Deep Neural Pipeline for Churn Prediction and Mitigation

Andrei SIMION-CONSTANTINESCU[1], Nicolae ȚĂPUȘ[2], Andrei Ionuț DAMIAN[3], Bogdan DUMITRESCU[4]

***Abstract*— Customer Churn is a essential retail metric used in business predictive analytics systems to quantify the number of customers who left a company. All retail and business to consumer companies carefully analyses customer behavior to prevent them to cease their relationship with the company, in other words to make churn. With the latest development of Artificial Intelligence and the ones particularly related to Deep Learning, we have a new set of powerful tools ready to employ within multiple horizontal and vertical domain – such as the horizontal of predictive business analytics domain. One of the main goals of predictive analytics is the research and development of the almost-perfect churn detection system. This paper objective is to propose a beyond state-of-the-art churn prediction and mitigation model together based on deep neural models and employing Big Data processing on massive parallel computing using GPU cells.**

***Keywords* – machine learning, business predictive analytics, massive parallel computing, on GPU, deep learning, customer retention, big data, churn prediction**

I. INTRODUCTION

Custom churn is an event that occurs when clients or subscribers stop doing business with a company or quit using a service, being a crucial metric in evaluating clients satisfaction over a period of time. Predicting customers behavior has always been an important research area for all retail and business to consumers companies with high impact in the structure of the marketing campaigns. Identifying early who is at risk to leave is the top priority, due to the fact that it is more expensive and time-consuming to acquire a new customers than retaining old ones.

This paper presents many churn prediction models starting from different types of neural networks for computing a churn probability for each customer and finishing with a time-to-event model able to determine the number of days to the next event for each client, all of this applied on Pharmaceutical Industry taking advantage of Big Data provided from the largest Pharmaceutical Group in our country. Churn prediction techniques attempt to understand customer behaviors and attributes which signal the risk and timing of clients churn. Pharmaceutical Industry has been used as a target industry for our applied industrial research and we do have actual successful research experiment in our work.

[1]Computer Science and Engineering student at Faculty of Automatic Control and Computers University Politehnica of Bucharest, 313 Splaiul Independentei, Romania andrei.simion.c at gmail.com

[2]Full Professor at the Department of Computer Science and Engineering University Politehnica of Bucharest, 313 Splaiul Independentei, Romania ntapus at cs.pub.ro

[3]Chief Data Scientist University Politehnica of Bucharest, 313 Splaiul Independentei, Romania damian at cloudifier.net

[4]General Manager at High-Tech System & Software Bucharest, Romania bogdan.dumitrescu at htss.ro

The first step in creating our models was the exploratory analysis, in which we took the raw data for all clients, we analyzed and processed keeping only the attributes that were proved to be relevant to our mathematical models. After preprocessing the raw data, we split the customers into 4 clusters, from the *weakest* clients to the *strongest* ones using Recency, Frequency and Monetary Mass analysis (RFM) [12]. The main customers segments were used for high-level targeting. Following main segmentation, we moved further and split every macro-cluster into micro clusters allowing us to target more specific customer ranges.

Another important stage of our project was to set the definition of churn for our real life problem. For the deep learning models that generate a churn probability, a client is flagged as making churn if in the previous year it has a minimum number of one transaction and in the next one that client has not made any purchases. For the time-to-event model, we needed to refine the churn definition in terms of days from the last transaction.

The next step was choosing the right architecture for the final ensemble model based on space search odyssey. In this paper, we studied all ranges of mathematical models beside the actual final research results, starting from a simple linear model and all the way down to a complex Directed Acyclic Recurrent Graph. We presented the architecture and performance for numerous neural models such as Fully-connected neural networks, Recurrent neural networks and ensemble of different models. As described in this paper, our experimentation has been conducted using a Pharmaceutical Industry database of over 150 millions observations, thus resulting in a real base for benchmarking in conjunction with the plethora of proposed concurrently models.

Our model performance will be quantified using standard metrics, from which we are especially interested in two performance evaluating measures: precision and recall. Recall rate tell us who may churn clients our model identifies from the real ones and precision give us the false positive rate of our model. We tried to maintain a balance between a high recall rate and a decent precision one, trying to get a steady state in which we accept only a tiny decrease in recall for a better increase in precision.

II. RELATED WORK

A. Neural networks used for churn prediction

Artificial neural network were used for predicting customer churn mainly in telecommunication field [7] among other prediction techniques such as Logistic Regression [6], Decision Trees [15], Naive Bayes [16] and Support Vector Machines [5].

Multilayer Perceptron Neural Networks(MLP) or more commonly know as fully-connected neural networks were acquired for churn prediction problem in Telecom Industry [2]. The attributes used as inputs from the fully-connected neural network range from Total Consumption (monthly fee for SMS and calling), Local/International calling/SMS fees and counts, Outgoing calls local and international in minutes and so on. The architecture of the MLP can be seen in Figure 1.

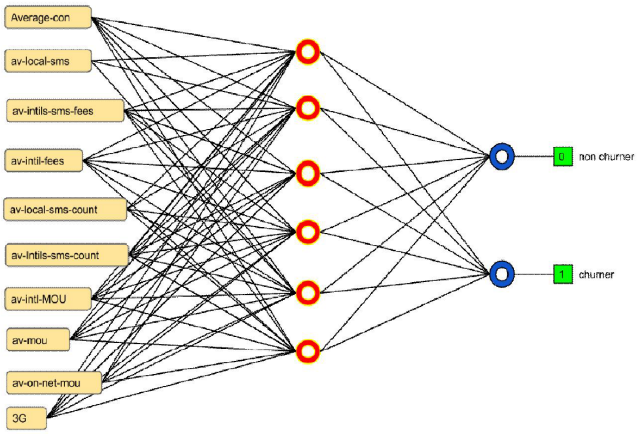

Fig. 1. Fully-connected neural network used for telecom churn prediction with 10 customer attributes [1]

The image classification capabilities of Convolutional Neural Networks [8] were used for churn analysis by representing customers as $2D$ images [17]. As can be seen in Figure 2, each image row represents a day in the life cycle of a client and each column encodes an attribute such as Data usage, SMS in or voice out.

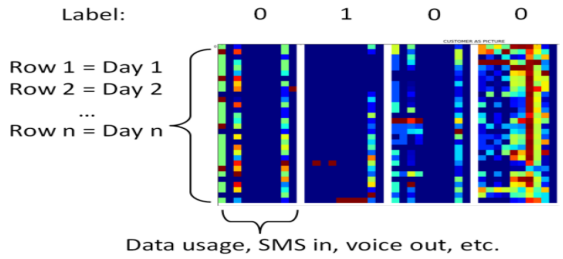

Fig. 2. Customers monthly activity encoded into a $2D$ image with days on rows and features on columns [2]

B. Recurrent networks used for time-to-event prediction

Another way of identifying clients with high churn probability is by determining the time to its next event. This approach was presented by Egil Martinson in his Master Thesis [11]. His proposed model estimates the distribution of time to the next event using a Weibull distribution [13] whose parameters being the output of a recurrent neural network [10]. The training process is made based on recorded events v_t, having the true time-to-next-event y_t and the predicted output $\overline{y}_t$ as seen in Figure 3.

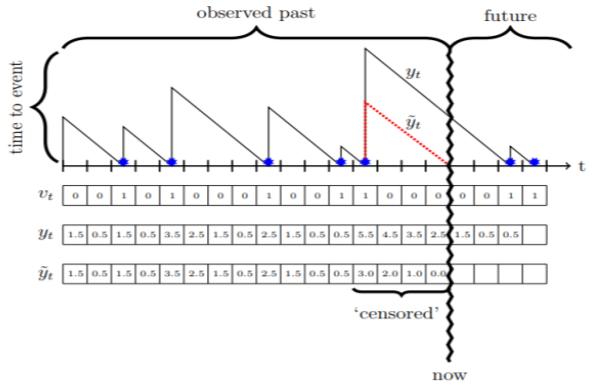

Fig. 3. Time-to-event prediction based on observed-past with recorded events and treu time-to-event from past[3]

III. OUR WORK

A. Data structure

There are two main datasets used both for training and testing purposes: **Customer attributes** and **Customer transactions**. The **Customer attributes** dataset has around 2 millions clients, each observation having 168 predictors variables. The **Customer transactions** dataset has 4 millions clients, each client having a variable number of observations equal to the number of its transactions. It should be underlined that **Customer attributes** has clients data for one year and **Customer transactions** has for multiple years.

To provide a formal description of data from **Customer attributes** dataset, each client is described by $X = \{x_i\}$, $i \in [1,42]$, 42 features such as recency (when it made the last purchase), frequency (how often it made purchases), monetary mass and revenue on different brands and category of products. Each of this 42 features is split in quarters, meaning that each observations has $4X42 = 168$ predictors variables. The target variable is the churn flag $y \in \{0,1\}$. Due to year-to-year churn definition, **Customer attributes** dataset contains active clients features over one year and churn flag determined for the next year, thus having **Customer attributes 2015-2016** and **Customer attributes 2016-2017** (A $\rightarrow$ B: attributes recorded for transactions in year A with churn flag computed based on transactions in year B).

Analyzing data from **Customer transactions** dataset, each row is described by *Customer ID*, *Transaction ID*, *Site ID*, *Transaction amount*, *Transaction 128-embeddings* and *Transaction time-stamp*. Formalizing this description, each observation is $X = \{x_i\}$, $i \in [1,133]$. This dataset is group by customers, each client having a list of their purchases, meaning that the number of rows for each client is equal with

[1]©http://www.lifesciencesite.com/lsj/life1103/011_22840life110314_75_81.pdf

[2]©https://arxiv.org/ftp/arxiv/papers/1604/1604.05377.pdf

[3]©http://publications.lib.chalmers.se/records/fulltext/253611/253611.pdf

its transactions count. All products contained in a transaction are encoded into $128-embeddings$ computed as an average of the $128-embeddings$ of each product determined using NLP-like techniques like GloVe [14], applied on product universe. The target variable is time-to-event represented in days, $y \in [0,365]$. Unlike the other dataset, the target variable in not present in the original data and needed to be computed based on time-stamp differences.

B. Training data preparation

For the churn prediction based on probability models, the input training data was fed into three different shapes depending on each type of neural network architecture: fully-connected, convolutional and recurrent.

For he fully-connected architecture, the input layer was loaded with batches of data of size $[168,]$ taking into consideration the fact that this type of neural network is unable to understand sequences, thus splitting the $[168,]$ input vectors into quarters would not be useful.

For the convolutional neural network, the input is reshaped as a $1D$ image of size $[168, 1]$. We used a kernel of size $[42, 1]$ with stride 42 to generate one activation map for each quarter. The quarterly data is then combined by another convolutional layer with kernel size $[4, 1]$ and stride 4. This type of architecture performed much weaker than the other experimented neural network, therefore we will not further analyze its architecture.

For the recurrent neural network, the input layer accepts tensors of shape $[4, 42]$ modeling the life cycle of a customer in 4 time-steps, each for one quarter. At one arbitrary time-step i, $i \in [1,4]$, we have 42 values for recency, frequency, monetary mass and revenue on brands and category of products. The 4 vectors of shape $[42,]$ are stacked one after another forming a sequence of 4 time-steps.

Time-to-event preparation of training data was more complex. Firstly, we needed to compute another two columns for each existing row from Customer transactions dataset: time from previous event - *TPE*, time to next event - *TTE*. Let's assume that for one customer we had n events (transactions). For this model to work, we enforced $n \geq 2$. TPE and TTE columns are computed as follows (eq. 1) with *df_customer* being a dataset with all transactions for a particular client.

$$\begin{aligned} TPE &= 0 + df\_customer[1:n-1] \\ TTE &= df\_customer[2:n] \end{aligned} \tag{1}$$

The training data for time-to-event model is represented as a tuple (X, y) where $X = [x_1, x_2, x_3]$ and $y = TTE$. The 3 inputs for this model are $Input\_user = x_1$ with shape $[1, n]$, $Input\_site = x_2$ with shape $[1, n]$ and $Input\_emb\_val\_tpe = x_3$ with shape $[1, n, 130]$. The $Input\_user$ and $Input\_site$ have the customer ID, respectively the site ID replicated for each transaction. The $Input\_emb\_val\_tpe$ combines the $128 - embeddings$ of transaction computed as the average between the product embeddings from that particular transaction with the amount of the transaction and TPE.

C. Models architecture

In order to simplify our research and experimentation process, we used Keras [3] high-level Deep-Learning framework on top of TensorFlow [1].

For churn prediction deep neural networks we will present the fully-connected neural network, the recurrent neural network formed with stacked Long-short term memory (LSTM) and the ensemble model which combined the advantages of the two types of neural networks architectures. In the end of this section, we will analyze time-to-event model architecture consisting of trainable embeddings layers and LSTM units.

The fully-connected neural network has 2 hidden layers, first with the number of units equal with the input size and second with half of the input size. To avoid over-fitting, we used dropout technique, deactivating 50% of neurons between first hidden layer and last hidden layer. The Keras architecture can be seen in Figure 4.

Layer (type)	Output Shape	Param #
Input (InputLayer)	(None, 168)	0
Hidden_1 (Dense)	(None, 168)	28392
Dropout_half (Dropout)	(None, 168)	0
Hidden_2 (Dense)	(None, 84)	14196
Output (Dense)	(None, 1)	85

Total params: 42,673
Trainable params: 42,673
Non-trainable params: 0

Fig. 4. Keras summary of fully-connected neural network used for churn prediction

The final recurrent neural network designed consisted of 3 stacked LSTM layers with skip-connections to avoid signal loss, each with 256 units followed by the fully-connected network presented above. Instead of computing the churn probability with only one fully-connected layer at the end, we used the final part of the best fully-connected architecture that was fed with aspects learned from data sequentiality by LSTM units. The Keras architecture can be seen in Figure 5.

Layer (type)	Output Shape	Param #
Input (InputLayer)	(None, 4, 42)	0
LSTM_1 (CuDNNLSTM)	(None, 4, 256)	307200
Concat_1 (Concatenate)	(None, 4, 298)	0
LSTM_2 (CuDNNLSTM)	(None, 4, 256)	569344
Concat_2 (Concatenate)	(None, 4, 554)	0
LSTM_3 (CuDNNLSTM)	(None, 256)	831488
Hidden_1 (Dense)	(None, 84)	21588
Output (Dense)	(None, 1)	85

Total params: 1,729,705
Trainable params: 1,729,705
Non-trainable params: 0

Fig. 5. Keras summary of recurrent neural network used for churn prediction

Time-to-event architecture was the most complex one, due to the fact that it incorporate both training of customer and site embeddings and time-to-next-event regression. We wanted to train customers embeddings based on their transaction, as well as to train site embeddings based on types of transaction for each pharmacy. Customers were represented as $16 - embeddings$ and sites as $8 - embeddings$. After obtaining user and site embeddings, we concatenated them with the $128 - embeddings$ of a transaction, the amount of the transaction and TPE, meaning that at each time-step we had 154 features that entered into the LSTM layer with 128 units. At the end of the pipeline, we had a fully-connected layer that computes time-to-next-event in days.The Keras architecture can be seen in Figure 6.

```
Layer (type)                 Output Shape              Param #
=================================================================
Input_user (InputLayer)      (None, None)              0
_________________________________________________________________
Input_site (InputLayer)      (None, None)              0
_________________________________________________________________
Input_emb_val_tpe (InputLayer) (None, None, 130)       0
_________________________________________________________________
User_emb (Embedding)         (None, None, 16)          533328
_________________________________________________________________
Site_emb (Embedding)         (None, None, 8)           5440
_________________________________________________________________
Concat_layer (Concatenate)   (None, None, 154)         0


_________________________________________________________________
LSTM (CuDNNLSTM)             (None, None, 128)         145408
_________________________________________________________________
Output (Dense)               (None, None, 1)           129
=================================================================
Total params: 684,305
Trainable params: 684,305
Non-trainable params: 0
_________________________________________________________________
```

Fig. 6. Keras summary for time-to-event architecture

The final production-grade churn prediction system is represented by an ensemble of models, combining the high recall of fully-connected architecture with a higher precision of the recurrent architecture. After an exhaustive grid-search, we determined the best two fully-connected network and the best two recurrent networks, both types being trained on 2015-2016 Customer attributes dataset and 2016-2017 Customer attributes dataset. The probabilities generated by the two fully-connected models are combined with equal weight for each year-to-year. Same combining technique is used for the two recurrent models. After that, the final churn probability in computed by the weighted sum of fully-connected final result and recurrent final result, with a higher weight for fully-connected architecture because a high recall is our primary interest.

IV. RESUTS

A. Churn prediction based on probability

When evaluating the churn prediction model, we have to take into consideration the skewness of data from Customers attributes dataset: 27% churn flagged clients and 63% not churn customers. For classification task, like the one we have to deal with, accuracy is the most common evaluation metric. If we use a naive model which predicts all clients into the dominant class, we will get a false high accuracy rate. For balance distributions, accuracy is a good metric, but for positive/negative skewed data recall and precision are the recommended metrics. Recall rate tell us how many churn clients our model identifies from the real churn customers and precision rate give us an estimation of how many tries are necessary to identify a churn client. To monitor both recall and precision, but with more emphasis on recall we also used f2-score.

The best fully-connected models, recurrent models and the final ensemble model are presented in Table I.

TABLE I
BEST MODELS FOR CHURN PREDICTION

Model ID	Model description	Train data
FC1	FC_168-84	2015-2016
FC2	FC_168-84	2016-2017
LSTM1	LSTM_3-256-skip-FC_84	2015-2016
LSTM2	LSTM_3-256-skip-FC_84	2016-2017
ENS	Ensemble_FC-LSTM	2015-16-17

The results in terms of recall, precision and f2 for all models from Table I are summarized in Table II. To capture the power of models to generalize we tested them over a year-to-year interval different from the training period.

TABLE II
BEST NEURAL MODELS FOR CHURN PREDICTION RESULTS

Model ID	Test data	Recall	Precision	F2
FC1	2016-2017	93.02%	41.15%	74.29%
FC2	2015-2016	92.21%	42.04%	74.44%
LSTM1	2016-2017	83.04%	51.64%	74.04%
LSTM2	2015-2016	83.95%	50.08%	73.95%
ENS	2015-2016	89.57%	46.29%	75.46%
ENS	2016-2017	90.71%	45.16%	75.48%

Bellow can be found the confusion matrix for the final production-grade ensemble model (*ENS*) tested on 2016-2017 data (Figure 7).

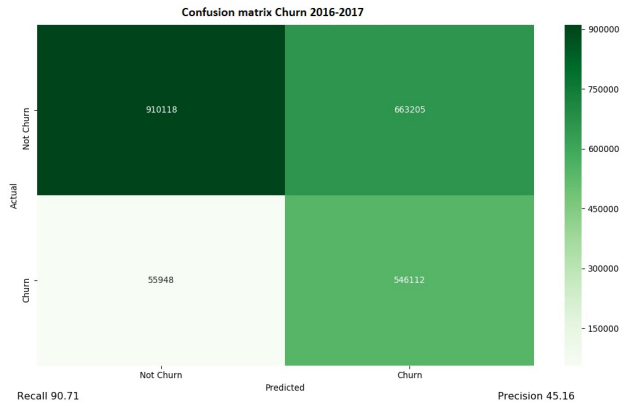

Fig. 7. Confusion matrix generated by testing of the ensemble model on Customer attributes 2016-2017 dataset

To compute the binary churn flag, we needed to convert the churn probability based on a threshold. To determine the best model in terms of recall and precision, we performed a grid-search for the threshold setting using ROC method [4] to better visualize the optimal value of the threshold.

B. Time to next event results

Most of clients from Customer transactions dataset have a recorded number of events between 100 and 500. The transaction distribution for customers with a number of events in the range of [100,500] for a small part of Customer transactions dataset can be seen in Figure 8.

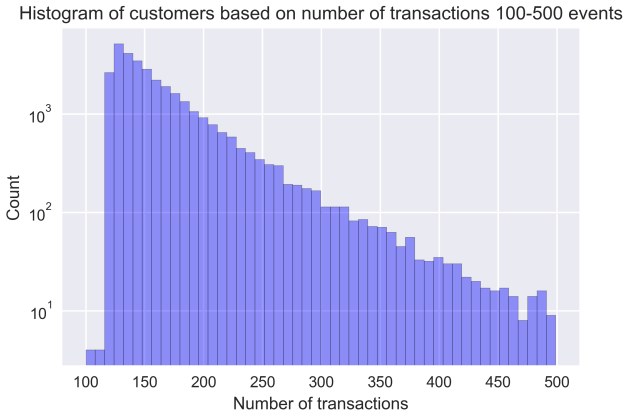

Fig. 8. Customer distribution based on their transactions for clients with number of events between 100 and 500

In order to test the time-to-event performance we randomly chose 30% of the clients from Customer transaction dataset and based on them we built a validation dataset as follows: during training for each of the validation customers we hide the final transaction, excluding it from training allowing us to use it as test data.

We randomly chose ten customers from validation dataset and run a prediction with the trained time-to-event model, recording the user ID, the real time-to-next-event, the predicted time-to-next-event as well as the number of events for that particular client. The aggregate results can be seen in Table III.

TABLE III
TIME-TO-EVENT PREDICTION RESULTS ON VALIDATION DATASET (30% OF CUSTOMERS WITH LAST TRANSACTION HIDDEN)

Client ID	Real TTE	Predicted TTE	Number of events
121	80	82.18	122
1803	6	7.77	129
9630	12	8.063	121
14765	9	20.87	139
15841	4	9.99	359
18811	12	11.51	121
21774	0	3.54	255
22548	60	59.68	147
27100	25	17.99	184
32166	50	35.61	154

Using the trained Customer Embeddings, we plotted a Customer Map based on their transactional behavior using a reducing dimensionality techniques like t-SNE [9] that allows us to represent in 2D 16-dimensional embeddings. At Appendix V-A the Customer Map for all clients with number of transaction between 200 and 300 can be found. The clients distribution for the customers plotted in Customer Map can be seen in Figure 9.

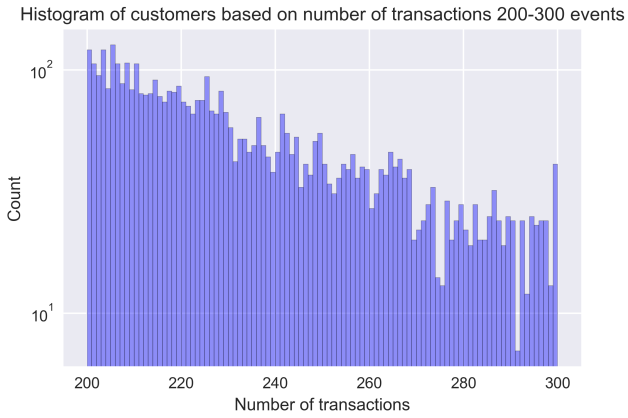

Fig. 9. Customer distribution based on their transactions for clients with number of events between 200 and 300

V. CONCLUSIONS

The two presented approaches for customer churn prediction achieve excellent result for the problem at hand. The churn prediction based on probability model achieve better than current state-of-the-art results with a recall rate of over 90% and almost 50% precision. This results was obtained by an exhaustive grid-search for the determination of model layers and parameters, for the threshold setting and for the weights of individually generated probabilities by the each model used in the final production-grade ensemble model.

The time-to-event model provide a good approximation for the days until a customer will make the next transaction, very useful information that could be used in marketing campaigns with purpose of reducing the time-to-next-event of particular targeted customers.

Further work

It remains to explore the other benefits in churn prediction analyses that the time-to-event model brings, as well as exploring in more detail the customer segmentation generated by time-to-event trained embeddings. We aim to improve the time-to-event architecture using a more advanced model based on encoder(encode past transaction)-decoder(decode future transactions) structure.

ACKNOWLEDGMENT

I would like to express my gratitude and acknowledge that production grade results was supported by the Data Science Team from 4E Software and the Business Intelligence Team from High-Tech Systems & Software.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] 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. *Life Science Journal*, 11(3):75–81, 2014.

[3] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[4] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[5] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[6] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[7] Bingquan Huang, Mohand Tahar Kechadi, and Brian Buckley. Customer churn prediction in telecommunications. *Expert Systems with Applications*, 39(1):1414–1425, 2012.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[10] Danilo P Mandic and Jonathon Chambers. *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. John Wiley & Sons, Inc., 2001.

[11] EGIL Martinsson. *Wtte-rnn: Weibull time to event recurrent neural network*. PhD thesis, Master's thesis, University of Gothenburg, Sweden, 2016.

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

[13] Toshio Nakagawa and Shunji Osaki. The discrete weibull distribution. *IEEE Transactions on Reliability*, 24(5):300–301, 1975.

[14] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[15] J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[16] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York, 2001.

[17] Artit Wangperawong, Cyrille Brun, Olav Laudy, and Rujikorn Pavasuthipaisit. Churn analysis using deep convolutional neural networks and autoencoders. *arXiv preprint arXiv:1604.05377*, 2016.

APPENDIX

A. *Customer Map for clients with 200-300 transactions*