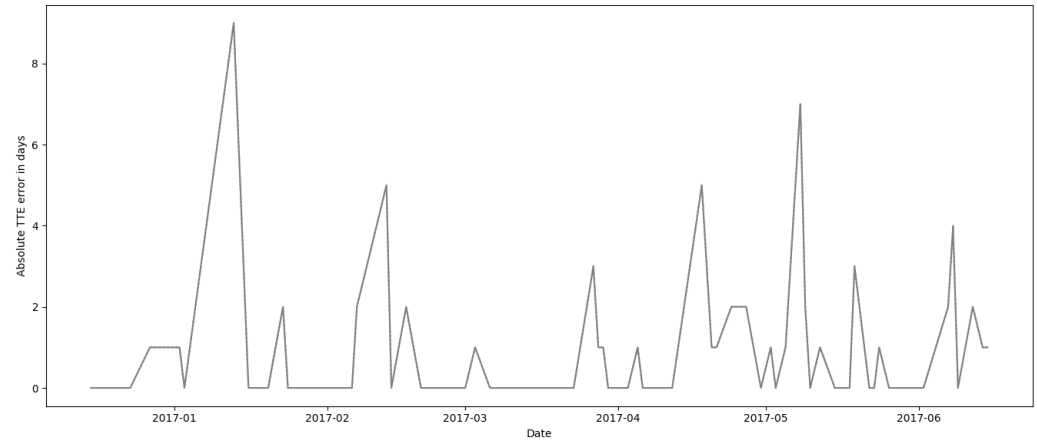


Figure 4.9: Graph displays absolute error from the HTM prediction results of TTE after 900 lines of input. Swarmed on different customer

While it might not be immediately visible from Figure 4.7 and Figure 4.9 which model is better, the RMSE results are a bit more conclusive. Table 4.1 shows how the model with parameters from swarming on data from a different customer outperforms the other. This difference is likely the result of more conservative predictions. In Figure 4.6 the model can be seen predicting a TTE of 10 days twice in the span of the graph, while the other model only predicts one TTE of 10 days in Figure 4.8.

After *k* lines	Swarmed on input data	Swarmed on different data
100	**1.708088**	1.796894
200	2.651883	**1.941066**
300	2.339471	**1.768150**
400	2.439988	**2.128562**
500	2.247450	**1.978735**
600	2.089589	**1.861112**
700	1.990846	**1.767931**
800	2.016969	**1.859569**
900	2.031855	**1.869472**

Table 4.1: RMSE after k records.

Although the swarming data affects the models ability to predict, the input data affects the predictions to a much larger degree. If the input data exhibits nothing but a clear, repeating pattern the predictions quickly become accurate. Figure 4.10 displays a segment of the absolute error of the model when the input data only consists of a repeating pattern of alternating transactions every third and fourth day. As the data is overly simplistic the results should only be seen as an indicator or baseline for what the model is capable of. At the point of Figure 4.10 the RMSE was 0.432771 and trending downwards.

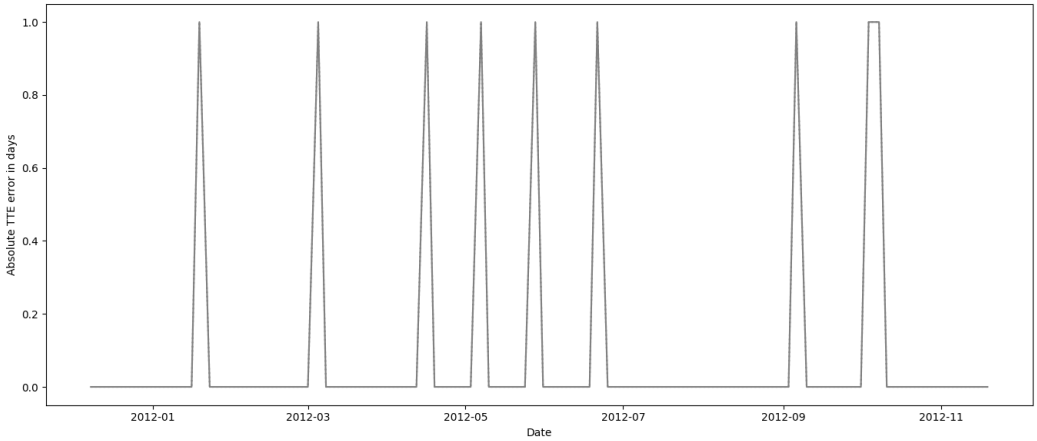


Figure 4.10: Graph displays absolute error from the HTM prediction results of TTE after 300 lines of input from generated data.

Learning the general pattern in the data is important for a churn prediction model, but it also needs to adapt to new changes in the data in a timely manner. Otherwise the model will fail to indicate that a customer is churning. Figure 4.11 illustrates how the model behaves when the data changes from the old pattern to the pattern in the generated data. It struggles a bit for the first few transactions before it adapts to the new range of TTE. The transition might not be as clean in a real scenario, and the new pattern will most likely be more noisy, but the results are still indicative of HTMs ability to adapt to new data. As Figure 4.12 demonstrates, the model has not yet learned the new pattern, but after a short spike in absolute error it is predicting in the right range of values.

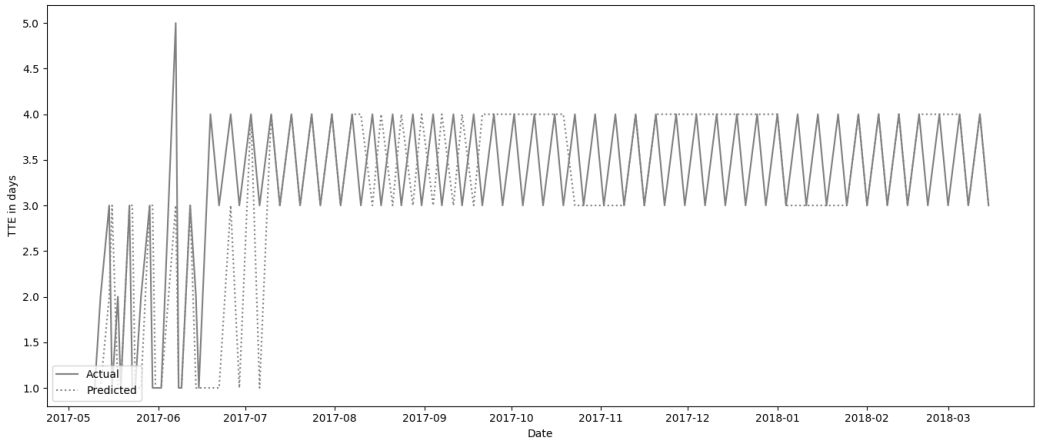


Figure 4.11: Graph displays HTM prediction results of TTE after 900 lines of input data changes to a new pattern.

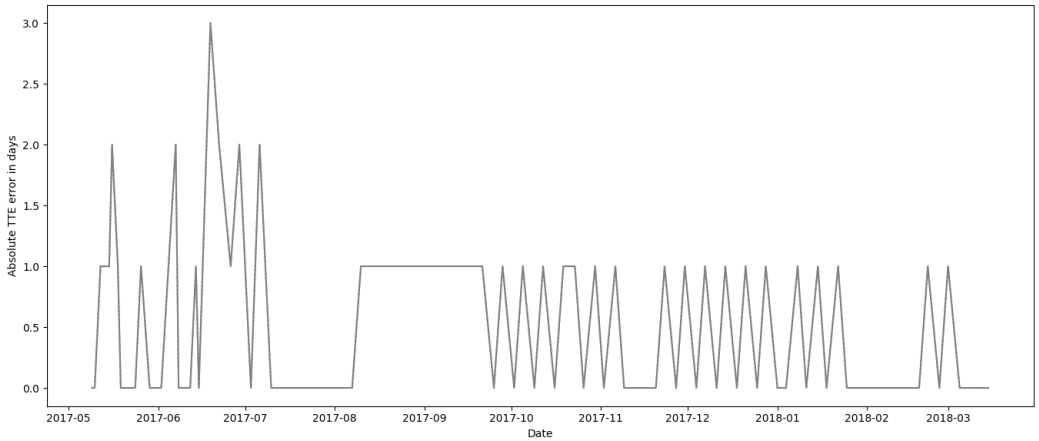
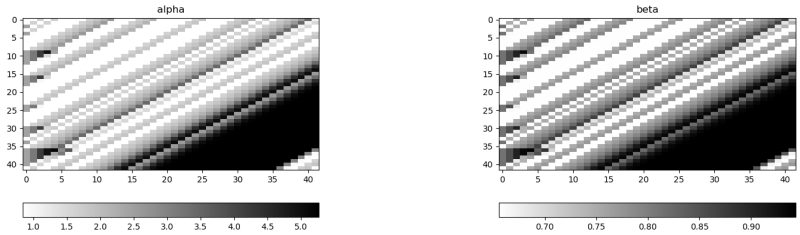


Figure 4.12: Graph displays absolute error from the HTM prediction results of TTE after 900 lines of input data changes to a new pattern.

4.2 WTTE-RNN Testing

When testing the WTTE-RNN it trained on right-censored data. The network received sequences of 42 time steps with information on how many days had passed since last transaction and whether or not the next transaction is still censored. Several events may be present in those steps, and usually were in the data used for the testing. The network then outputs a tensor with alpha and beta values which are parameters for a weibull probability distribution. The alpha values indicates when the transactions are likely to happen and the beta values indicate the confidence. Figure 4.13 visualizes the values and gives an idea of the transaction pattern. The color spectrum does however not represent the minimum and maximum values of the possible alpha and beta values.



(a) Alpha values (b) Beta values

Figure 4.13: The above graphs display the corresponding alpha and beta values calculated from a customer in the provided data set

Figure 4.14 illustrates the CMF calculated using the predictions made by the WTTE-RNN on the provided data set with a real transaction record. Although not a guaranteed outcome, it is a likely representation of a realistic scenario where the probability of a transaction happening within x amount of days are not a linear function. The probability does not reach 1.0 as it is possible a transaction will not happen within twelve days.

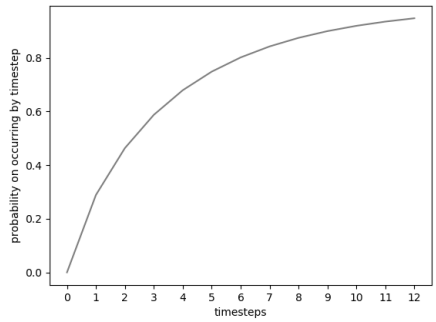
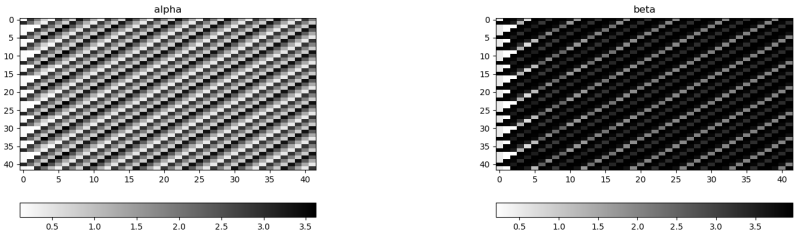


Figure 4.14: The graph displays the CMF for the calculated alpha and beta values. It reaches a close probability of an event occurring within the displayed timesteps.

Figure 4.15 visualizes the alpha and beta values after training and predicting on the generated data set, which solely contains a clear, simple, repeating pattern. The transaction pattern is clearly visible in the gradients of Figure 4.15a.



(a) Alpha values (b) Beta values

Figure 4.15: The above graphs display the corresponding alpha and beta values calculated from a the generated data set

The CMF illustrated in Figure 4.16 predicts a transaction will happen in one or two days, which indicates that the prediction is made two days after a transaction as the network has only seen transactions three and four days apart.

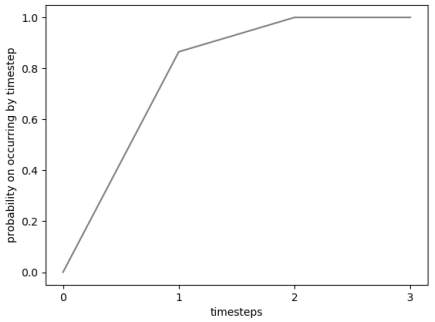


Figure 4.16: Graph displays the calculated CMF from the corresponding alpha and beta values, it shows that according to the alpha and beta values, the next event occurs within two days with high probability, and 3rd day guaranteed.

Chapter 5

Conclusion and further work

5.1 Summary of Results

Using a timestamp, transaction amount and number of time steps since last transaction, a HTM model is able to learn a general repeating pattern in the transactions. It is also able to adapt when presented with a new pattern with a different range of TTE. While it still needs time to learn the specific pattern, it is able to quickly change to the new range. The data used when swarming for model parameters affects the models performance, but not to a significant degree if the data is in the same paradigm as future input.

Using a sequence of events and non-events, WTTE-RNN shows capabilities in recognizing patterns for non-contractual transaction data. The implementation predicts a distribution that provides a likelihood for the respective predictions. Having an approach that explains its results intuitively is an improvement to existing state of the art. The proposed predictions are clearly influenced by the temporal nature of the data, but can also incorrectly represent the occurrence of the next event. With more available data, especially with more apparent patterns, the approach can provide a prediction with greater likelihood.

5.2 Conclusion

A robust non-contractual churn prediction model should be able to predict a changing pattern in the customers purchasing behaviour. By predicting a decreasing frequency in transactions a company may take proactive measures to avoid customers from churning.

This research has explored the use of HTM to predict TTE as a way to find customers with a declining purchasing behaviour. When provided streaming online data the HTM model is able to identify and learn a common pattern. After appending generated data to real customer data, the result show the model is capable of adapting to changes. This indicates that HTM may be used to predict potential non-contractual churners.

A HTM solution for churn prediction is also quite robust in the sense that switching the data for a different market should provide a model suitable for the different transaction patterns. There are no tweaking of hyperparameters necessary for the model to work. The features proposed also require very little pre-processing and the solution could be applied to any type of store with transaction data for their customers.

A limitation to predicting TTE, as proposed, is its sensitivity to the frequency of transactions. If the transactions occur frequently, the model will only predict a few time steps into the future, which might be fine considering a frequent customer is unlikely to churn. However, in a market with a more sparse transaction pattern, the model will update less often as it needs new events to make the next predictions. Comparatively, a WTTE-RNN is at an advantage as it is trained to predict on right-censored data. It will use the fact that x amount of time steps have passed without observing any transactions to provide parameters for the weibull probability distribution.

While WTTE-RNN can be considered more difficult to implement than HTM, even if only for the additional pre-processing, it produces more malleable results. The results are malleable in the sense that the resulting probability distribution may be utilized in a variety of ways without retraining the network. As demonstrated the results may be used to calculate a CMF representing the probability of a transaction taking place within a number of days. A probability distribution will also inherently indicate when the next transaction is likely to happen, as well as the projected probability. It might even indicate that it is more likely for a transaction not to happen.

In conclusion, using HTM to predict TTE is a viable solution for non-contractual churn prediction, that may be applied to different markets without any tuning of hyperparameters. And WTTE-RNN can provide more practical results, although it requires a more complex setup.

5.3 Further Work

5.3.1 More features

Utilizing more of the information involved in a transaction should improve the results. While the HTM uses a date encoder to keep the temporal context of when the transaction happens, the WTTE-RNN could still benefit from knowing what weekday, month or season the transaction happens. Providing the WTTE-RNN with the transaction amount might also increase its ability to provide accurate predictions.

A feature both solutions might benefit from is what product group the transaction is for. For instance, using a grocery store as example, it might be beneficial to include if the products purchased are perishables or otherwise. Some products might indicate a shorter period until next transaction. Including such a feature would require development of an encoder able to process it.

5.3.2 Clustering customers

Clustering customers based on their transaction patterns could lead to more accurate predictions. By training the WTTE-RNN on more specialized data for a group of customers the network could provide more specialized predictions. Exposing the HTM to transaction patterns related to a cluster of customers with similar behaviours could make it more responsive to variations in the data when predicting. By providing the HTM model with more data it will also have more of a chance to learn the patterns within the data.

References

[1] Geospatial tracking: Learning the patterns in movements and detecting anomalies. (White Paper) Available at https://numenta.com/assets/pdf/whitepapers/Geospatial%20Tracking%20White%20Paper.pdf.

[2] Nupic, one hot gym prediction. Available at https://github.com/numenta/nupic/tree/master/examples/opf/clients/hotgym/prediction/one_gym.

[3] Rouge behavior detection: Identifying behavor anomalies in human generated data. (White Paper) Avialable at https://numenta.com/assets/pdf/whitepapers/Rogue%20Behavior%20Detection%20White%20Paper.pdf.

[4] Spatial pooling: Learning (episode 8), 2016. Available at https://youtu.be/rHvjykCIrZM.

[5] Wtte-rnn - less hacky churn modeling, 2016. Available at https://ragulpr.github.io/2016/12/22/WTTE-RNN-Hackless-churn-modeling/.

[6] Implementation of wtte-rnn, 2017. Available at https://github.com/ragulpr/wtte-rnn/.

[7] Nupic - github, 2018. Available at https://github.com/numenta/nupic.

[8] Nupic 1.0.4 api documentation, 2018. Available at http://nupic.docs.numenta.org/1.0.4/index.html.

[9] The swarming algorithm nupic 1.0.4 documentation 2018, 2018. Available at `http://nupic.docs.numenta.org/stable/guides/swarming/algorithm.html`.

[10] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[11] Jae-Hyeon Ahn, Sang-Pil Han, and Yung-Seop Lee. Customer churn analysis: Churn determinants and mediation effects of partial defection in the korean mobile telecommunications service industry. *Telecommunications Policy*, 30(10):552 – 568, 2006.

[12] Amal M Almana, Mehmet Sabih Aksoy, and Rasheed Alzahrani. A survey on data mining techniques in customer churn analysis for telecom industry. *International Journal of Engineering Research and Applications*, 45:165–171, 2014.

[13] Plamen Angelov. *Autonomous learning systems: from data streams to knowledge in real-time*. John Wiley & Sons, 2012.

[14] Wai-Ho Au, K. C. C. Chan, and Xin Yao. A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE Transactions on Evolutionary Computation*, 7(6):532–545, Dec 2003.

[15] Leo Breiman. *Classification and Regression Trees*. Routledge, 1984.

[16] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[17] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[18] Linlin Chao, Jianhua Tao, Minghao Yang, Ya Li, and Zhengqi Wen. Long short term memory recurrent neural network based multimodal dimensional emotion recognition. In *Proceedings of the 5th International Workshop on Audio/Visual Emotion Challenge*, AVEC '15, pages 65–72, New York, NY, USA, 2015. ACM.

[19] 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.

[20] J. T. Connor, R. D. Martin, and L. E. Atlas. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, 5(2):240–254, Mar 1994.

[21] Kristof Coussement, Dries F. Benoit, and Dirk Van den Poel. Improved marketing decision making in a customer churn prediction context using generalized additive models. *Expert Systems with Applications*, 37(3):2132 – 2143, 2010.

[22] Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. Continuous online sequence learning with an unsupervised neural network model. *Neural computation*, 28(11):2474–2504, 2016.

[23] Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. The htm spatial pooler—a neocortical algorithm for online sparse distributed coding. *Frontiers in Computational Neuroscience*, 11, 2017.

[24] Walter J. Ferrier, Ken G. Smith, and Curtis M. Grimm. The role of competitive action in market share erosion and industry dethronement: A study of industry leaders and challengers. *The Academy of Management Journal*, 42(4):372–388, 1999.

[25] Joao Gama. *Knowledge discovery from data streams*. CRC Press, 2010.

[26] Dileep George and Jeff Hawkins. A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 3, pages 1812–1817. IEEE, 2005.

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

[28] A. Graves, A. r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013.

[29] Christian Grönroos. A service quality model and its marketing implications. *European Journal of Marketing*, 18(4):36–44, 1984.

[30] J. Hawkins, S. Ahmad, S. Purdy, and A. Lavin. Biological and machine intelligence (bami). Initial online release 0.4, Available at `https://numenta.com/resources/biological-and-machine-intelligence/`, 2016.

[31] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735, 1997.

[32] Kyoung jae Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1):307 – 319, 2003. Support Vector Machines.

[33] Zachary Chase Lipton. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019, 2015.

[34] Jesper Ljungehed, Kevin Smith, and Viggo Kann. Predicting customer churn using recurrent neural networks, 2017.

[35] Egil Martinsson. Wtte-rnn : Weibull time to event recurrent neural network. Master's thesis, Chalmers University Of Technology, 2016.

[36] M. C. Mozer, R. Wolniewicz, D. B. Grimes, E. Johnson, and H. Kaushansky. Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry. *IEEE Transactions on Neural Networks*, 11(3):690–696, May 2000.

[37] 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.

[38] E.W.T. Ngai, Li Xiu, and D.C.K. Chau. Application of data mining techniques in customer relationship management: A literature review and classification. *Expert Systems with Applications*, 36(2, Part 2):2592 – 2602, 2009.

[39] Parag C. Pendharkar. Genetic algorithm based neural network approaches for predicting churn in cellular wireless network services. *Expert Systems with Applications*, 36(3, Part 2):6714 – 6720, 2009.

[40] Mahardhika Pratama, Edwin Lughofer, and Dianhui Wang. Online real-time learning strategies for data streams for neurocomputing. *Neurocomputing*, 262:1 – 3, 2017. Online Real-Time Learning Strategies for Data Streams.

[41] Werner J. Reinartz and V. Kumar. On the profitability of long-life customers in a noncontractual setting: An empirical investigation and implications for marketing. *Journal of Marketing*, 64(4):17–35, 2000.

[42] David Rozado, Francisco B Rodriguez, and Pablo Varona. Optimizing hierarchical temporal memory for multivariable time series. In *International Conference on Artificial Neural Networks*, pages 506–518. Springer, 2010.

[43] Chris Rygielski, Jyun-Cheng Wang, and David C. Yen. Data mining techniques for customer relationship management. *Technology in Society*, 24(4):483 – 502, 2002.

[44] N. I. Sapankevych and R. Sankar. Time series prediction using support vector machines: A survey. *IEEE Computational Intelligence Magazine*, 4(2):24–38, May 2009.

[45] Wouter Verbeke, David Martens, Christophe Mues, and Bart Baesens. Building comprehensible customer churn prediction models with advanced rule induction techniques. *Expert Systems with Applications*, 38(3):2354 – 2364, 2011.

[46] Yaya Xie, Xiu Li, E.W.T. Ngai, and Weiyun Ying. Customer churn prediction using improved balanced random forests. *Expert Systems with Applications*, 36(3, Part 1):5445 – 5449, 2009.

[47] G.Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159 – 175, 2003.

Appendices

A Code

`https://github.com/JoneBakkevig/jkbmm`

UiA University of Agder
Master's thesis
Faculty of Engineering and Science
Department of ICT