Deep Neural Pipeline for Churn Prediction

Andrei SIMION-CONSTANTINESCU
Knowledge Investment Group
Bucharest, Romania
andrei.simion.c@gmail.com

Andrei Ionuț DAMIAN
Knowledge Investment Group
Bucharest, Romania
damian@kig.ro

Nicolae ȚĂPUȘ
University Politehnica of Bucharest
Bucharest, Romania
ntapus@cs.pub.ro

Laurențiu-Gheorghe PICIU
Knowledge Investment Group
Bucharest, Romania
laurentiu.piciu@kig.ro

Alexandru PURDILĂ
Knowledge Investment Group
Bucharest, Romania
alex.purdila@kig.ro

Bogdan DUMITRESCU
High-Tech Systems & Software
Bucharest, Romania
bogdan.dumitrescu@htss.ro

***Abstract*—Customer churn is an 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 analyze customer behavior to prevent them to cease their relationship with the company, in other words to make churn. With the latest advancements of Artificial Intelligence and 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 state-of-the-art churn prediction model based on deep neural models, time-to-next-event 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

Customer 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 components in all industrial verticals, having a high impact in the approach, strategy, structure and content of the marketing campaigns. Identifying early who is at risk to leave is top priority considering that it is more expensive and time-consuming to acquire a new customers than retaining the existing ones.

This paper presents multiple churn prediction model architectures 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 (such as a purchase event) for each individual client. For our experiments we applied the architectures and evaluated the experiments within the retail area of pharmaceutical industry taking advantage of the Big Data streams provided from some of the largest pharma retail companies in the region. Churn prediction techniques attempt to understand customer behaviors and in particular the intrinsic attributes which signal the risk and actual timing of the customer churn. Pharmaceutical industry has been used as a target industry for our applied research.

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 it by keeping only the attributes that were considered to be relevant to our mathematical models (attributes that encode behavioral patterns). Therefore, we removedirrelevant customer attributes such as sex, age, email adresses and other personal information that are filled when a customer registers with a fidelity card. 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 engineered features and resulting cluster analysis (RFM) [1]. The process that revealed the optimal number of clusters was a grid-search on KMeans algorithm whose results were analyzed using elbow method. 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. The 4 main clients segments were used for choosing what type of clients the commercial campaigns will target, after the churn model identify the clients with the highest probability to make churn.

Another important stage of our project was to set the definition of churn for our real life problem. In our basic scenario, for the deep learning models that generate a churn probability, it requires for a supervised optimization procedure and thus labeled data. A client is flagged as making churn if he did not make any transaction in the past 12 months.

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 present the architecture and the resulting optimization and inference performance for numerous neural models such as fully-connected neural networks, recurrent neural networks and ensemble of different models. As already mentioned, our experiments have 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 concurent 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 is calculated as the number of clients our model identifies as the ones which may churn divided by the real number of clients that actually churned. Precision rate is calculated as the number of real churned clients which our models identified devided by the total number of real churned clients. We tried to maintain a balance between a high recall rate and a decent precision rate, 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 [2] among other prediction techniques such as Logistic Regression [3], Decision Trees [4], Naive Bayes [5] and Support Vector Machines [6].

Multilayer Perceptron Neural Networks (MLP) or more commonly known as fully-connected neural networks were acquired for churn prediction problem in Telecom Industry [7]. 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 Fig. 1.

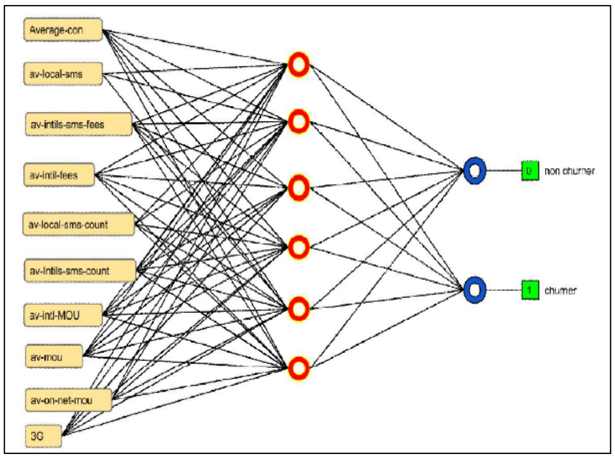


Fig. 1 Fully-connected neural network used for telecom churn prediction with 10 customer attributes [1]

The image classification capabilities of Convolutional Neural Networks [8] were used for churn analysis by representing customers as 2D images [9]. As can be seen in Fig. 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.

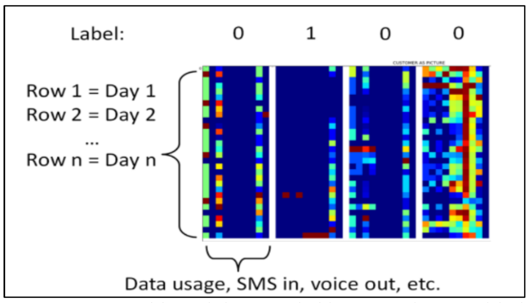


Fig. 2 Customers monthly activity encoded into a 2D image with days on rows and features on columns [2]

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 [10]. His proposed model estimates the distribution of time to the next event using a Weibull distribution [11] whose parameters being the output of a recurrent neural network [12]. 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 Fig. 3.

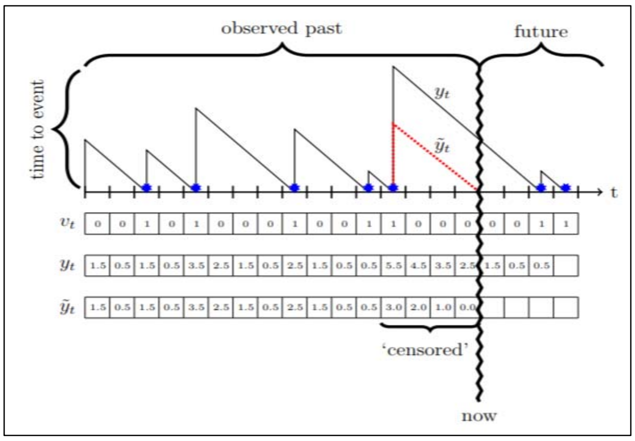


Fig. 3 Time-to-event prediction based on observed-past with recorded events and true time-to-event from past[3]

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 million clients**, each observation having *168* predictors variables. The customer transactions dataset has approximately **4 million 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. The massive number of observations were enough to train all proposed models, even if they have a large number of trainable parameters, as will be described in the following subsections.

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 (chosen after preprocessing the raw clients data and eliminating the irrelevant attributes such as the customers personal information provided when signing for a fidelity card) is split in quarters, meaning that each observations has $4 X 42 = 168$ 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 ($A \rightarrow B$: attributes recorded for transactions in year A with churn flag computed based on transactions in year B).

[1] http://www.lifesciencesite.com/lsj/life1103

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

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

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 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 Neural Language Model approaches such as GloVe [13], applied on product latent space. Although the computing of the embeddings averages for each transaction could lead to loosing individual information of each product including the actual structure of the transaction market basked we found that this method provided solid production grade results.

The target variable is time-to-event represented in days, $y \in [0, 365]$ where *365* is arbitrary defined as maximum time-to-event. 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 the fully-connected architecture, the input layer was loaded with batches of data characterized by 168 attributes.

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. The sequence of 4 timesteps has been chosen based on the fact that the non-sequence processing models have used the same input data. For the more advanced Time-To-Next-Purchase model we have used a variable length time-series. The actual testing of the system has been done in production environment by generating future predictions and then collecting actual real values from the transactional databases. The results speak for themselves.

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, based on transactions' timestamps: time from previous event - *TPE*, time to next event - *TTE*. In order to determine a historical transactional pattern for each customer, we skipped all individuals that had less than 2 transactions.

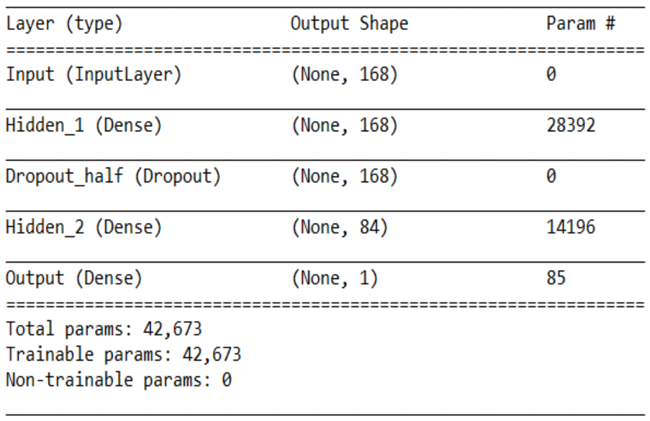
The training data for time-to-event model is represented as a tuple (X, y) where $X = [x_1, x_2, x_3]$ and $y = TTE$. The 3 inputs for this model are $Input\_user = x_1$ with shape $[1, n]$, $Input\_site = x_2$ with shape $[1, n]$ and $Input\_emb\_val\_tpe = x_3$ with shape $[1, n, 130]$. The $Input\_user$ and $Input\_site$ have the customer ID, respectively the site ID – the place (pharmacy) where the transaction was registered. 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. As we already mentioned, even if this approach could lead to loosing individual information of products, it provided solid production grade results.

C. *Models architecture*

In order to simplify our research and experimentation process, we used Keras [14] high-level Deep-Learning framework on top of TensorFlow [15].

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 embedding's 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 model architecture can be seen in Fig. 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 Model summary of fully-connected neural network used for churn prediction

GPU-optimized RNNs

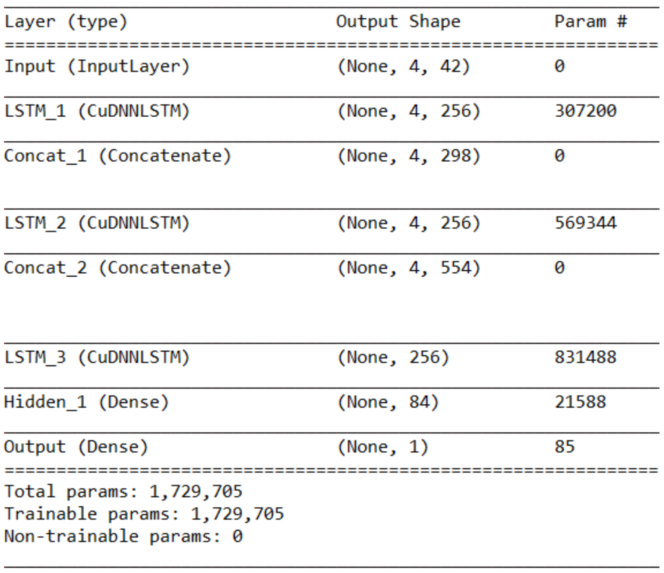
One of the recent state-of-the-art approaches that have been benchmarked in our experiments and finally adopted in our work has been that of employing advanced GPU-optimized LSTM cells. Recent research and experimentation, particularly done by NVidia corporation but also from various academic sources [16] [17], show that low-level GPU based kernels can be developed in order to optimize both the forward-propagation and back-prop within LSTM and the simpler GRU cells. As a result, we are using CuDNNLSTM cells that are optimized directly for CUDA GPU technology with dramatic improvement both in terms of training and inference time.

The final architecture

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 sequentially by LSTM units. The model architecture is represented in Fig. 5.

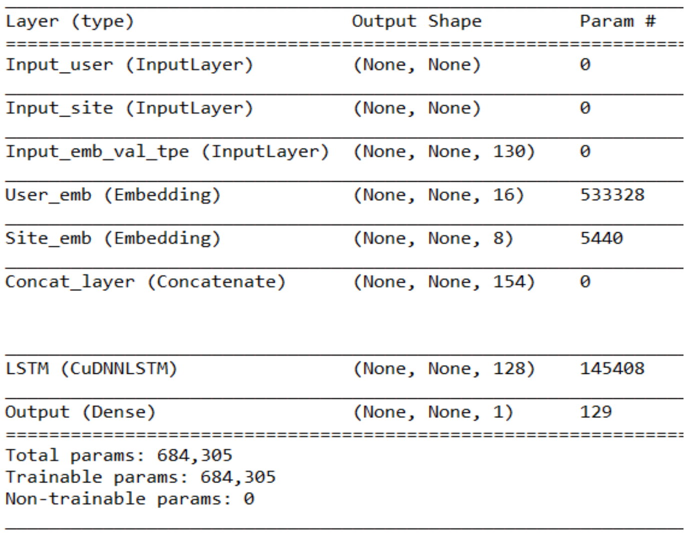
The final production-grade churn prediction system is represented by an ensemble of models, combining the high recall of fully-connected architecture with a higher precision of the recurrent architecture. After an exhaustive grid-search, we determined the best fully-connected network and the best recurrent networks. After that, the final churn probability is 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.

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 desired to train customers embeddings based on their transaction behavior, as well as to train site embeddings based on types of transaction for each pharmacy. Customers were represented as 16-dimensional embeddings and sites as 8-dimensional embeddings. After obtaining user and site embeddings, we concatenated them with the 128-dimensional averaged embeddings of the products in 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 readout layer composed of a single neuron that computes time-to-next-event in days. The model architecture can be seen in Fig. 6.



```
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 Model summary of recurrent neural network used for churn prediction



```
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 Model summary for time-to-event architecture

IV. RESULTS

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	2015-26-17

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%
ENS1	2015-2016	89.57%	46.29%	75.46%
ENS2	2016-2017	90.71%	45.16%	75.48%

In Fig. 7 can be found the confusion matrix for the final production-grade ensemble model *ENS* tested on 2016-2017 data.

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 [18] to better visualize the optimal value of the threshold.

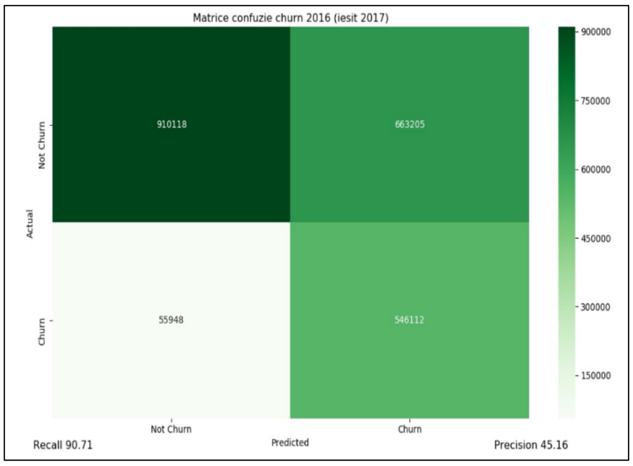


Fig. 7 Confusion matrix generated by testing of the ensemble model on customer attributes 2016-2017 dataset

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 Fig. 8.

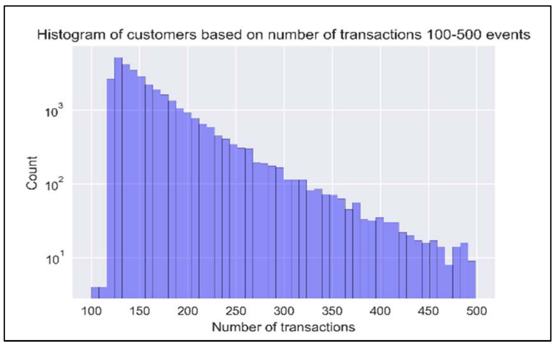


Fig. 8 Customer distribution based on their transactions for clients with number of events between 100 and 500

The training dataset of the time-to-next-event contains all recorded transactions until the last day of the year 2017. In order to test the model performance, we acquire the date of the first transaction of each client that took place in 2018 and compare it to the predicted date generated by the time-to-next-event model. **For 25% of clients, the model obtained an r2-score greater than 30% and was able to predict the date when they will make a pharmaceutical transaction with an accuracy of +/- 10 days**.

We will present a part of our results summarized in the next table by randomly selecting 20 clients and displaying the following data for each of them: customer id, the last recorded event from 2017, the date of the first 2018 transaction, the real time to next event in days calculated based on the date of the first 2018 transaction, the predicted time to next event, the difference in days between the model prediction and the ground truth and the r2-score of the whole sequence. A snapshot of the results can be seen in Table III.

Table III Time-to-event prediction results

ID	LAST DATE	REAL NEXT DATE	REAL DAYS	PRED DAYS	PRED DATE	DIFF	R2
5649	22-Dec-17	27-Apr-18	126	125	26-Apr-18	1	0.6054
5181	12-Nov-17	16-Mar-18	124	117	9-Mar-18	7	0.4804
6201	26-Oct-17	19-Jan-18	85	78	12-Jan-18	7	0.8136
3487	17-Nov-17	6-Feb-18	81	85	10-Feb-18	4	0.7386
7413	9-Dec-17	19-Feb-18	72	70	17-Feb-18	2	0.3247
4441	11-Nov-17	15-Jan-18	65	62	12-Jan-18	3	0.6107
3390	3-Dec-17	5-Feb-18	64	67	8-Feb-18	3	0.5223
3893	31-Dec-17	3-Mar-18	62	63	4-Mar-18	1	0.8236
2515	29-Nov-17	5-Jan-18	37	37	5-Jan-18	0	0.7992
2606	30-Dec-17	25-Feb-18	57	55	23-Feb-18	2	0.8035
4563	26-Dec-17	18-Feb-18	54	63	27-Feb-18	9	0.3165
6802	27-Nov-17	19-Jan-18	53	46	12-Jan-18	7	0.7993
4915	22-Dec-17	13-Feb-18	53	46	6-Feb-18	7	0.5347
5900	21-Dec-17	11-Feb-18	52	46	5-Feb-18	6	0.568
7307	18-Dec-17	25-Jan-18	38	37	24-Jan-18	1	0.9405
4736	22-Nov-17	12-Jan-18	51	50	11-Jan-18	1	0.7957
7862	30-Dec-17	18-Feb-18	50	55	23-Feb-18	5	0.7066
4457	23-Dec-17	9-Feb-18	48	40	1-Feb-18	8	0.4477
4087	20-Nov-17	7-Jan-18	48	43	2-Jan-18	5	0.6791
5649	22-Dec-17	27-Apr-18	126	125	26-Apr-18	1	0.6054

Using the trained Customer Embeddings, we plotted a Customer Map based on their transactional behavior using a reducing dimensionality techniques like t-SNE [19] that allows us to represent in 2D 16-dimensional embeddings. At Appendix 1, the Customer Map for all clients with number of transaction between 200 and 300 transactions with 7 K-Means clusters can be found. The clients distribution for the customers plotted in Customer Map can be seen in Fig. 9.

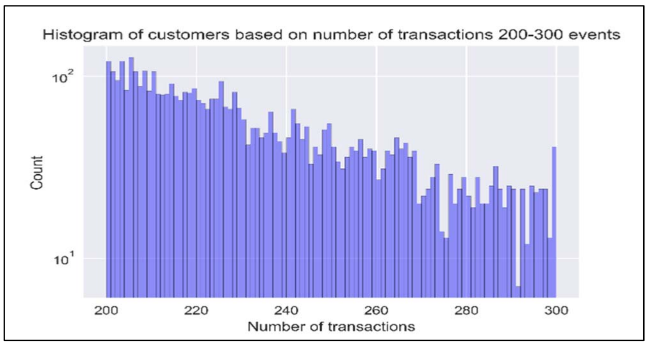


Fig. 9 Customer distribution based on their transactions for clients with number of events between 200 and 300

V. CONCLUSIONS

The two presented approaches for customer churn prediction achieved excellent results for the problem at hand. Usual business intelligence system that generate potential churn prediction together with shallow machine learning approaches available in some commercial products have the following issue: inability to capture the whole potential churn target due to a recall in the range of 50%-70% and the high marketing investment costs of targeting the churn potential target due to a low precision of the targeting algorithm. Our models have achieved a very high recall score of over 85% while the precision led to the conclusion that the target must be roughly double of the actual churn target (with a precision of around 50%). Based on our real-life experimentation and implementation these two indicators proved more than feasible for actual production grade business applications. These results were 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 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.

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.

VI. REFERENCES

[1] J. A. McCarty and M. Hastak, "Segmentation approaches in data-mining: A comparison of RFM, CHAID, and logistic regression," *Journal of business research,* vol. 60, no. 6, pp. 656-662, 2007.

[2] B. Huang, M. T. Kechadi and B. Buckley, "Customer churn prediction in telecommunications," *Expert Systems with Applications,* vol. 39, no. 1, pp. 1414-1425, 2012.

[3] D. W. Hosmer Jr, S. Lemeshow and R. X. Sturdivant, Applied logistic regression, vol. 398, John Wiley & Sons, 2013.

[4] J. R. Quinlan, "Simplifying decision trees," *International journal of man-machine studies,* vol. 27, no. 3, pp. 221-234, 1987.

[5] I. Rish and others, "An empirical study of the naive Bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, 2001.

[6] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications,* vol. 13, no. 4, pp. 18-28, 1998.

[7] O. Adwan, H. Faris, K. Jaradat, O. Harfoushi and N. Ghatasheh, "Predicting customer churn in telecom industry using multilayer preceptron neural networks: Modeling and analysis," *Life Science Journal,* vol. 11, no. 3, pp. 75-81, 2014.

[8] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.

[9] A. Wangperawong, C. Brun, O. Laudy and R. Pavasuthipaisit, "Churn analysis using deep convolutional neural networks and autoencoders," *arXiv preprint arXiv:1604.05377,* 2016.

[10] E. Martinsson, "Wtte-rnn: Weibull time to event recurrent neural network," 2016.

[11] T. Nakagawa and S. Osaki, "The discrete Weibull distribution," *IEEE Transactions on Reliability,* vol. 24, no. 5, pp. 300-301, 1975.

[12] D. P. Mandic and J. Chambers, Recurrent neural networks for prediction: learning algorithms, architectures and stability, John Wiley & Sons, Inc., 2001.

[13] J. Pennington, R. Socher and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014.

[14] A. Gulli and S. Pal, Deep Learning with Keras, Packt Publishing Ltd, 2017.

[15] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard and others, "TensorFlow: A System for Large-Scale Machine Learning.," in *OSDI*, 2016.

[16] T. Lei and Y. Zhang, "Training rnns as fast as cnns," *arXiv preprint arXiv:1709.02755,* 2017.

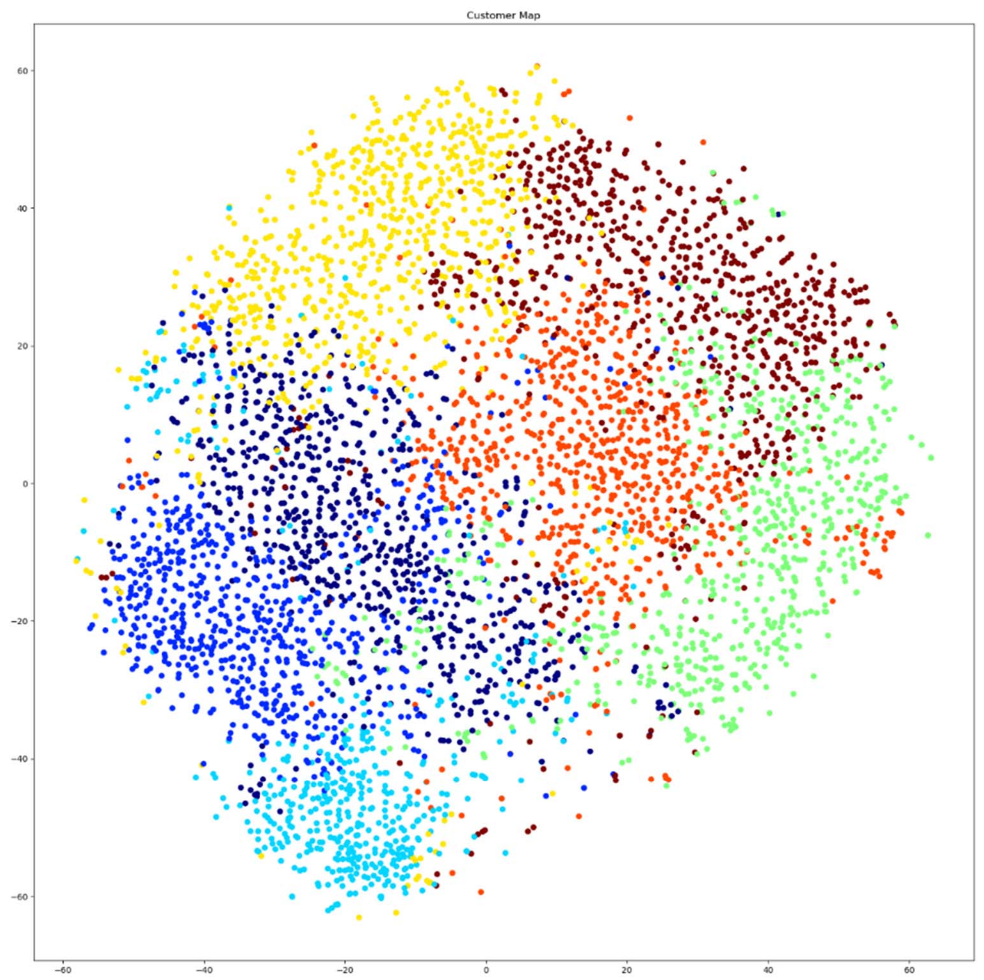
[17] M. Müller, "Optimizing recurrent neural network language model GPU training".

[18] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve.," *Radiology,* vol. 143, no. 1, pp. 29-36, 1982.

[19] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research,* vol. 9, no. Nov, pp. 2579-2605, 2008.

APPENDIX

1. Customer Map for clients with 200-300 transactions


Customer Map