

Churn Prediction Methods Evaluation and Implementation for Telecom Industry

by

Yingwen Ren

B.Sc., Simon Fraser University, 2018

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

© **Yingwen Ren 2021**
SIMON FRASER UNIVERSITY
Summer 2021

Copyright in this work is held by the author. Please ensure that any reproduction
or re-use is done in accordance with the relevant national copyright legislation.

Declaration of Committee

Name: Yingwen Ren

Degree: Master of Science

Thesis title: Churn Prediction Methods Evaluation and Implementation for Telecom Industry

Committee: **Chair:** Mo Chen
Assistant Professor, Computing Science

Ke Wang
Supervisor
Professor, Computing Science

Jian Pei
Committee Member
Professor, Computing Science

Ke Li
Examiner
Assistant Professor, Computing Science

Abstract

With the rapid growth in the telecom market, there is an emerging trend to focus on customer retention, which is a critical factor for designing future customer incentive strategies to help a company manage customer relationships. Our main contribution is to build an effective churn prediction system for a telecom company providing real-time communication to predict whether a customer may cease to do business with the company, i.e., stop using the service provided by the company to make phone calls. Due to the dynamic market environment, developing such a system is challenging, as it should not involve frequent retraining processes, leading to a high computational cost. Many different techniques are available to identify customers who are most likely to leave, however, which technique is the most suitable and applicable in practice is not clear because the performance of prediction methods depends heavily on the characteristics of the data. In our thesis, we implemented and evaluated two methods, namely MLP (Multilayer Perceptron) and WTTE-RNN (Weibull Time To Event Recurrent Neural Network), and the model evaluation is based on accuracy and computational cost. We conducted experiments on the real-world dataset containing customer call activity records, experimental results demonstrate that the model performance of MLP is better than the WTTE-RNN, achieving a higher AUC, precision, and Recall. Considering the computational cost, the WTTE-RNN takes more time than the MLP as the WTTE-RNN needs to be retrained, it cannot be directly applied for new data. Furthermore, a detailed feature engineering process was presented in our project, especially how to extract temporal call behavior from raw data. A user-friendly interface was implemented, in order to let users better use our churn prediction system.

Keywords: churn, telecommunication, temporal patterns, multilayer perception, weibull time event recurrent neural network

Acknowledgements

This project would not have been possible without the help and cooperation of many people. I would like to thank the people who helped me directly and indirectly in the completion of this project work.

First and foremost, I would like to express the deepest appreciation to my supervisor, professor Ke Wang, who has provided excellent guidance, encouragement, inspiration, constant and timely support throughout my Master studies, and I learned a lot from him during this past one and half year studies. Without his guidance and persistent help, this thesis would not have been possible.

I would also like to thank all the faculty members in the Dept. of Applied Science and my colleagues in the Database & Data Mining lab for their support in various aspects.

Last but not least, I would like to express my gratitude to our cooperator, a telecommunication company M800 Limited, their guidance and assistance have been of a great value for the completion of this work.

Table of Contents

Declaration of Committee	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Tables	vii
List of Figures	viii
1 Introduction	1
1.1 Contributions	1
1.2 Outline	2
2 Literature Review	3
3 Methods	5
3.1 Granularity Approach	5
3.2 Multilayer Perceptron (MLP)	5
3.2.1 Abstract Definition of MLP	5
3.2.2 How to use MLP as Churn Model	7
3.3 Weibull Time To Event Recurrent Neural Network	7
3.3.1 Abstract Definition of WTTE-RNN	8
3.3.2 How to Use WTTE-RNN as Churn Model	9
3.4 Performance Metrics	10
4 Feature Engineering	12
4.1 Data Understanding	12
4.2 Data Preparation	12
4.2.1 Features	12
4.2.2 Data Transformation	13

5	Experiments and Results	19
5.1	Experimental Setup	19
5.1.1	Hardware	19
5.1.2	Software	19
5.2	Model Details	19
5.2.1	Multilayer Perceptron	19
5.2.2	Weibull Time to Event Recurrent Neural Network	20
5.3	Model Comparison	20
5.3.1	Accuracy	20
5.3.2	Efficiency	22
5.4	Summary	24
6	System Development	25
6.1	Architecture Overview	25
6.2	System Implementation	25
6.2.1	Apache Spark	25
6.2.2	Tensorflow	25
6.2.3	Flask	27
6.3	Churn Prediction Interface	27
7	Conclusions and Future Works	33
7.1	Conclusions	33
7.2	Future Works	34
	Bibliography	35

List of Tables

Table 4.1	Selected Attributes from the Activity Data	13
Table 4.2	Extracted Features Describing Customer Behavioral Information . . .	14
Table 4.3	Sampled activity data	14
Table 4.4	Sampled customer data for MLP model	15
Table 4.5	Sampled customer data for WTTE-RNN model	15
Table 5.1	Current Window (Months): Comparison of MLP and WTTE-RNN on Accuracy	21
Table 5.2	Current Window (Weeks): Comparison of MLP and WTTE-RNN on Accuracy	21
Table 5.3	Comparison of MLP and WTTE-RNN on time	23
Table 6.1	Recent saved models	29

List of Figures

Figure 3.1	A simple MLP, adapted from [1]	6
Figure 3.2	In each step i we output estimated Weibull parameters $\alpha_i^{(j)}$ and $\beta_i^{(j)}$ for each customer j , from Martinsson, 2016 [2]	10
Figure 4.1	A comparison of using sliding window approach for MLP and WTTE-RNN	17
Figure 5.1	The process of model training and prediction	22
Figure 6.1	System overview	26
Figure 6.2	Interface overview	27
Figure 6.3	Define active customer and churn customer with a , b , C and F	28
Figure 6.4	Select data on a saved model	29
Figure 6.5	Recent saved predictions	30
Figure 6.6	Zoom in or out on the monthly activity data	31
Figure 6.7	Boxplot distribution for customer monthly call count	32

Chapter 1

Introduction

The telecommunication industry is becoming increasingly saturated, and it is becoming easier for customers to switch between carriers. Loss of customers equals to the loss of future revenue and loss of initial investment cost spent on those customers, and acquiring new customers becomes more difficult because of the increasing intense competition and saturation of the market. As reported in the literature, the cost of acquiring a new customer may be higher than that of retaining an existing customer by as much as 700%, and an increase in customer retention rates by a mere 5% may increase profits by 25% to 95% [3]. Facing this challenge, the company needs to focus on how to reduce customer churn. Churn is defined as a customer quitting the usage of the service [4]. To deal with churn problem, churn management strategy is used to make necessary reactions to retain customers who are at the high risk of leaving [5]. For example, providing with business incentives strategies. However, companies do not have enough resources for the entire customer base and not each customer prepares to leave. Therefore, companies should know what factors lead to customer churn and what are the counter actions that the company can take to prevent such a situation [6]. Moreover, knowing in advance before customers really churn will provide companies more room to react and more chances of success to retain their customers [5].

1.1 Contributions

In our thesis, we presented a collaboration research with a telecommunication company called M800 Limited which provides real-time communication. One of their products, a cross-platform messaging app, offers phone calls and text messaging services. We aimed to build an accurate and efficient churn prediction system to predict whether a customer may cease to do business with the company. In our project, we implemented and compared two types of churn predictions methods, one is used to estimate customer's churn probability (Multilayer Perceptron), another is used to estimate when churn happens (Weibull Time-To-Event RNN) proposed by Martinsson, using call details records (CDR) of this app that contain details of customer activities e.g., call duration, caller city and start time. By ana-

lyzing CDR of this app, we can determine each customer's future churn probability, a chance of a customer to leave in the future, so that M800 Limited can have better understanding of their customers in order to retain their customers who are more likely to churn.

To determine which method is most suitable and applicable in practice, we evaluated the mentioned churn methods on both accuracy and efficiency. In addition, we need to solve two challenges. The first one is that in the telecommunication market, there are two types of customers, named contract-based customers and pre-paid customers. A contract-based customer has to sign a contract with the company for a period of time, e.g., a month or a year, while pre-paid customers could use the service at any time as long as they have calling credits remaining in their accounts. Predicting churning probability for pre-paid customers is more difficult than contract-based customers. Different from a contract-based customer who is bound by contracts, a signal of a contract-based customer's leaving is that he does not extend the contract when the contract renewal date is reached. However, in a pre-paid situation, customers can leave the service without time constraints, and there does not exist a criterion to define churn for pre-paid customers. Even though we focused on pre-paid customers in our project, one of churn methods used in our project that is used to estimate customer's future churn probability can also be applied for contract-based customers. Given an explicit churn definition for contract-based customers, for example, a contract churner is defined as a customer does not renew his contract in 15 days, we can estimate customer's churn probability by analyzing the underlying patterns between his call behavior and pre-defined churn labels. Our second challenge comes from the retraining process. Usually, for a method that models sequential data, once new time series data come in, the old model may become invalid, so a retraining process is required. But if the retraining process is conducted too frequently, the computational cost could be high, especially for a system with a large amount of data.

Finally, we also considered customer's temporal call patterns that is defined as segment of data in a time sequence, we assume it might be related with his churning signal. And an analysis of the detection of long-range temporal dependencies is also presented in our study. Furthermore, the feature engineering process for our churn methods is demonstrated explicitly in our thesis.

1.2 Outline

Chapter 2 reviews how previous machine learning algorithms are applied to churn prediction. Chapter 3 provides an explicit description of selected techniques that have been used in this research. In Chapter 4, feature engineering is demonstrated thoroughly. An analysis of the results is presented in Chapter 5. Chapter 6 describes the system implementation of the churn-prediction system and demonstrates the use of this system step by step. Finally, Chapter 7 provides a conclusion and some ideas for future works.

Chapter 2

Literature Review

As we described previously, it is important for a company to know whether a current customer would leave the plan in the near future, that's why building models to predict the probability of customer churn and classify such behaviors are in demand. Many approaches were applied to predict churners in the telecom industry, the major research approaches focused on using single method like data mining to extract knowledge, and others focused on comparing several methods to predict churners [7].

In recent years, many researchers defined the churn problem as the classification problem [8, 9, 10, 11]. Au et al. discovered classification rules and used it to classify a record to one of the predefined class whose label is not known [8]. Lu et al. proposed a churn prediction model using the boosting algorithm, where customers were divided into two clusters based on the weights assigned by the boosting algorithm [10]. The logistic algorithm is trained for each customer group to predict the likelihood of a customer will churn sometime in the future [10]. Neural networks are also applied for customer churn prediction. In the literature, the Multilayer Perceptron Neural Network (MLP-ANN) is one of the most commonly applied neural network models for churn problem [12]. In [13], a MLP was proposed to predict customer churn in one of the leading Malaysian's telecommunication companies. The results were compared with other popular algorithms such as Support Vector Machines, Decision Trees and Linear Regression. The obtained results have proven the supremacy of MLP over the statistics models in prediction tasks [13]. In [14], authors compared the performance of MLP with Backpropagation to other popular algorithms such as Support Vector Machine, Decision Trees, and Linear Regression using an open access dataset. They found that MLP networks and Decision Tress achieved the best performance for their case, while SVM came very close [14]. Similar applications of using MLP for churn problem are also introduced in several researches [15, 16].

Most of previous researches did not present the feature engineering phase or the process of building features from raw dataset, they relied on given features provided either by telecom companies or open access datasets from the Internet. In our project, the feature engineering phase was thoroughly presented.

Recent researchers started to focus not only on the churn probability but also on the time period when the churn happens [17]. Deep neural network for sequential data is applied to estimate churn time. In [18, 19, 20], Recurrent Neural Network was employed in predicting a sequence of distribution parameters for a random series of “time to next event” (i.e., inter-arrival times). RNN was used to model sequential data with complex temporal dependencies as well as non-linear associations. However, this method encounters a problem that we are uncertain about random arrival times, because this information is hidden in the future data which is said to be censored [21]. For example, during the observation window, a customer’s next call does not been observed. Thus, researchers started to consider survival analysis, a method is used to analyze an event of interest, e.g., a customer’s next call event [22]. Mariia and Nataliia explored and compared the use of semi-parametric Cox Proportional Model and parametric Weibull and Log-normal survival models for churn problems. A recent paper described a framework makes use of survival analysis and RNN to model inter-arrival times for each customer called WTTE-RNN (Weibull Time-To-Event RNN) [2]. At each timestep, we either observe complete information when an arrival has been observed or incomplete information about a non-arrival event. This induces a conditional distribution on the partially observed “inter-arrival time”, which allows us to maximize a likelihood function to obtain optimal RNN parameters [21]. The detailed description is described in Section 3.3. However, in [2], Martinsson did not evaluate the WTTE-RNN on a real-life dataset.

We discussed the related works proposed by the recent researchers in this chapter, and we learned that different machine learning techniques are widely used for churn prediction. However, we do not know which method the most suitable and applicable in practice because the performance of a prediction method depends heavily on the characteristics of the data [23, 24, 25]. Thus a comparative analysis was needed for analyzing which churn prediction model should be used best. Two methods were investigated and evaluated on accuracy and efficiency using a real-world dataset containing customer call activities, one is Multilayer Perceptron that is the most commonly applied neural network models for churn prediction, another is the Weibull Time-To-Event RNN that takes advantage of the survival analysis and recurrent neural network to model recurrent call events [2].

Chapter 3

Methods

A small selection of tricks that address the problem of churn prediction are presented in this chapter. First of all, we want to introduce the granularity approach that is a method to obtain temporal call patterns, and a demonstration of how to apply Multilayer Perceptron (MLP) and Weibull Time-To-Event RNN (WTTE-RNN) to churn problem is also presented in the following.

3.1 Granularity Approach

We proposed a conjecture that customer’s temporal patterns, a term describing segment of data in a whole time sequence, have an impact on the model performance, especially when we aggregated customer’s call behavior over shorter time periods. In order to derive temporal patterns, we proposed the “Granularity” approach, summarizing the raw data into different time intervals i.e., days, weeks or months. Granularity is related to the level of details or summarization of the units of data, a lower granularity is equivalent to a higher level of details that describes the data. For example, if we use a 1-day “granularity”, then the raw data is aggregated over days.

3.2 Multilayer Perceptron (MLP)

MLP is one type of method to estimate churn probability. We choose to use MLP is to reduce computational cost as it is a method to approximate a function f that maps the input to the output value, and once a MLP is obtained, it can be applied to other months of data. And in order to derive temporal patterns which might be related to the signal of churning, we proposed the granularity approach.

3.2.1 Abstract Definition of MLP

In our project, a Multilayer Perceptron is used to learn the relationship between customer call features such that average call received per day, total number of calls received and

average duration, etc, which serve as inputs to the network, and churn labels such that 1 denotes churn and 0 denotes non-churn, which are designated as outputs of the network [26].

Suppose we have customer data that contain one record for each customer summarizing all activities of the customer. Let our data in the form $Z = (x_1, y_1), (x_2, y_2) \dots, (x_i, y_i) \dots (x_k, y_k) \in (X, Y)^k$, aka "training data", where x_i is the call features for a customer aka "input", y_i is the churn label for this customer aka "output" and k is the number of customers in the data. f denotes a hypothesis that maps the input to the output, it can also be regarded as the network output. W denotes parameters of f and $f(x_i; W)$ denotes the network output. Before talking about the details, we first introduce the architecture of the MLP. A MLP is made up of an arrangement of interconnected neurons with a simple

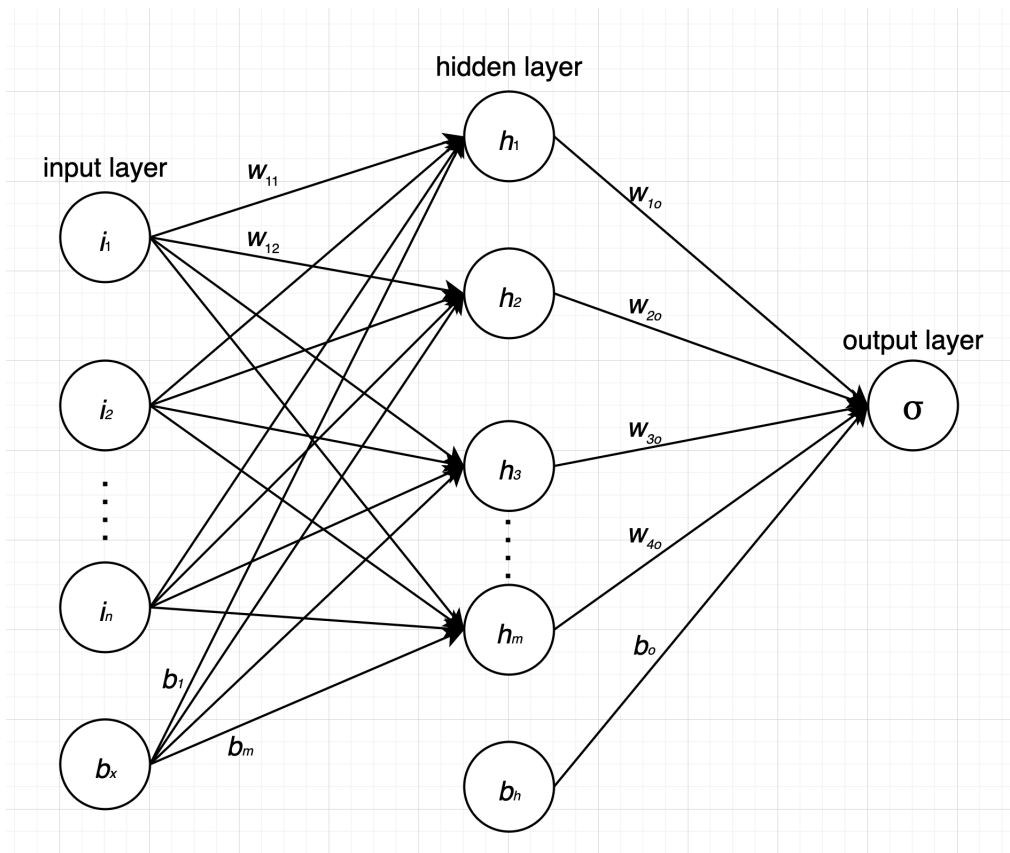


Figure 3.1: A simple MLP, adapted from [1]

activation function. A typical MLP consists of the input layer, output layer, and hidden layer (topological structure is as shown in Figure 3.1). Connections between neurons are represented as weights w . Each neuron consists of a summation function and an activation function. We use the process to produce model output for a single customer i , $i \in [1, k]$ as an example. The summation functions S_i sum up the product of neuron in the hidden layer h and weights w , and a bias b as shown in Figure 3.1, where w_{jo} is the connection weight

connecting hidden neuron j to output neuron o , $j \in [0, m]$, b_o is a bias weight and m is the number of hidden neurons. S_i can be described in the following:

$$S_i = \sum_{j=1}^m w_{jo} h_j + b_o$$

The output of the summation function will be the input of the activation function, the sigmoid function $\sigma(x) = (1 + e^{-x})^{-1}$, it is used to return a probability as to know how likely the customer belongs to churn class.

Therefore, the predicted probability for customer i produced by the model can be described in Formula 3.2.1:

$$f(x_i; W) = \sigma\left(\sum_{j=1}^m w_{jo} h_j + b_o\right)$$

Let $\mathcal{L}(f(x_i; W), y_i)$ be the loss of each instance $i \in [1, k]$ in the training data Z , the main problem is to find parameters W of f that minimize the total loss \mathcal{L} over the entire dataset Z .

$$\min_W \mathcal{L}(W, x, y) = \frac{1}{k} \sum_{i=1}^k \mathcal{L}(f(x_i; W), y_i)$$

Since our project is a binary classification problem, thus binary cross-entropy loss is used. The new functional optimization problem becomes:

$$\min_W \mathcal{L}(W, x, y) = -\frac{1}{k} \sum_{i=1}^k [y_i \log(f(x_i; W)) + (1 - y_i) \log(1 - f(x_i; W))]$$

3.2.2 How to use MLP as Churn Model

Once the MLP model has completed its training, a hypothesis f has already found its optimum parameters W . Suppose we want to estimate the probability of Customer A 's future leave. Given his call features x_i including aggregated call duration and aggregated call counts, we can attain his churn probability by calculating $f(x_i; W)$. If his

$$f(x_i; W) \geq 0.5$$

then Customer A is a potential churner. Because we use 1 denotes churn, a higher predicted value indicates a higher churn probability.

3.3 Weibull Time To Event Recurrent Neural Network

WTTE-RNN is one type of method to estimate time when churn happens, it is a framework that is used to model inter-arrival times. The WTTE-RNN models sequential data by minimizing a likelihood function that utilizes partial information, e.g., unobserved call

events [2]. Survival analysis is used to study the intervals between successive customer’s call events. The time to the next event from some position on this timeline is called waiting time. The main problem of the WTTE-RNN is to model this waiting time.

3.3.1 Abstract Definition of WTTE-RNN

Assume we have a time series dataset, containing multiple customers $j \in [0, N]$, and each customer has a sequence of observed call events at different timesteps $i \in [1, T]$, where T denotes the length of the time series. Let $x_{0:i}^{(j)}$ denotes sequence of call feature vectors from start until timestep i for customer j , $t_i^{(j)}$ denotes observed waiting times between timestep i and the next call event for customer j , $u_i^{(j)}$ denotes event indicator at timestep i indicating that whether for customer j ’s next call event occurs in this time series: $u_i = 1$ means customer’s next call event after timestep i is observed (**uncensored**), otherwise customer’s next call event after timestep i is not observed (**censored**). $R(x_{0:i}^{(j)}, s)$ is defined as the recurrent cumulative hazard function that is the integrated hazard rate over $[0, s]$ for $s \in [0, \infty]$ for input data $x_{0:i}^{(j)}$ at timestep i . The hazard rate is the conditional probability of a call event will happen at a given time point [2]. Here, the input data $x_{0:i}^{(j)}$ govern the shape of $R(x_{0:i}^{(j)}, s)$, and when using R for prediction, s represents the size of a future time window, and we want to figure out the probability of a customer’s future calls in this future time window [2].

The main problem is to find an optimum function R that maximizes the log-likelihood for waiting times $t_i^{(j)}$ for censored observation (i.e., an observation has not been observed) given the feature data $x_{0:i}^{(j)}$ observed until timestep i at each timestep i for each customer j . Note that we use $u_i^{(j)}$ to indicate whether customer j ’s next call event after timestep i is observed (uncensored) or not,

$$\max_{\hat{R} \in \mathcal{S}} \mathcal{L}(\hat{R}, t, u, x) = \sum_{j=0}^N \sum_{i=0}^T u_i^{(j)} \cdot \log[\hat{R}_s(x_{0:i}^{(j)}, t_i^{(j)})] - \hat{R}(x_{0:i}^{(j)}, t_i^{(j)}) \quad (3.1)$$

Here, \mathcal{S} is the solution space, R_s denotes the estimated recurrent hazard function evaluated at time s , which describes the instantaneous potential of a call event at time s given the customer j has not called before time s . In WTTE-RNN [2], the Weibull distribution is used to estimate the Hazard function R_s and R by fitting the data.

Before going into details, we want to first introduce the Weibull distribution and some characteristics of the Weibull distribution.

Weibull distribution Martinsson [2] chooses the Weibull distribution to model call observations because of several nice characteristics (namely, empirically feasible, expressive, which means it can take many shapes by adjusting its two parameters, location-scale transformation, and regularization mechanisms). The Weibull distribution has two parameters,

α and β . The scale parameter α is to pull the height of the probability distribution and the shape parameter β is equivalent to the slope of the line in the probability distribution, different values of β have different effect on the behavior of probability distribution.

Assume that the parameters $\alpha_i^{(j)}$ and $\beta_i^{(j)}$ of the Weibull distribution are the function of historical customer features $x_{0:i}^{(j)}$ at each timestep i for each customer j . The Weibull Hazard function R_t is the probability that an event will occur at time $t_i^{(j)}$ given that a customer does not call before time $t_i^{(j)}$:

$$\hat{R}_t(x_{0:i}^{(j)}, t_i^{(j)}) = \frac{\beta(x_{0:i}^{(j)})}{\alpha(x_{0:i}^{(j)})} \cdot \left(\frac{t_i^{(j)}}{\alpha(x_{0:i}^{(j)})}\right)^{\beta(x_{0:i}^{(j)})-1} \quad (3.2)$$

The Weibull Cumulative Hazard function R is the accumulation of the hazard in the interval $[0, t_i^{(j)}]$:

$$\hat{R}(x_{0:i}^{(j)}, t_i^{(j)}) = \left(\frac{t_i^{(j)}}{\alpha(x_{0:i}^{(j)})}\right)^{\beta(x_{0:i}^{(j)})} \quad (3.3)$$

And in the WTTE-RNN, Weibull parameters $\alpha_i^{(j)}$ and $\beta_i^{(j)}$ at each timestep i for each customer j are the outputs of the Recurrent Neural Network whose parameters are Θ . Thus, applying Formula 3.2 and 3.3 into Formula 3.1, the objective functions can be represented in the following, the goal of the WTTE-RNN is to maximize it with respect to Θ instead of $\alpha_i^{(j)}$ and $\beta_i^{(j)}$,

$$\max_{\Theta} \mathcal{L}(\alpha, \beta, t, u, x) = \sum_{j=0}^N \sum_{i=0}^T u_i^{(j)} \cdot \log \left[\frac{\beta(x_{0:i}^{(j)})}{\alpha(x_{0:i}^{(j)})} \cdot \left(\frac{t_i^{(j)}}{\alpha(x_{0:i}^{(j)})}\right)^{\beta(x_{0:i}^{(j)})-1} - \left(\frac{t_i^{(j)}}{\alpha(x_{0:i}^{(j)})}\right)^{\beta(x_{0:i}^{(j)})} \right] \quad (3.4)$$

From here we may understand that an artificial neural network is required as it can be trained using a differentiable loss function in order to attain an optimum Weibull parameters $\alpha_i^{(j)}$ and $\beta_i^{(j)}$ at timestep i for each customer j to maximize Weibull Loss for all customers. The process of using the RNN to learn Weibull parameters is shown in Figure 3.2.

3.3.2 How to Use WTTE-RNN as Churn Model

Once a WTTE-RNN is trained, RNN has already attained optimum parameters at each timestep i to obtain the maximum log-likelihood for all customers.

Suppose that we have a customer j 's call feature sequence $x_{0:i}^{(j)}$ before timestep i including aggregated call duration and aggregated call counts. Applying customer j 's sequential data until timestep i on a trained WTTE-RNN model, the RNN will produce the Weibull parameters $\alpha_i^{(j)}$ and $\beta_i^{(j)}$ at timestep i for this customer. Since we have already achieved Weibull parameters value $\alpha_i^{(j)}$ and $\beta_i^{(j)}$, and given a future time $t_i^{(j)}$ that is a time period,

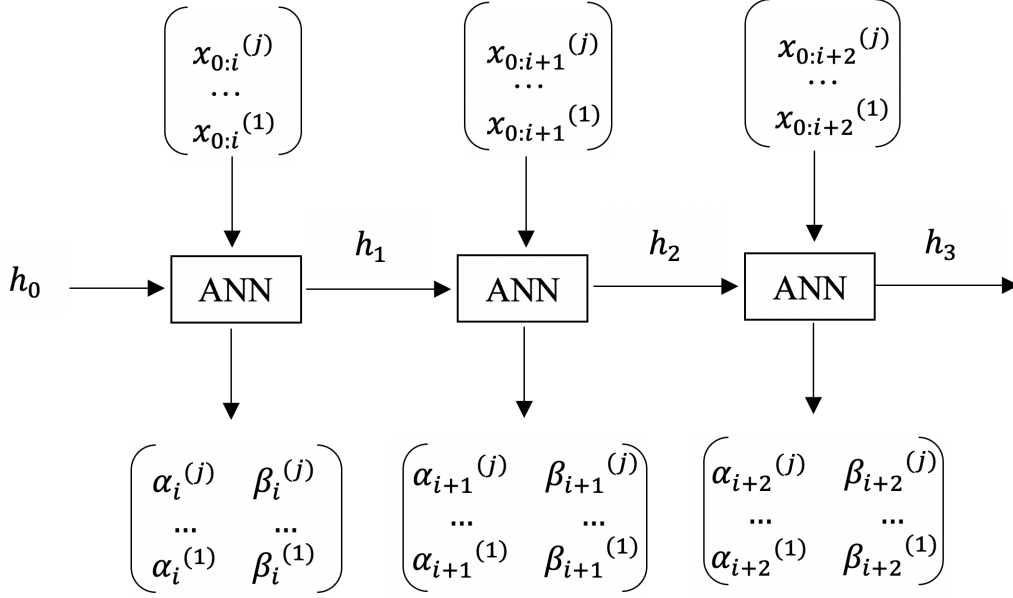


Figure 3.2: In each step i we output estimated Weibull parameters $\alpha_i^{(j)}$ and $\beta_i^{(j)}$ for each customer j , from Martinsson, 2016 [2]

we can attain his cumulative density function $F(t : i, j)$ (Formula 3.5) that is the call probability for customer j will call in the future $t_i^{(j)}$ time after timestep i .

$$F(t; i, j) = 1 - \exp\left[-\left(\frac{t_i^{(j)}}{\alpha_i^{(j)}}\right)^{\beta_i^{(j)}}\right] \quad (3.5)$$

Suppose the outputs of RNN at timestep i for customer j are $\alpha_i^{(j)} = 30$ and $\beta_i^{(j)} = 4$. If we want to know customer's probability of future call after timestep i over the next 30 days, thus future time period t is 30, using the CDF formula above, we can attain customer j 's probability of future call during the next 30 days:

$$F(30) = 1 - \exp\left[-\left(\frac{30}{30}\right)^4\right] \simeq 1 - 0.36 = 0.64 \quad (3.6)$$

If $F(30) < 0.5$, the customer will be predicted as a potential churner.

3.4 Performance Metrics

To have a quantitative measure on the performance of our model, three metrics, including Precision, Recall, and the area under the ROC Curve (AUC) are respectively used to evaluate the performance of the models established in this project. The following four measures

need to be calculated beforehand:

True Positive (TP): an outcome where the model correctly predicts the positive class.

True Negative (TN): an outcome where the model correctly predicts the negative class.

False Positive (FP): an outcome where the model incorrectly predicts the positive class.

False Negative (FN): an outcome where the model incorrectly predicts the negative class.

Precision Precision measures the proportion of the positive predictions that are actually correct, it is defined as follows:

$$precision = \frac{TP}{TP + FP}$$

Recall - True Positive Rate (TPR) Recall, a synonym for True Positive Rate (TPR), measures the proportion of positive cases that are predicted correctly, it is defined as below:

$$recall (TPR) = \frac{TP}{TP + FN}$$

False Positive Rate (FPR) False Positive Rate (FPR) measures the proportion of negative cases that are predicted as positive cases, it is defined as below:

$$FPR = \frac{FP}{FP + TN}$$

ROC Curve A commonly used metric to evaluate model predictive performance is the receiver operating characteristic (ROC) curve. ROC curve plots true positive rate (TPR) as a function of false positive rate (FPR). Furthermore, the area under a ROC curve is a standard metric that judges the overall performance of the classification model [27]. AUC ranges from 0 to 1, and a better model performance yields a higher AUC, for example, 1 denotes a model with no error, 0.5 denotes a random model, and < 0.5 denotes that a model more often makes wrong predictions [27].

Chapter 4

Feature Engineering

In this chapter, the feature engineering is presented explicitly. We start with the data understanding (Section 4.1), and then present features generated from the raw data and two data transformation methods used in the data preparation (Section 4.2).

4.1 Data Understanding

The data understanding phase deals with the collection and exploration of data to get insight into data, some basic information is gathered in this phase.

We used call activity data from M800 Limited as our working data, it is in a period of 12 months from 2017_01 to 2017_12, and it contains 17,353,973 call records and 227,749 customers, composed with “customer to customer” interaction records. Each activity record contains detailed call information between a caller and callee. We selected 13 attributes from activity data to build customer call behavior features, and Table 4.1 provides the details of selected attributes.

4.2 Data Preparation

In the data preparation phase, the raw data is transformed into a feature matrix. In the following, we will show the features that are generated from the raw data and explain how we transform customer activity data into customer data for the MLP model and WTTE-RNN model respectively.

4.2.1 Features

In order to describe call temporal patterns for a customer, the call behavior over different periods (e.g., days, weeks, or months) is summarized. The extracted features describing customer behavioral information from a time interval is demonstrated in Table 4.2.

Name	Description	Example
Caller	The caller identifier	phone number
Callee	The callee identifier	phone number
Start Time	The call start time	2018/01/02/ 12:30:23am
Duration	Call duration	120, the unit is in seconds
Caller City	The 2 letter city identifier for caller	NY
Callee City	The 2 letter city identifier for callee	NY
Caller Carrier	The carrier name of caller	Fido
Callee Carrier	The carrier name of callee	Fido
Caller Plat- form	The OS platform for caller	Android
Callee Plat- form	The OS platform for callee	Android
Success	An indicator of whether the call is successful, true or false	True
Type	The call type	ONNET or OFFNET, ONNET is used when caller and callee use the same carrier, OFFNET is used when caller and callee use different carriers
Bye Reason	Bye reason code	CANCEL, BYE or ERROR

Table 4.1: Selected Attributes from the Activity Data

4.2.2 Data Transformation

Fixed Granularity Approach

In order to extract temporal patterns, fixed granularity approach is applied to summarize the activity data over different time intervals (e.g., days, weeks, or months) for each customer. As stated in Chapter 3.1, granularity refers to the level of details or summarisation of the units of data. We use the same sample activity data (Table 4.3) for both MLP and WTTE-RNN methods. Table 4.3 represents activity data containing three customers' sequences of call activity records from 2017_01_01 to 2017_01_03. 1-day "granularity", in other words, data is summarized over days, is used as an example to demonstrate a comparison of applying fixed granularity approach on both MLP and WTTE-RNN.

Applied on Multilayer Perceptron The MLP uses the fixed granularity approach to generate customer data, containing one record for each customer summarizing all activities of the customer into fixed time windows. Thus the final feature matrix is indexed per customer, with a list of summarized call patterns over different time periods for each customer.

Feature Name	Description
avg duration	What is the average duration that a customer calls another customer over a time interval.
percentage of no-answered calls	What is the percentage of the calls that a customer calls another customer but no one answers over a time interval.
call count	What is the total number of calls that a customer calls another customer over a time interval.
unique set of cities a customer calls from	Which cities a customer calling another customer from over a time interval, it should be a list of city names, and the name of a city should be unique.
unique set of cities a customer calls to	Which cities a customer calling another customer to over a time interval, it should be a list of city names, and the name of a city should be unique.
unique set of carriers a customer used before	Which carriers a caller uses over a time interval, it should be a list of carrier names, and the name of a carrier should be unique.
unique set of platforms a customer used before	Which platforms a customer uses over a time interval, it should be a list of platform names, and the name of a platform should be unique.
unique set of success codes	Whether calls from a customer are successful over a time interval, it should be a list of success codes, and the value of a success code should be unique.
unique set of call types	Whether calls from a customer are ONNET or OFFNET over a time interval, it should be a list of call types, and the value of a call type should be unique.
unique set of reasons a customer ends calls	What reasons a customer ends calls over a time interval, it should be a list of call end reasons, and the value of a call end reason should be unique.

Table 4.2: Extracted Features Describing Customer Behavioral Information

Customer ID	Call Time	Duration (seconds)
1	2017-01-01 13:21:45	324
2	2017-01-01 13:48:12	362
1	2017-01-01 14:12:15	725
2	2017-01-01 19:24:01	1209
2	2017-01-02 08:45:30	180
3	2017-01-02 08:51:42	0
3	2017-01-02 13:26:55	318
3	2017-01-03 05:12:25	354

Table 4.3: Sampled activity data

For example, total number of calls received over a fixed time period, average duration over a fixed time period, percentage of no-answered calls over a fixed time period, etc.

Example Because we have a sequence of call observations from 2017_01_01 to 2017_01_03, customer’s calling behavior patterns will be summarized into three daily intervals, 2017_01_01, 2017_01_02 and 2017_01_03. The final matrix is shown in Table 4.4, it is indexed per customer, with a list of summarized call patterns over different time periods for each customer.

Let N denotes the number of customers, T denotes the number of daily intervals, and F denotes the number of the call behavior patterns such as summarized call duration and summarized call count. The generated data is represented as a multidimensional feature matrix $X_{N*(T*F)}$ (Table 4.4), where N equals 3, T equals 3, and F equals 2.

Customer ID	Duration 1 st interval (seconds)	Call Count 1 st interval	Duration 2 nd interval (seconds)	Call Count 2 nd interval	Duration 3 st interval (seconds)	Call Count 3 st interval
1	1049	2	0	0	0	0
2	1571	2	180	1	0	0
3	0	0	318	2	354	1

Table 4.4: Sampled customer data for MLP model

Applied on WTTE-RNN The WTTE-RNN uses the fixed granularity approach to generate time series data that is a series of customer call behavioral patterns ordered in time. Thus, the customer activity data will be aggregated at fixed times and indexed per customer at timestep $i = 0, \dots, n$, e.g., n denotes the number of timesteps.

Customer ID	Call Date	Call Count	Duration (seconds)	Observed Waiting Times t (days)	Event Indicator u
1	2017-01-01	2	1049	2	0
	2017-01-02	0	0	1	0
	2017-01-03	0	0	0	0
2	2017-01-01	2	1571	1	1
	2017-01-02	1	180	1	0
	2017-01-03	0	0	0	0
3	2017-01-01	0	0	1	1
	2017-01-02	5	318	1	1
	2017-01-03	1	354	0	0

Table 4.5: Sampled customer data for WTTE-RNN model

As it can be seen from the Table 4.5, in the data transformation phase for the WTTE-RNN model, apart from summarizing customer call patterns over days, it is also required

to derive observed waiting times t (days), indicating time to the next event, and event indicator u . We described the meaning of the event indicator in Section 3.3.1, when $u = 1$ then the observation has been observed (uncensored observation), otherwise the observation has not been observed (censored observation).

Example Suppose the **observation window** ends on 2017_01_03, if there is more than one call records at one timestep e.g., a day or a week, it is considered that the user has a call event at that timestep. We will use the above sample customer data (Table 4.5) as an example to show how to generate event indicator u and the observed waiting times t .

Event Indicator u : The event indicator u determines whether next call event has been observed or not.

- Censored observation: e.g., Customer 1 at timestep 2017_01_01, his next call has not been observed, thus his event indicator is 0 at timestep 2017_01_01.
- Uncensored observation: e.g., Customer 2 at timestep 2017_01_01, his next call is on the 2017_01_02, thus his event indicator is 1 at timestep 2017_01_01.

Observed Waiting Times t : The observed waiting times t determines the observed duration of time between current timestep and next event.

- Censored observation: e.g., Customer 1 at timestep 2017_01_01, because his next call has not been observed, thus we define his observed waiting days as days difference between end day of the observation window and timestep itself (observed waiting times is 2 at timestep 2017_01_01).
- Uncensored observation: e.g., Customer 2 at timestep 2017_01_01, because his next call has been observed, thus we define his observed waiting times as days difference between the next call event and timestep itself (observed waiting times is 1 at timestep 2017_01_01).

Let N denotes the number of customers, T denotes the number of daily intervals, and F denotes the number of customer features, including call behavior patterns (summarized call duration and summarized call count), observed waiting days t and event indicator u . The generated customer data is represented as a multinational feature matrix X_{N*T*F} (Table 4.5), where N equals 3, T equals 3, and F equals 4.

Sliding Window Approach

In the following, we want to compare the application of sliding window approach for both MLP and WTTE-RNN method. For the MLP, sliding window is used for churn labeling, and for the WTTE-RNN, sliding window is used for estimating the probability of a customer's future call in a time window. We use Figure 4.1 to demonstrate the differences in general.

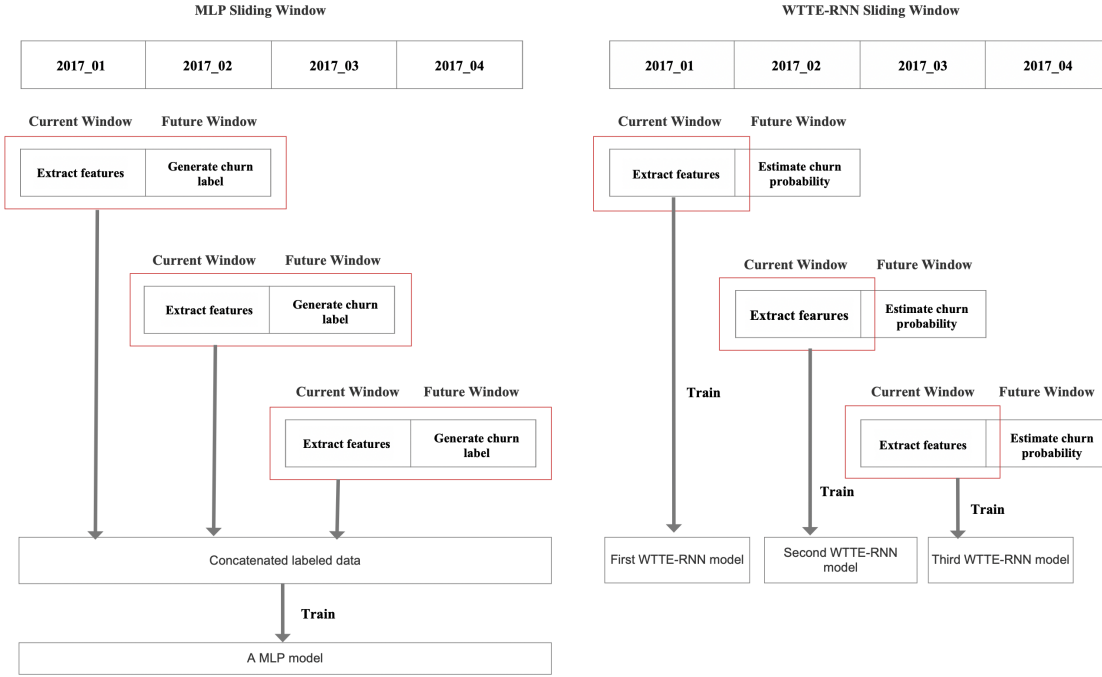


Figure 4.1: A comparison of using sliding window approach for MLP and WTTE-RNN

Function as labelling for Multilayer Perceptron As we can see from left graph in Figure 4.1, we split each window into Current Window C and Future Window F . C is used to extract customer features, F is used to extract churn labels, we keep customers whose total number of calls in C is at least a and label these customers whose total number of calls in F is less than b as “churner” otherwise “non-churner”. Suppose $C = 1$, $F = 1$, $a = 5$, $b = 0$, and the training data consists of the data in four consecutive months 2017_01, 2017_02, 2017_03, and 2017_04. If we use the first window containing 2017_01 and 2017_02 as an example: we derive features from customers who call at least 5 times in 2017_01 a.k.a active customers, and extract churn labels for those active customers from 2017_02, an active customer who does not call in 2017_02 will be labelled as a churner.

In the MLP sliding window approach, by sliding each $C + F$ window, one month at a time, we can slide the $C + F$ window three times, each window produces labeled data, and the final training data is the concatenation of generated labeled data.

Function as estimating probability for WTTE-RNN In the right plot in Figure 4.1, each window is also splitted into Current Window C and Future Window F . C is used for extracting time series data using the “granularity” approach, and F is used for estimating the probability for a customer to call in F months. As described in Section 3.3.2, once a WTTE-RNN model is trained, it can be used as a prediction model. Using the WTTE-RNN, we can look F steps into the future from end of C . Suppose we want to predict

the probability of customer j 's future call in F months, and s is the last time step in C . Given the estimated Weibull parameters $\alpha_s^{(j)}$ and $\beta_s^{(j)}$ that is the outputs from RNN, we can attained his cumulative density function (CDF), the probability of customer j will call within F , using the formula mentioned in section 3.3.2. A customer whose probability is less than 0.5 will be predicted as a churner. Thus in the WTTE-RNN, the sliding window approach is used to reformulate the problem of estimating the next event of interest into a binary classification problem where we ask whether customer's next call will happen or not in the F months. Using the generated churn labels of customers in C window for the MLP as the ground truth, we can evaluate the performance of the WTTE-RNN by comparing ground truth and estimated churn labels produced by the WTTE-RNN.

Suppose $C = 1$, $F = 1$. Use the right plot in Figure 4.1 as an example, by sliding each $C + F$ window, one month at a time, we can slide the $C + F$ window three times. In each window, we train a WTTE-RNN model using the customer call patterns derived from C , and use F to predict churn labels for customers in C . The overall model performance is the average performance among three models.

Chapter 5

Experiments and Results

In this chapter, we will present all experiments that been used to evaluate the models used in our project and their corresponding experimental results. Model comparison was based on the accuracy and efficiency. In terms of accuracy, we will evaluate our churn model from two different perspectives, one is from the impact of granularity level on model performance and the other is from the impact of current window's length on model performance. Efficiency will be measured by the time spending for both training and prediction phase. We applied the architecture of WTTE-RNN proposed by Martinsson that is posted on **GitHub** for our real-world telecom data to validate the theoretic works in his thesis.

5.1 Experimental Setup

5.1.1 Hardware

All the experiments were conducted in the context of a OS X system machine with 2.3GHz 8-core 9th-generation Intel Core I9 processor, 16GB 2666MHz DDR4 memory and 1TB SSD storage.

5.1.2 Software

We used Spark SQL as the data transformation engine, and the implementation is performed under a Python environment. We stored the transformed data on disk after the offline data transformation phase and retrieved the relative transformed data during the model training phase.

5.2 Model Details

5.2.1 Multilayer Perceptron

The model used in our experiment has two hidden layers with 64 neurons each, the RELU activation function is applied to both hidden layers without dropout. The model was trained with Adam optimizer using a learning rate of 0.001, batch size of 128, and the batch size

can be adaptively increased to a maximum of 512. The model was trained for 100 epochs, and early-stop is also applied in the model training phase.

5.2.2 Weibull Time to Event Recurrent Neural Network

We modified the architect used in [2] that is posted on the github repository <https://github.com/ragulpr/wtte-rnn>. The model used in our experiment has two GRU layers with 64 neurons each, followed by a dense output layer of dimensionality 2 with a custom activation layer to output Weibull parameters α and β . The model was trained with Adam optimizer using a learning rate of 0.01, batch size 200. The model was trained for 50 epochs, and to avoid over-fitting, two regularization methods dropout and early-stop were adopted in our experiments.

5.3 Model Comparison

In order to choose the most suitable model, we evaluated the model performance on both accuracy and efficiency.

5.3.1 Accuracy

The accuracy is measured by AUC, precision, and recall collected on the testing set. We conducted two experiments. One evaluates the impact of temporal granularity on the model performance.

In our experiment, granularity of 3-day, 5-day, 15-day and 30-day was used to evaluate whether customer’s behaviors at different granularity will affect the model performance. The other evaluates whether MLP and WTTE-RNN can capture the long term dependency in the data through the current window size C . We used two different Current Window setting, one is month-based current window, using 3-month, 4-month, and 5-month as the current Window size, another is week-based current window, using 3-week, 4-week, and 5-week as the current Window size.

The future window F is set to 1-month, following [28] that defines a churner as a customer without call events in the next 30 days. The performance of MLP and WTTE-RNN in terms of AUC, Precision, and Recall is shown in Table 5.1 (month-based current window) and Table 5.2 (week-based current window). The first two columns denote the settings of temporal granularity and current window size. For each of AUC, Precision, and Recall, the best performer in each setting is marked in bold face.

It can be observed from Table 5.1, as more days are aggregated (i.e., a lower granularity), the model performance gets worse, and this trend is observed on both MLP and WTTE-RNN. For example, the model performance on the monthly granularity is worse than on the daily granularity. For each granularity, we tested 3-month, 4-month, 5-month cur-

Granularity (days)	Current Win- dow (months)	MLP's AUC	WTTE's AUC	MLP's Precision	WTTE's Precision	MLP's Recall	WTTE's Recall
3	3	0.921	0.886	0.963	0.875	0.922	0.914
3	4	0.901	0.877	0.921	0.8	0.878	0.862
3	5	0.884	0.872	0.931	0.851	0.857	0.862
5	3	0.917	0.879	0.947	0.853	0.932	0.859
5	4	0.897	0.87	0.88	0.862	0.872	0.85
5	5	0.871	0.875	0.869	0.859	0.87	0.849
15	3	0.916	0.862	0.935	0.86	0.937	0.865
15	4	0.881	0.856	0.918	0.851	0.871	0.861
15	5	0.86	0.854	0.856	0.849	0.858	0.839
30	3	0.904	0.861	0.871	0.852	0.901	0.856
30	4	0.876	0.857	0.861	0.86	0.86	0.853
30	5	0.85	0.859	0.837	0.831	0.856	0.825

Table 5.1: Current Window (Months): Comparison of MLP and WTTE-RNN on Accuracy

Granularity (days)	Current Win- dow (weeks)	MLP's AUC	WTTE's AUC	MLP's Precision	WTTE's Precision	MLP's Recall	WTTE's Recall
3	3	0.8	0.736	0.76	0.732	0.67	0.681
3	4	0.808	0.802	0.787	0.748	0.672	0.687
3	5	0.815	0.782	0.794	0.75	0.701	0.69
5	3	0.809	0.623	0.799	0.747	0.665	0.643
5	4	0.813	0.746	0.83	0.771	0.667	0.647
5	5	0.774	0.784	0.817	0.78	0.689	0.651
15	3	0.768	0.737	0.75	0.699	0.62	0.625
15	4	0.802	0.766	0.783	0.73	0.632	0.629
15	5	0.804	0.779	0.789	0.741	0.653	0.64

Table 5.2: Current Window (Weeks): Comparison of MLP and WTTE-RNN on Accuracy

rent window sizes to study whether MLP and WTTE-RNN can learn long-range temporal patterns.

The capability of MLP to model temporal patterns is negatively impacted by the current window size, and this is true across all granularity studied. An explanation is that model prediction is more likely influenced by recent customers' call behavior activities, and as the current window size gets longer, more irrelevant information will be added and will decrease the model's ability. In the month-based current window setting, unlike the MLP model, the WTTE-RNN model shows similar results with different sized current windows. In fact, RNN tends to pay more attention to the recent past, thus the performance of the WTTE-RNN

is less affected by irrelevant remote past. However, this trend should be interpreted with caution because our current window size is in the unit of month. If the current window size is in a smaller unit such as week, the 5-week window may contain more relevant information than the 3-week window, so the trend may change. Thus we conducted another experiment using week-based current window.

The experiment results based on Current Window (weeks) can be seen from Figure 5.2, for both MLP and WTTE-RNN, the model performs better as the current window size gets larger. It verifies our conjecture that when we compare with smaller sized current window, for example, comparing 5-week Current Window with 3-week Current Window, model performance on a 5-week Current Window is better than on a 3-week Current Window. It is because that 5-week Current Window contains more information than 3-week Current Window.

5.3.2 Efficiency

Except comparing the model performance in terms of accuracy, we also evaluate their efficiency based on their time performance including duration on both training and prediction phase. We conducted our experiments on a one-year dataset, containing 17,353,973 call records and 227,749 customers.

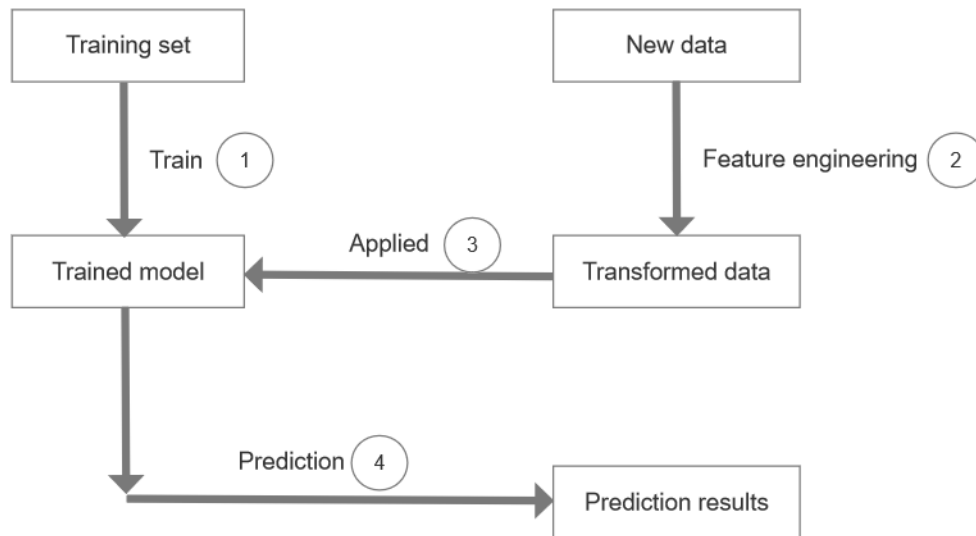


Figure 5.1: The process of model training and prediction

As it is shown in Figure 5.1, we used two metrics, one is the training time, time spending on job 1, another is the prediction time, the summation of time spending on job 2, job 3 and job 4.

Experimental setting:

- Data size: one-year dataset
- Granularity level: daily
- Current window: 3-month
- Future window: 1-month

Multilayer Perceptron The first 6-month was used as the training set containing 8,890,503 call records and 128,492 customers, and the other half was used as the validation set containing 8,188,064 call records and 99,257 customers. Both MLP and WTTE-RNN model use a 3-month Current window and 1-month Future window, as it shows in Figure 4.1, each sliding window generates labeled data, and training set is the concatenation of all labeled data.

Weibull Time to Event Recurrent Neural Network As we described in Figure 4.1, a WTTE-RNN uses the C months to train a WTTE-RNN model and uses the F months to predict churn labels for customers in C and evaluate the model performance. In order to compare with MLP model, we use MLP’s validation dataset as the experimental data for WTTE-RNN.

Model	Training Time (seconds/per customer)	Prediction Time (seconds/per customer)
MLP	0.003	0.022
WTTE-RNN	0.018	0.03

Table 5.3: Comparison of MLP and WTTE-RNN on time

We can see from Table 5.3 that the WTTE-RNN model takes much more time than the MLP model in training phase, almost six times than MLP. It is because that the WTTE-RNN requires retraining processes, each sliding window needs to train a new WTTE-RNN model as new time series data enters (Section 4.2.2), however a MLP only needs to train once because a trained MLP can be directly applied for new time series data thus reducing computational cost. The prediction time of the two models is similar with WTTE-RNN’s being slightly higher than MLP’s, as expected.

5.4 Summary

To sum up, based on the results shown in Table 5.1, both MLP and WTTE-RNN behave better when the granularity level of data is higher. And the model performance of MLP for churn prediction is better than that of the WTTE-RNN model, it can be seen from the results shown in Table 5.1, the MLP model achieves a slightly higher AUC, precision and Recall. However, the capability of MLP to model long-term dependency is less than WTTE-RNN, which can be seen from the Table 5.1, the MLP’s model performance decreases from 0.921 to 0.884 using a Current Window from 3 month current window to 5 month current window under a 3-day granularity setting. This method would be more suitable for simple time series data without long-range temporal dependencies, and all the relevant information is contained in a few recent inputs.

For the WTTE-RNN model, training and tuning is a much more time-consuming process than the MLP model. This is because that the theory and maths underlying the working of the WTTE-RNN model are more complicated and challenging to understand. Moreover, RNN requires memory-bandwidth-bound computation, which ultimately limits the applicability of neural network solution [29]. In [29], author mentions that RNN requires multiple layers per cell to run for time step for each sequence [29]. And linear layers require large amounts of memory bandwidth to be computed which requires many computational units [29]. In addition, the WTTE-RNN approach is not efficient, every time when a new customer enters the system, the WTTE-RNN model needs to be retrained for predicting this customer’s future call activity, greatly increasing the computational cost. In contrast, the MLP approach learns the underlying logic between customer call behavior and corresponding churn labels, a trained model can be directly applied to new data without frequent retraining processes, thus reducing the computational cost.

In short, the WTTE-RNN is not convenient in our case because its frequent retraining will lead to a high computational cost. Thus eventually we choose to use the MLP model as our churn prediction model.

Chapter 6

System Development

This chapter demonstrates our system development for the churn problem and presents a user interface we developed.

6.1 Architecture Overview

Figure 6.1 shows the functional architecture of the Churn Prediction System. The Churn Prediction System is composed with three layers, namely data layer, model layer and application layer. The data layer communicates with the database to migrate customer activity data from the database to the local file system and transform customer activity data into customer data for future model training and prediction. We set an automatic task to perform offline daily data migration and daily data transformation jobs, saving online data transformation time. The model layer is used for model training, validation, and prediction. The application layer is mainly used for user interaction.

6.2 System Implementation

6.2.1 Apache Spark

Spark is known as a fast engine for big data preprocessing that has built-in modules for streaming, SQL, machine learning and graph processing [30]. We used Spark in the data layer to perform the data preprocessing.

6.2.2 Tensorflow

Tensorflow is an open source framework initially developed by Google that implements many popular machine learning algorithms. We used Tensorflow in the model layer to perform the model training, validation and prediction.

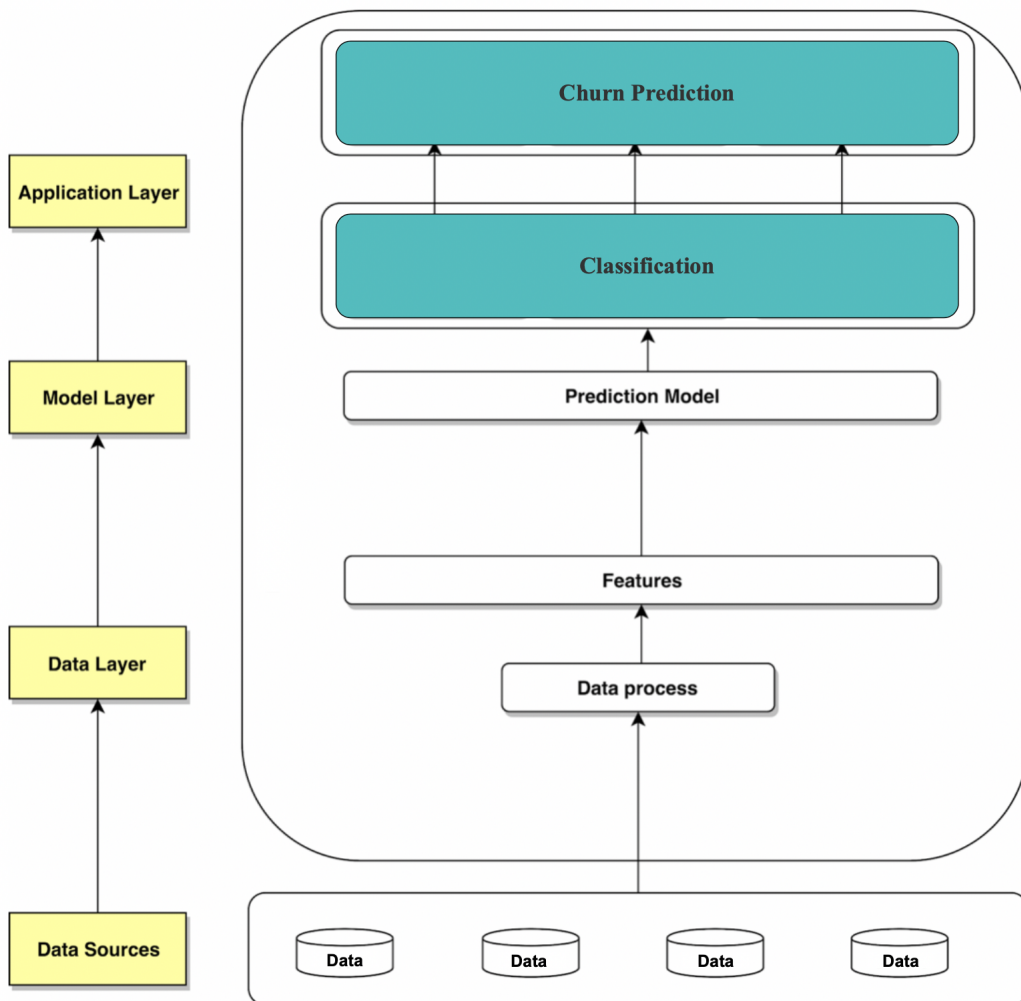


Figure 6.1: System overview

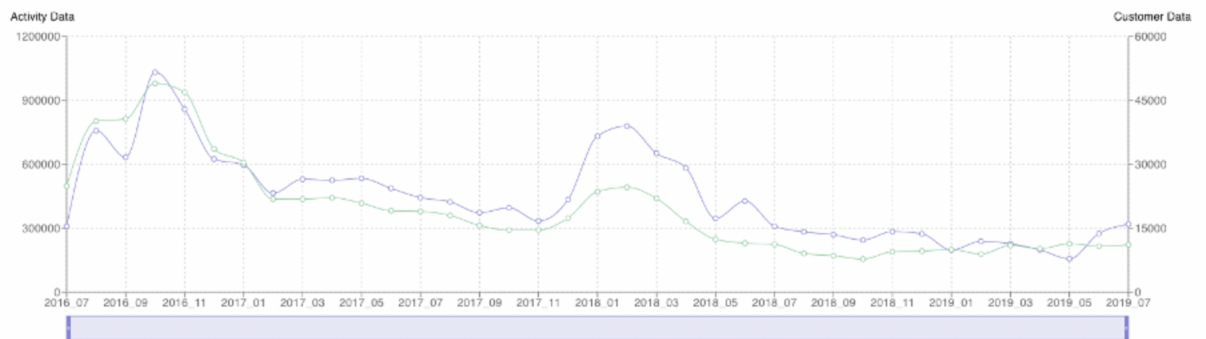
6.2.3 Flask

Flask is an API of Python that allows us to build web applications, and the framework of Flask is more explicit than other framework. We used Flask in the application layer to build a web application and deployed it to the cloud.

6.3 Churn Prediction Interface

Step 1: Select Data

Activity Data and Customer Data



Skip

Step 2: Define Churn

Back Skip

Step 3: Train Model

Back Skip

Step 4: Prediction

Back

Figure 6.2: Interface overview

The term “user” refers to the person who uses the Churn Prediction System. To help those users who do not have experience with machine learning and software development, we developed a user interface for users to use our machine learning system. The interface is compound with four steps, namely “Select Data”, “Define Churn”, “Train Model”, and

“Prediction” (Figure 6.2). A user can skip early steps by clicking the button “Skip” to move to later steps. For instance, if a user does not want to train a new model, one just wants to perform prediction tasks, he can skip “Select Data”, “Define Churn”, “Train Model” steps and move directly to the “Prediction” step.

The first step is “Select Data”, a user selects available data from a desired database for future model training and validation steps. In this step, a user first needs to select an available database, and an “Activity Data and Customer Data” graph (Figure 6.6) is used to help to select range of the activity data. In Figure 6.6, the purple curve demonstrates an overview of the number of activities in the monthly activity data, and the green curve demonstrates an overview of the number of customers in the monthly activity data. A user can put a cursor on a curve to see the exact values for different months. If a database contains enormous monthly activity data, one can also drag the horizontal scrollbars at the bottom to zoom in or out on the monthly activity data to see the value clearly.

The next step is “Define Churn”, a user needs to enter the values of activity threshold a and churn threshold b , and selects the values from the dropdown window for current window C and future window F (Figure 6.3). A graph “Boxplot distribution for customer monthly call count” (Figure 6.7) containing boxplots for each month is used to guide the user to choose appropriate values of a , b , C and F , and the user can put the cursor over the boxplot to see these statistics about a month: minimum, first quartile, second quartile, third quartile and the maximum.

Parameters to Define Active Customers: Activity Threshold a : Current Window C :

Parameters to Define Churn Customers: Churn Threshold b : Future Window F :

Figure 6.3: Define active customer and churn customer with a , b , C and F

In the step of the “Train Model”, a user needs to click button “Training”, a model will be automatically trained to learn from the selected labeled examples. An estimated time of training will be displayed on the screen. Once model training is completed, the model information including the training data summary and the model performance will be saved into the table “Recent Saved Models” (Table 6.1), and information about all recent saved models can be found in this table, ordered by the model creation time, the recent saved model is placed at the top. The model performance is evaluated based on a criterion called the top $K\%$ precision. The user can select the value K by dragging the mouse using the slider under the column “Select K ”, and then the column “Top $K\%$ Precision” will

Recent Saved Models






		Train Data Summary					Model Result On Test Data		
Model Id	Model Create Date	Activity Threshold	Churn Threshold	Date	Customer Count	Churn Percentage	Select K	Top K% Precision	
1567717407	09/05/2019, 09:03:27 PM	30	2	2017_10 - 2017-11	19000	0.3	 50	0.89	Delete
1567719048	09/05/2019, 06:02:15 PM	30	2	2018_01 - 2018-04	17692	0.28			Delete
1567717407	09/05/2019, 05:30:48 PM	30	2	2018_01 - 2018-04	17692	0.28			Delete
1567555982	09/05/2019, 05:03:27 PM	30	2	2018_01 - 2018-04	17692	0.28			Delete
1567551735	09/03/2019, 11:02:15 PM	30	2	2018_01 - 2018-04	17692	0.28			Delete

Table 6.1: Recent saved models

automatically display the corresponding precision value. The user can also delete a model from the table by clicking the button “Delete”.

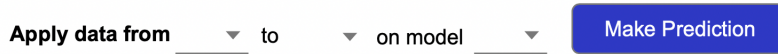


Figure 6.4: Select data on a saved model

Once a model is saved into the table “Recent saved model”, the user can apply a saved model to predict the labels of unlabeled examples. This step is the third step of the Churn Prediction Interface called “Prediction”. The user can select a sequence of month data as unlabeled examples and then choose one model from a dropdown menu and click the button “Make Prediction” (Figure 6.4). A prediction result will be automatically saved into the table “Recent Saved Predictions” (Table 6.5), the recent saved prediction record is highlighted with blue color and is placed at the top. Each row contains the information about model Id, model creation time, train date range, and prediction date range. The user can see top churners by selecting a K value to determine the number of predicted potential churners ordered by churn probability, and clicking the button “Show”, only the top $K\%$ predictions will be presented in the list. The top $K\%$ prediction list contains information about churn probability and top-5 features, contributing the most five influencing factors to determine each customer’s predicted churn score. This prediction list can be exported to a csv file by clicking the button “Export”.

Recent Saved Predictions



Model Id	Model Create Date	Train Date	Prediction Date	Select K	Top K% Prediction
1567717407	09/05/2019, 05:30:48 PM	2018_01 - 2018-04	2018_01 - 2018-04	 50	Show Export Delete
1567555982	09/05/2019, 05:03:27 PM	2018_02 - 2018-06	2018_06 - 2018-06		Show Export Delete

Figure 6.5: Recent saved predictions

Activity Data and Customer Data

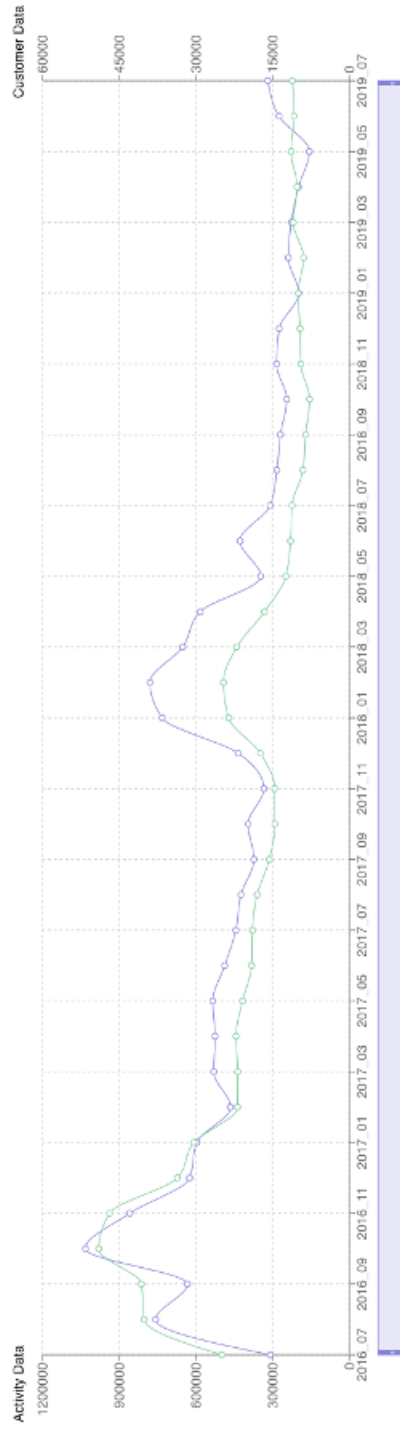


Figure 6.6: Zoom in or out on the monthly activity data

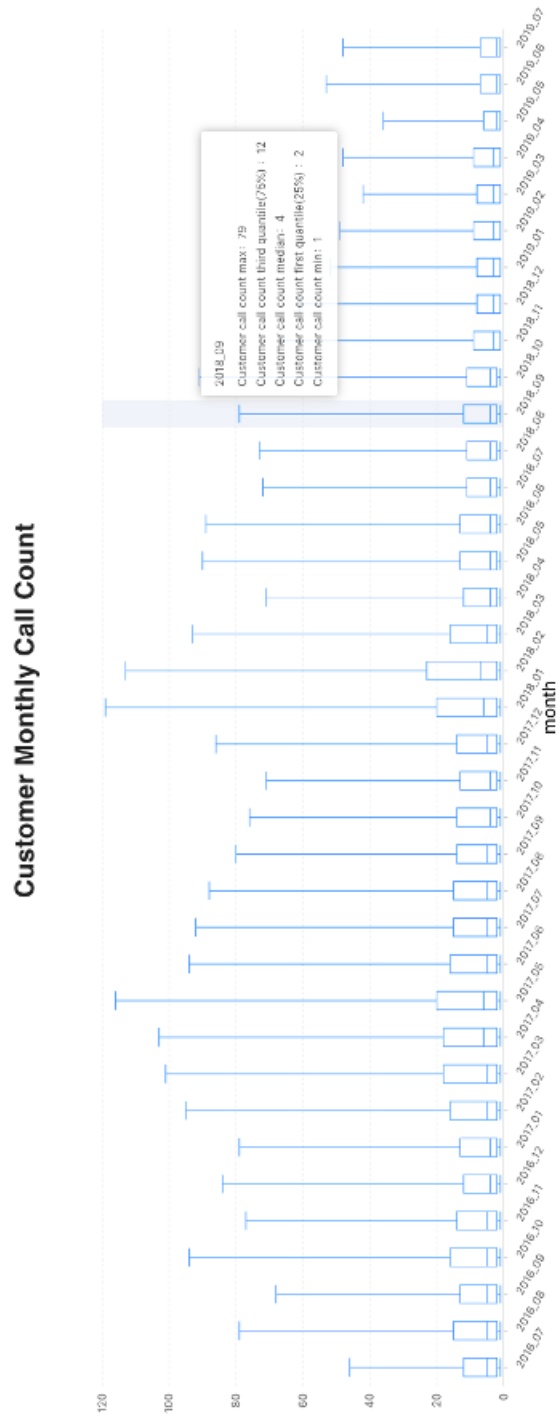


Figure 6.7: Boxplot distribution for customer monthly call count

Chapter 7

Conclusions and Future Works

7.1 Conclusions

Churn prediction plays a vital role in operating a competitive company. In our project, we aim to build a comprehensive churn prediction system that can predict potential churners for pre-paid customers with high accuracy, and can adapt to changes in a dynamic business environment, more importantly, it should have reasonable computational time, for reducing computational cost.

This project is aimed to implement and find a most suitable churn prediction model for a telecom company, the MLP and WTTE-RNN were tested and evaluated on a real-world dataset containing customer call activities in our research. In practice, various considerations should be taken into account, such as accuracy and efficiency. Regarding the model performance on the accuracy which is shown in Table 5.1, the MLP attains the highest AUC value, achieving as high as 0.92 AUC. Efficiency is another evaluation metric needs to be considered, an efficient model should avoid high computational cost, as shown in Table 5.3, the WTTE-RNN takes more time than the MLP, a trained WTTE-RNN cannot be directly applied for new time series data containing different months of data which causes computationally expensive. However, the MLP is a method that learns the hypothesis to map the input customer behavior patterns with the churn labels, new time series data can be directly applied on a trained MLP. Thus, we chose to use the MLP as our churn prediction model finally.

In addition, we presented a detailed feature engineering process for both MLP and WTTE-RNN model in our project, i.e., how to generate temporal patterns from the raw activity data. We also presented how we develop a churn prediction system for M800 Limited in this thesis, and an explicit demonstration of the user interface is demonstrated in Chapter 6.2. However, there still much more works need to be done, in the following, we will present some ideas of future works.

7.2 Future Works

Since the dimensions of the input data for the MLP model depend on the length of the input time window, with the increase of the input time window, the dimensions of the customer feature matrix become higher, which may lead to the deterioration of the model performance. In the future work, we could remove some irrelevant features using feature selection method.

We also plan to use the deep belief networks to solve churn problem since it has been shown that using unsupervised pre-training layers can improve the predictive performance.

Bibliography

- [1] Hossam Faris, Ibrahim Aljarah, and Seyedali Mirjalili. *Training feedforward neural networks using multi-verse optimizer for binary classification problems*. 2016.
- [2] Egil Martinsson. *WTTE-RNN : Weibull Time To Event Recurrent Neural Network A model for sequential prediction of time-to-event in the case of discrete or continuous censored data, recurrent events or time-varying covariates*. 2017.
- [3] Reichheld Frederick F. *Loyalty-based management*. Harvard Business Review, 1993.
- [4] Carl Yang, Xiaolin Shi, Luo Jie, and Jiawei Han. *I Know You'll Be Back*. 2018.
- [5] Emilia Huong Xuan Nguyen. *Customer Churn Prediction for the Icelandic Mobile Telephony Market*. 2011.
- [6] Jin Xiao, Xiaoyi Jiang, Changzheng He, and Geer Teng. *Churn Prediction in Customer Relationship Management via GMDH-Based Multiple Classifiers Ensemble*. 2016.
- [7] bdelrahim Kasem Ahmad, Assef Jafar, and K. Aljoumaa. *Customer Churn Prediction in Telecom Using Machine Learning in Big Data Platform*. 2019.
- [8] Wai-Ho Au, K.C.C. Chan, and Xin Yao. *A novel evolutionary data mining algorithm with applications to churn prediction*. 2003.
- [9] Adnan Idris and Asifullah Khan. *Ensemble Based Efficient Churn Prediction Model for Telecom*. 2014.
- [10] Ning Lu, Hua Lin, Jie Lu, and Guangquan Zhang. *A Customer Churn Prediction Model in Telecom Industry Using Boosting*. 2014.
- [11] A. Mishra and U. S. Reddy. *A Novel Approach for Churn Prediction Using Deep Learning*. 2017.
- [12] Amjad Hudaib, Reham Dannoun, Osama Harfoushi, Ruba Obiedat, and Hossam Faris. *Hybrid Data Mining Models for Predicting Customer Churn*. 2015.
- [13] Mohammad Ismail, Mohd Khalid Awang, Mohd Nordin Abdul Rahman, and Mokhairi Makhtar. *A Multi-Layer Perceptron Approach for Customer Churn Prediction*. 2015.
- [14] Thanasis Vafeiadis, Kostas Diamantaras, G. Sarigiannidis, and Konstantinos Chatzisavvas. *A Comparison of Machine Learning Techniques for Customer Churn Prediction*. 2015.

- [15] Omar Adwan, Hossam Faris, Khalid Jaradat, Osama Harfoushi, and Nazeeh Ghatasheh. *Predicting Customer Churn in Telecom Industry using Multilayer Preceptron Neural Networks: Modeling and Analysis*. 2014.
- [16] Anuj Sharma and Prabin Kumar Panigrahi. *A Neural Network based Approach for Predicting Customer Churn in Cellular Network Services*. 2011.
- [17] Havrylovykh Mariia and Kuznietsova Nataliia. *Survival analysis methods for churn prevention in telecommunications industry*. 2019.
- [18] Edward Choi, Taha Bahadori, and J. Sun. *Doctor AI: Predicting Clinical Events via Recurrent Neural Networks*. 11 2015.
- [19] R. Joshi and C. Reeves. *Beyond The Cox Model: Artificial Neural Networks For Survival Analysis Part II*. 2006.
- [20] Shin-Yuan Hung, David C. Yen, and Hsiu-Yu Wang. *Applying data mining to telecom churn management*. 2006.
- [21] T. Chen, Brian Keng, and J. Moreno. *Multivariate Arrival Times with Recurrent Neural Networks for Personalized Demand Forecasting*. 2018.
- [22] Jane Lu, AstraZeneca Pharmaceuticals, Gaithersburg, and David Shen. *An Application of Survival Analysis Modeling Using SAS*. 2017.
- [23] R. D. KING, C. FENG, and A. SUTHERLAND. *STATLOG: COMPARISON OF CLASSIFICATION ALGORITHMS ON LARGE REAL-WORLD PROBLEMS*. Taylor Francis, 1995.
- [24] T S Lim, Wei-Yin Loh, and Yu-Shan Shih. *A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms*. 2000.
- [25] Claudia Perlich, Foster Provost, and Jeffrey Simonoff. *Tree Induction vs. Logistic Regression: A Learning-Curve Analysis*. 2003.
- [26] Steven D. Brown, Romà Tauler i Ferré, and Beata Walczak. *Comprehensive chemometrics: chemical and biochemical data analysis*. Elsevier, 2020.
- [27] Bruce G. Marcot. *Metrics for evaluating performance and uncertainty of Bayesian network models*. 2012.
- [28] Federico Castanedo. *Using Deep Learning to Predict Customer Churn in a Mobile Telecommunication Network*. 2014.
- [29] Eugenio Culurciello. *The fall of RNN / LSTM*. Towards Data Science, 2019.
- [30] *Apache Spark Tutorial: Machine Learning*. 2017.