Expert Systems with Applications 162 (2020) 113779

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Dynamic behavior based churn prediction in mobile telecom

Nadia Alboukaey *, Ammar Joukhadar, Nada Ghneim

Department of Artificial Intelligence, Faculty of Information Technology Engineering, Damascus University, Damascus, Syrian Arab Republic

ARTICLE INFO

Article history:
Received 25 January 2020
Revised 24 June 2020
Accepted 17 July 2020
Available online 23 July 2020

Keywords:
Churn prediction
Mobile telecom
Machine learning
Deep learning
RFM
Dynamic behavior

ABSTRACT

Customer churn is one of the most challenging problems that affects revenue and customer base in mobile telecom operators. The success of retention campaigns depends not only on the accuracy of predicting potential churners, but with equal importance, it depends on the timing when the prediction is done. Previous works related to churn prediction presented models to predict churn monthly with a focus on the static behavior of customers, and even the studies that considered the dynamic behavior of the customer, looked mainly at the monthly level behavior. However, customer behavior is susceptible to changes over days of month, and during the time leading up to a customer decision to churn, he/she starts behaving differently. Therefore, considering monthly behavioral features negatively affects the predictive performance, because it ignores changes in behavior over days of month. Moreover, predicting churners on monthly basis will be late for customers who decided to leave at the beginning of the month because they will not be detected as churners until the next month. To address these issues, in this paper, we propose daily churn prediction instead of monthly based on the daily dynamic behavior of customer instead of his monthly one. More precisely, we represent customer's daily behavior as multivariate time series and propose four models to predict churn daily based on this representation. Two models depend on features extracted from the multivariate time series, namely RFM-based model and statistics-based model. While the other models, exploit deep learning techniques for automatic feature extraction, namely LSTM-based model and CNN-based model. The predictive performance of the proposed models were investigated by evaluating them using a 150-day-long dataset collected from MTN operator in the country. The results showed that the daily models significantly outperform the monthly models in terms of predicting churners earlier and more accurately. Furthermore, the LSTM-based model significantly outperforms the CNN-based model. However, the prediction performances of the LSTM-based and the CNN-based models are equal to the prediction performance of the RFM-based model. Moreover, all of these three models significantly outperform the Statistics-based model.

1. Introduction

The mobile telecommunication industry is becoming increasingly more saturated, with more and more customers swapping their registered services between competing companies. Thus, companies have realized that they should focus their marketing efforts on customer retention rather than customer acquisition. In fact, it is less profitable for providers to attract new customers than to prevent the current customers from quitting (Hadden, Tiwari, Roy, & Ruta, 2007). Therefore, providers are engaged more and more in building predictive models in order to identify which customers are most likely to leave or churn, so that they offer them promotions to persuade them to keep using their lines (Óskarsdóttir et al., 2018). More recently, the value of churn prediction models has been challenged. Meaning that, the outcome of churn prediction model is insufficient for targeting of retention campaign, since it identifies customers' risk of churning regardless their sensitivity to the retention campaign (Ascarza, 2018). Furthermore, proactively targeting high-risk customers might not be an effective strategy to reduce churn because, by doing so, companies are wasting resources on customers who are not responsive (or who may even respond negatively) to the campaign (Ascarza, 2018). Therefore, the targeting of retention campaign should be approached by means of uplift models (Devriendt et al., 2019; Gubela et al., 2020). While, churn prediction models focus only on whether a customer will churn, uplift models focus on whether a customer is intent on churning and will be retained when targeted with the campaign. Even though, uplift models fully align with business objective, churn prediction models are still useful. Churn prediction models can be used to develop uplift models according to the two-model approach (Kane et al., 2014). In this

* Corresponding author.
E-mail address: iteng.nadia@gmail.com (N. Alboukaey).

https://doi.org/10.1016/j.eswa.2020.113779
0957-4174/

approach, two churn prediction models are built: one for treatment group and the other for the control group, and the aggregated uplift model subtracts the probabilities resulting from both models to determine the uplift (Devriendt et al., 2019). Moreover, churn prediction model can be extended to learn the corresponding uplift model by adding a dummy variable to denote the group of origin for each customer, then the prediction model is developed from 1) the original predictor variables (customer's behavior features), 2) the added dummy variable and 3) additional interaction variables between the predictor and dummy variables. The resulting models allow predicting the probability of churn in both treatment and no-treatment scenarios, and the difference being the uplift (Kane et al., 2014).

Churn prediction is a management science problem for which a machine learning approach can be adopted. Based on historical data, a model can be trained to classify customers as future churners or non-churners. Research show that customer's behavioral features (e.g. calls duration, calls count, refill/bill mount, etc.) are good predictors of churn (Buckinx & Van den Poel, 2005; Hung et al., 2006; Wei & Chiu, 2002). Different prior works have looked at churn as a static prediction problem. In other words, many static features have been extracted during the whole observation period to be used as churn predictors. These static features can be: behavioral features aggregated in some way (Castanedo et al., 2014; Vafeiadis et al., 2015), mix of demographic and behavioral features (Hung et al., 2006; Wei & Chiu, 2002), or even social features inclusion (Phadke et al., 2013). Even though, these studies have proposed good predictors, the main drawback of their approach is ignoring the temporal nature of customer behavior, which may cause a loss of the discriminative ability (Chen et al., 2012). To tackle this drawback, other studies considered dynamic churn modeling. More precisely, they considered the behavioral features at monthly level, then predict churners using traditional machine learning models (Ahmad et al., 2019; Khan et al., 2015; Prashanth et al., 2017), sequential and hybrid models (Hu et al., 2018; Zhang et al., 2016) or adapted learning models that consider the sequential nature of the input (Chen et al., 2012). These studies considered dynamic behavior at monthly level; however, customer behavior is susceptible to changes over days of the month and during the time leading up to a customer decision to churn, he/she starts behaving differently. Ignoring these changes may reduce the discriminative ability of the prediction model and thus its predictive performance. For example, let x_monthly = (600, 600, 600) be a vector that represents the monthly spending of a customer in the last three months. This representation can be for a non-churning customer with a stable behavior. It also represents a churning customer that his daily spending is stable over the first sixty days (i.e. the first two months) while it has a peak followed by a slope at the beginning of the last month e.g. x_daily = (20, . . ., 20, 180, 160, 120, 100, 40, 0, 0, 0, . . .,0). Just few studies considered customer behavior at different levels rather than monthly, i.e. daily (Wangperawong et al., 2016), weekly (Óskarsdóttir et al., 2018), and hourly (Zaratiegui et al., 2015). Furthermore, all previous works predict churn at monthly level (once a month). However, in this case, it will be too late for marketing teams to do any proactive action to retain customers who decided to leave in the beginning of the last month and been identified by the monthly prediction models in the next month. The previous example also explains this case; where the customer started behaving differently at the beginning of the last month. The monthly prediction model will identify this customer as churner in the next month (i.e. 55 days after his last action), while the daily prediction will identify him in the same day (the last action day). Targeting this customer at this time with a retention offer will most probably find a response, while doing that after 55 days will not. In conclusion, daily estimates of churn propensity allows marketing team to design retention campaign at any point of time within the month depending on up-to-date propensities, not old propensities from the last month. Furthermore, daily models will identify soon the customers that start behaving differently at the beginning of the month, while monthly models could not identify those until the next month.

Therefore, in this paper, we propose the daily churn prediction based on the daily dynamic behavior of customer represented as a multivariate time series. We propose two main approaches to predict churn based on this representation, the feature-based approach and the deep learning approach. In the feature-based approach, we summarize each time series of the multivariate time series by its RFM or statistics features that are then concatenated and fed into a traditional machine learning classifier. We design two models using this approach, RFM-based model and Statistics-based model. On the other hand, in the deep learning approach, a deep learning model that can process sequential data are designed. More precisely, we design two models LSTM-based model and CNN-based model.

The main contributions of this paper can be summarized as follows:

- Addressing the daily churn prediction problem based on customer's dynamic behavior on a daily level.
- Propose two approaches to predict churn daily based on the daily behavior of customer: feature-based approach and deep learning approach.
- Compare the performance of these models with previous monthly churn prediction models based on a large dataset collected from MTN operator, which proved our models better performance for predicting potential churners earlier and more accurate.

The remainder of this paper is organized as follows: Section 2 summarizes the related work of the dynamic churn prediction. Then, the four models based on customer's daily behavior are proposed in Section 3. Section 4 discusses the experimental results. Finally, conclusions are drawn in Section 5.

2. Related work

In mobile telecom, customer defined as churner if he stops doing revenue-generating events for ninety consecutive days, these days are called inactivity period. Thus to build a model that can predict churn, the customer's behavior in the time period that precedes the inactivity period -this period is called observation period- are analyzed. In other words, some features that could represent the customer in the observation period along with the corresponding label (churn, non-churn) are fed into a binary classification model to be trained. The output of this model can be defined formally as follows:

$$ChurnScore(X, m) = P(Class = Churner | X_{[m-n,m-1]}) \quad (1)$$

Where:

- m: the current month
- n: the total number of months in the observation period
- $X_{[m-n,m-1]}$: features' values of the customer during the observation period$[m-n, m-1]$

All previous works predicted customer churn based on this definition. They mainly differ in the number of months in the observation period, the definition of the churner ($Class = Churner$), and the predictor variables used to represent the customer ($X_{[m-n,m-1]}$) that may include demographic features, behavioral features, and social features. In this paper, we are concerned with the method of rep-

resenting and processing behavioral features. Accordingly, we can classify models presented in previous works into two main categories: static-based and dynamic-based models. In static-based models, static features (non-temporal features) are used as predictor variables to identify churner using a traditional machine learning technique. Static features can be demographic, social, or behavioral features. The behavioral features are aggregated or derived in some way overall the total observation period ($[m-n, m-1]$). Different aggregation functions have been used in literature. For example, in (Vafeiadis et al., 2015) authors aggregated many behavioral features, such as call minutes and count per day and night, using the total sum in the observation period of six months. While, authors in (Castanedo et al., 2014) aggregated two main behavioral features (call duration, and refill amount) using their sums per each 30 min time interval over the complete observation period of one month. In (Hung et al., 2006; Wei & Chiu, 2002), the authors extracted many behavioral features from CDR (Call Detail Records) during the observation period of six months and aggregated them using count, sum, or average functions. The obvious drawback of aggregating behavioral features over the complete observation period is that it removes the temporal changes in customer's behavior and this may cause the loss of the discriminative ability of the prediction model. Therefore, other studies considered the dynamic behavior instead of the static one. In other words, each behavioral feature has a sequence of values during the observation period instead of just one value. Many studies considered the dynamic behavior on a monthly level. Chen et al. in (Chen et al., 2012) proposed a novel learning technique called hierarchal multiple kernel support vector machine to model both static and temporal behavioral features. Although, this technique achieved a better performance than feeding the classifier with temporal features as flattened vector, or transforming them using aggregation method, it is inconvenient when considering the behavioral features on a daily level, because it is computationally expensive, especially for large dataset and a long observation period, as in our case. In (Khan et al., 2015) authors applied Combinatoric Brute Force technique to extract behavioral features on a monthly level from Customer Detail Records in six months of observation period. These sequential features represented as a flattened vector to be fed into a traditional machine learning technique (Logistic Regression, SVM, Random Forest and KNN). In our case, where we consider daily behavioral features, the number of features will be 30 times greater. So we discuss and compare different scenarios of feeding these features to a machine learning classifier, i.e. feeding them directly (as a flattened vector) to a traditional machine learning model or aggregating them before that or employing a deep learning technique that consider the sequential nature of these features. Zhang et al. (2016) extracted behavioral features from Call Detail Records and service info in three months of observation period. They represented these behavioral features as multivariate time series (sequence length = 3), and designed a hybrid model in order to identify potential churners in several months ahead or earlier. The hybrid model included three sub models that process the multivariate time series in different ways. The first sub model clusters each univariate time series into two clusters, then the resulted clusters of these series are concatenated in one vector to be fed into Random Forest classifier. The second sub model applies the PCA (Principle Component Analysis) technique on the multivariate time series, and then the resulted principle components are concatenated to be fed into another Random Forest classifier. The third sub model is the Convolution Neural Network proposed in (Zaratiegui et al., 2015). The first two sub models are inconvenient in our case because they are computationally expensive. Whereas, the third sub model is suitable and we have designed a CNN model that take the daily behavioral features as input. In (Khan & Kozat, 2017) and (Hu et al., 2018) authors extracted monthly sequenced behavioral features and compared the performance of Long Short Term Memory (LSTM) with tradition machine learning techniques. They showed that Logistic Regression and Random Forest outperform LSTM based on monthly dynamic behavior. Although, the LSTM model is more suitable for sequence data than the tradition models, when monthly sequenced features are presented it preforms worse. However, in this work, we have designed LSTM model based on daily sequenced behavioral features and got better results than the tradition model (Random Forest).

While many studies considered the dynamic behavior on a monthly level, just few studies were concerned with dynamic behavior on other levels rather than monthly. In Wangperawong et al. (2016), the authors represented customer's behavioral features on a daily level as an image. Specifically, they constructed a two dimensional array of normalized pixels where each row correspond to a day in the observation period of 30 days and each column corresponds to a type of behavior tracked. Ten behavior types has been considered, including data usage, top up amount, top up frequency, voice calls, voice minutes, SMS messages, etc. They designed Convolution Neural Network (CNN) to predict churning customers based on that representation and showed that their model outperforms SPSS CHAID model that depends on handcrafted features and tree-based learning technique. In this paper, we have built a CNN model according to their architecture and compared it with our proposed models. In Zaratiegui et al. (2015), the authors also represented customer's behavioral features on an hourly level as an image but unlike (Wangperawong et al., 2016) they tracked three types of behavior outgoing, incoming minutes and topups every 2 h in the observation period of 4 weeks and represented these three types as RGB channels. They showed that their model provides more accurate predictions than traditional machine learning models built with handcrafted features. In Óskarsdóttir et al. (2018), the authors proposed a method for modeling the dynamic social behavior of customers on a weekly level by building ego network for each week in a total observation period of 20 week. They represented the customer's social behavior as a multivariate time series. Then, this multivariate time series was classified using Similarity Forest method (Sathe & Aggarwal, 2017), which they further extended in various ways to accommodate multivariate time series. They showed that their approach (dynamic social behavior-based Similarity Forest) performs better, compared to traditional approaches (static social behavior), when predicting further in the future, and therefore it is better at detecting churn early. While the similarity forest preforms well in this configurations, it is computationally expensive for our representation of daily behavior.

As detailed above, previous works on churn prediction focused on predicting churn monthly based on static or monthly dynamic behavior. Two main drawbacks can be drown from these works; the first one is that predicting churn monthly is late for customers that decided to leave at the beginning of the last month because the will be identified as churners by the monthly model at the next month. The second drawback is that considering the monthly behavior ignores the changes in customer's behavior over days of the month and this may reduce the discriminative ability of the prediction model, and thus, its predictive performance. Therefore, in this paper, we propose daily predicting of churn based on daily dynamic behavior. Considering daily behavior rather than the monthly one poses two main difficulties. First, the size of the data that represents customer's behavior becomes much larger ($\times 30$). This prevents many previously proposed approaches for monthly and weekly to be applied for the daily case. In other words, the methods that proposed in (Chen et al., 2012; Óskarsdóttir et al., 2018; Zhang et al., 2016) are inapplicable in our case because they are computationally expensive. Second, handcrafted feature engineering becomes more complicated. To overcome these difficulties,

in this paper, we propose two main approaches for the daily churn prediction based on customer's daily behavior: feature-based approach and deep learning approach. In the former, each behavioral feature (time series or sequence of values) is summarized using predefined functions. We design two models according to this approach, RFM-based model and Statistics-based model. While in the second approach, deep learning techniques are employed to learn the representative features from the daily behavior and predict churn. We design two models according to this approach, LSTM-based model and CNN-based model.

3. Methodology

We formulate the daily churn prediction as a binary classifier. The output of this classifier can be defined formally as follows:

$$ChurnScore(X, d) = P(Class = Churner | X_{[d-90,d-1]}) \quad (2)$$

Where:

- d: the current day
- $X_{[d-90,d-1]}$: A multivariate time series that represents customer's daily behavior during the observation period (the analysis window).

We define a churner as a customer that stops doing revenue-generating events during the next 30 days, while he was active during the observation period (from d-1 back to d-90). We chose 30 days for defining churner instead of 90, which is known in literature as partial churn, because when we analyzed the activity of the previously churned customers of our studied operator, we found that 81% of them have zero activity during the last thirty days of the observation period as shown in Fig. 1. Furthermore, we excluded customers that were inactive during the last 30 days (from d-1 back to d-30) from our final churner list, because those customers have already churned according to our definition. Moreover, the model will identify them as *high-risk customers* with high churn score (>=0.9), while they should not be targeted by any retention campaign. We investigated various automatic methods to extract meaningful features from $X_{[d-90,d-1]}$ in order to build a classifier that identifies potential churners. First, the RFM-based model has been designed in which each univariate time series in $X_{[d-90,d-1]}$ is summarized by our defined Recency, Frequency, and Monetary (RFM) values for two different levels (the total analysis window and every 30 days of it). Second, some statistics functions are employed to summarize every univariate time series in $X_{[d-90,d-1]}$ by its statistics values for two different levels too. Finally, LSTM and CNN neural networks are employed to learn representative features from the multivariate time series $(X_{[d-90,d-1]})$. In the following sub-sections, we are going to introduce the construction of $X_{[d-90,d-1]}$ (from now we will write X instead $X_{[d-90,d-1]}$ for simplicity) and these four models.

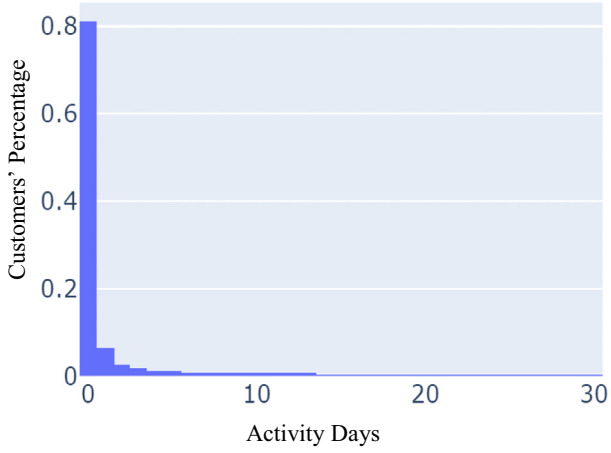

Fig. 1. The distribution of churners' activity during the last thirty days of the observation period.

3.1. Multivariate time series construction

There are several data sources available from the customer behavior point of view, such as Call Detail Records (CDRs), balance replenishment events (topups), KPI (Key Performance Indicators) Records, detailed usage records, to name a few. In this work we are concerned with temporal behavioral features. Although, demographic features are commonly used in the literature, we ignored them because they are unreliable in our studied operator. However, it is feasible for our proposed models to handle these features and any other static features along with the temporal features to make the prediction. In order to build and test our models, we just use the following data from our studied operator:

- KPI Records: daily aggregated data that summarizes customer spending on a specific day. This information includes the amounts of money spent on different types of telecom services, such as voice call, SMS, data, and other services besides balance replenishment amounts.
- Usage Records: daily aggregated data that summarizes customer behavior on a specific day. This includes the usage amounts for every type of service such as outgoing calls minutes, outgoing calls count, data volume and SMS count.

The original KPI and Usage records are provided by the operator and stored in separate tables per day. Before we construct the input data (Xmatrix) for our methods, a feature vector indicating customer behavior should be extracted for every day. This feature vector consists of six KPI features and four usage features that are listed in Table 1. To construct the X matrix, we stack these feature vectors vertically as illustrated in Fig. 2.

This process ends with a dataset D with s multivariate time series of dimension $r = 10$,and length $n = 90$, i.e. each observation (customer) $X \in D$ in the dataset consists of ten time series of length 90 correspond to ten temporal behavioral features.

$$D = \begin{pmatrix} \begin{pmatrix} f_1^1(1) & \cdots & f_r^1(1) \\ \vdots & \ddots & \vdots \\ f_1^1(n) & \cdots & f_r^1(n) \end{pmatrix} \\ \vdots \\ \begin{pmatrix} f_1^s(1) & \cdots & f_r^s(1) \\ \vdots & \ddots & \vdots \\ f_1^s(n) & \cdots & f_r^s(n) \end{pmatrix} \end{pmatrix}$$

Table 1
The features extracted from KPI and Usage records.

Name	Description
Consumption-amount	Total money expensed in a day
Call-amount	Money expensed for calls in a day
SMS-amount	Money expensed for SMS in a day
Data-amount	Money expensed for Data in a day
Refill-amount	Top-up amount in a day
Services-amount	Money expensed for services in a day
Calls-count	Number of outgoing calls in a day
Calls-duration	Total duration of outgoing calls in a day
SMS-count	Number of outgoing SMS in a day
Data-volume	MB of used data in a day

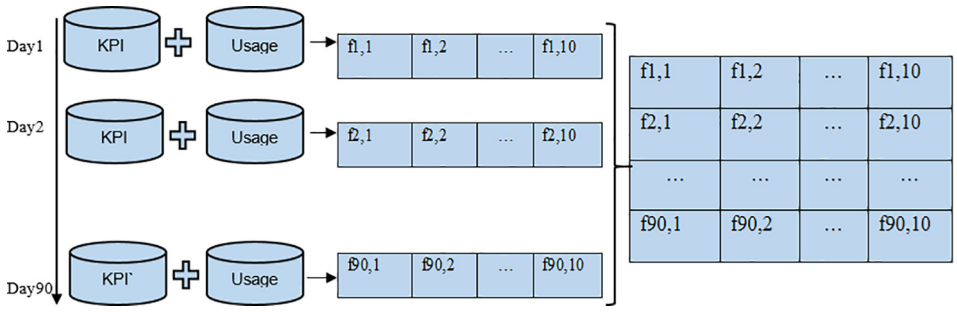

Fig. 2. Construction of customer's behavior matrix (X matrix).

Where $f_j^i(t)$ corresponds to the value of feature j for customer i at day t, $i = 1, \cdots, s$, $j = 1, \cdots, 10$, $t = 1, \cdots, 90$. Furthermore, each customer $X \in D$ in the dataset is labelled, meaning that it belongs to one of the two classes: churn or non-churn, denoted by 1 and 0 respectively. Thus, there is a label vector y:

$$y = \begin{pmatrix} l^1 \\ \vdots \\ l^s \end{pmatrix}$$

Where,

$$l^i = \begin{cases} 1 & if\ Activity(X, [d, d+30]) = 0 \\ 0 & otherwise \end{cases} \quad (3)$$

$$Activity(X, [t_1, t_2]) = \sum_{t=t_1}^{t_2} \sum_{j=1}^{10} f_j^i(t) > 0 \quad (4)$$

3.2. RFM-based model

The RFM model is a prevailing method used in the literature to characterize customer interactions and customer behavior in general. It is a simple, yet powerful method, which enables quantifying customer interactions in terms of recency, frequency and monetary dimensions. Recency is the time period since the customer's last purchase to the time of data collection; frequency is the number of purchases made by individual customer within a specified time period; and the monetary variable represents the amount of money a customer spent during a specified time period. The current literature related to churn prediction in telco devises many different RFM operationalizations (Mitrović et al., 2019). However, in this paper, we define RFM values for behavioral feature rather than the customer. More precisely, we summarize each temporal behavioral feature (each column in the X matrix) F_i by our defined Recency, Frequency, and Monetary values during multiple time windows.

Definition 1. *we define the Recency of F_i during a specific time window $[t_1, t_2]$, (where $t_2 - t_1 \leq 90$) as the number of time steps (days) from the time step of last non-zero value of F_i (ts) till t_2:*

$$R(F_i, t_1, t_2) = \sum_{ts}^{t_2} (1)\ if\ ts < t_2\ else\ 0 \quad (5)$$

Definition 2. *we define the Frequency of F_i during a specific time window $[t_1, t_2]$ (where $t_2 - t_1 \leq 90$) as the number of time steps (days) with non-zero values of F_i:*

$$Freq(F_i, t_1, t_2) = \sum_{t=t1}^{t2} (f_i(t) > 0) \quad (6)$$

Definition 3. *we define the Monetary of F_i during a specific time window $[t_1, t_2]$ (where $t_2 - t_1 \leq 90$) as the total sum of F_i values during $[t_1, t_2]$:*

$$M(F_i, t_1, t_2) = \sum_{t=t1}^{t2} f_i(t) \quad (7)$$

RFM values for each temporal feature F_i in X have been extracted in two levels:

- The total analysis window, with 90 days length.
- Consecutive sub windows of the analysis window, each with 30 days length.

Hence, the total RFM features vector for a temporal feature F_i can be defined as follows:

$$RFM(F_i) = [R(F_i, 1, 90), Freq(F_i, 1, 90), M(F_i, 1, 90), R(F_i, t_1 + 1, t_1 + 30), Freq(F_i, t_1 + 1, t_1 + 30), M(F_i, t_1 + 1, t_1 + 30)] where t_1 \in \{0, 30, 60\} \quad (8)$$

Applying RFM functions on a customer X, in the dataset D, gives a row vector of 120 values as representative features for customer's temporal behavior during the observation period. Thus, we get a transformed dataset $\widehat{D}$:

$$\widehat{D} = RFM(D) = \begin{pmatrix} RFM\left(F_1^1\right) & \cdots & RFM\left(F_{10}^1\right) \\ \vdots & \ddots & \vdots \\ RFM\left(F_1^s\right) & \cdots & RFM\left(F_{10}^s\right) \end{pmatrix}$$

The transformed dataset $\widehat{D}$ is a 2D matrix, each row represents a customer and each column represents a feature. Therefore, we can apply any traditional machine learning technique for binary classification to predict churn based on this representation. We have tested two main types of binary classifiers: linear binary classifier (Logistic Regression) and non-linear one (Random Forest). The Random Forest had the best performance, with 100 trees and maximum depth = 10. The RFM-based model is shown in Fig. 3.

While this model in its current architecture takes only temporal behavioral features, considering static features (such as demographic features) along with these features is feasible. This can be done by concatenating the new features with the RFM features vector just before feeding them to the Random Forest.

3.3. Statistics-based model

In this model, each behavioral feature is summarized by various descriptive statistics values instead of RFM values. These statistics are: Minimum, Maximum, Mean, variance, first quartile, second quartile (median), third quartile, skewness, kurtosis. These values are computed for each behavior type in two levels too: the total

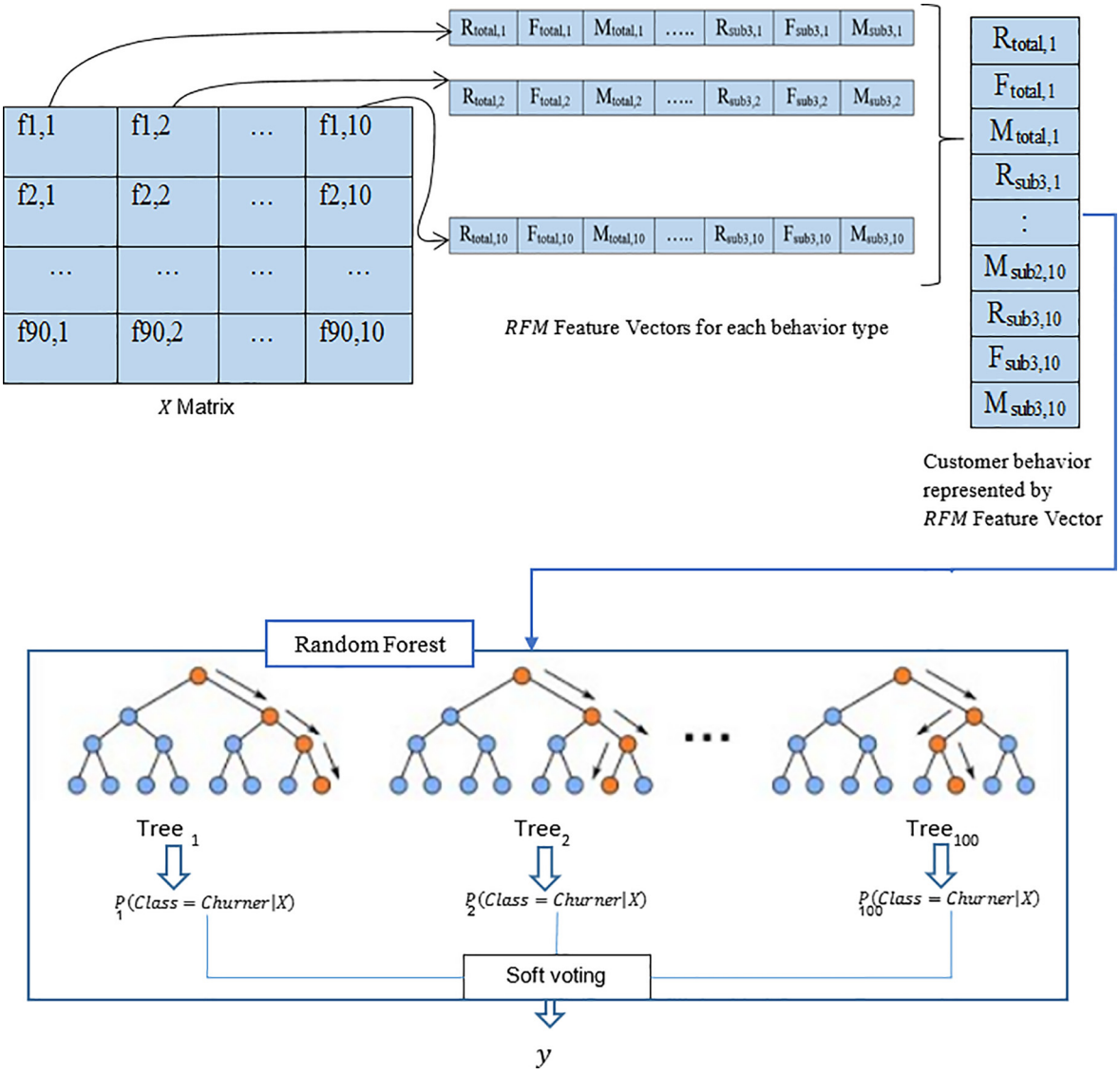

Fig. 3. RFM-based Model Architecture.

analysis window (i.e. 90 days) and every 30 days of it, so we get a vector of 36 value for each behavior type. Then, these vectors are concatenated to construct the final vector that represents customer behavior with 360 values. These feature vectors can be classified using any machine learning classifier. We tested Logistic Regression and Random Forest and the Random Forest classifier with 100 trees and maximum depth = 15 was the best.

While this model in its current architecture takes only temporal behavioral features, considering static features along with these features is feasible. This can be done by concatenating the new features with the feature vector of statistics just before feeding them to the Random Forest.

3.4. CNN-based model

Convolutional neural network is a Deep Learning model which can take in an input 2D or 3D matrix. CNN model is composed of two parts. The first part contains convolution and pooling operations that are used to learn deep features of the raw data. The second part contains flatten and fully connected layers that are used to classify the input based on the features learned in the first part. The convolutional layer in the CNN model learns different aggregation functions (number of functions = kernels) over multiple rolling time windows of length equals to kernel size. The pooling layer samples the output of the convolutional layer. Average pooling layer outputs the average value for each group of values, the number of values in the group is defined by the pool-size parameter.

CNNs are widely used in many image processing tasks and have demonstrated superior performance. However, in churn prediction domain there are just a few previous works that had employed this type of networks (Wangperawong et al., 2016; Zaratiegui et al., 2015). In this paper, we propose a different architecture for CNN and we will compare ours with that proposed in (Wangperawong et al., 2016) as we both work on daily behavior. Whilst, we cannot compare ours with (Zaratiegui et al., 2015) because we do not have the data on an hourly level, besides we have 10 types of behavior not just 3 to be mapped to RGB channels as in their representation.

We employed CNN to learn representative features of the customer from his daily behavior. Mainly, we used two one-dimensional convolutional layers in parallel, as shown in Fig. 4. One convolutional layer for extracting the weekly patterns in customer behavior (named Conv1D-weekly) and another layer for extracting the monthly patterns (named Conv1D-monthly). One-dimensional convolutional layer is considered here to include all

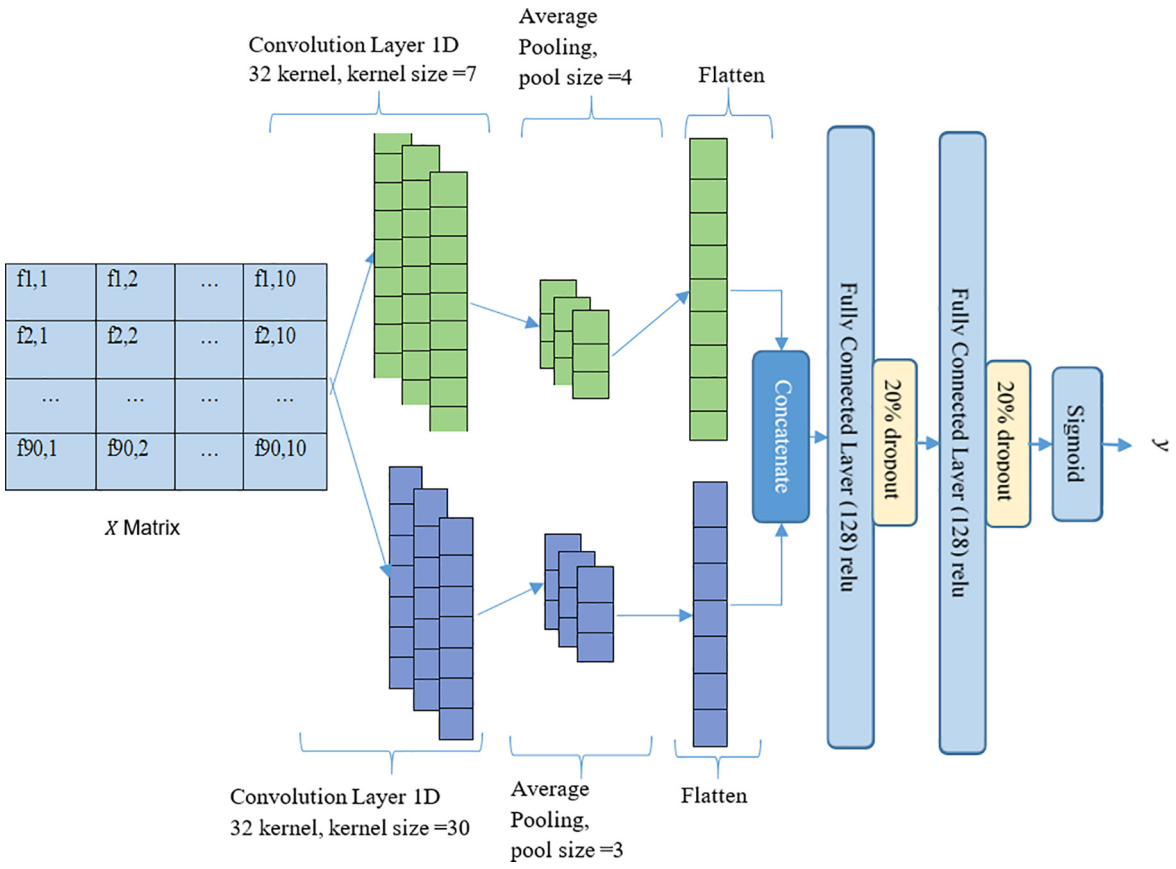

Fig. 4. CNN-based Model Architecture.

the behavioral features at the same time in the convolution process. The one-dimensional convolutional layer in the CNN model consists of multiple kernels. Each kernel corresponds to a moving weighted sum function with learnable weights shared over time. The length of window for this function equals to the kernel size. The Conv1D-weekly layer has 32 kernels of length 7 (number of days in a week). Moreover, the Conv1D-monthly layer has 32 kernels of length 30 (number of days in month). Each kernel convolve over the time dimension and applies an aggregation of all behavioral features over a number of days equals to the kernel size. This operation is identical to summarizing a time series using the moving sum with window length of 7 or 30, when all values of the kernel are equal to one. The output of the Conv1D-weekly are then processed by an average pooling layer with pool-size = 4, i.e. the average value of each four aggregated weeks is outputted. Similarly, the output the Conv1D-monthly layer is processed by another average pooling layer of size = 3. i.e. the average value of each three aggregated months is outputted. After pooling, the two outputs are flattened and concatenated to be fed into two fully connected layers each with 128 units and 20% dropout, then to the output layer of one sigmoid node. The all parameters and layers' details are listed in Table 2. While this model in its current architecture takes only temporal behavioral features, considering static features along with these features is feasible. This can be done by adding an input layer just before the first fully connected layer and concatenate this input layer with the input of the fully connected layer.

3.5. LSTM-based model

The Long short term memory (LSTM) is a variant of Recurrent Neural Network (RNN) that is capable of learning long term dependencies. LSTMs were first proposed by Hochreiter and Schmidhuber (Hochreiter & Schmidhuber, 1997) and refined by many other researchers. LSTMs work well on a large variety of problems and are the most widely used type of RNN. RNN uses the hidden state

Table 2
CNN-based Model layers' detail.

Layer	Kernels/hidden unites	Kernel/pool size	Output dim	stride	Activation
Input	–	–	90x10	–	
Conv1D-weekly	32	7	84x32	1	relu
Conv1D-monthly	32	30	61x32	1	relu
Average Pooling-weekly	–	4	21x32	–	–
Average Pooling-monthly	–	3	20x32	–	–
Fully connected	128	–	128x1	–	relu
Dropout 0.2	–	–	–	–	–
Fully connected	128	–	128x1	–	relu
Dropout 0.2	–	–	–	–	–
Output	1	–	1x1		sigmoid

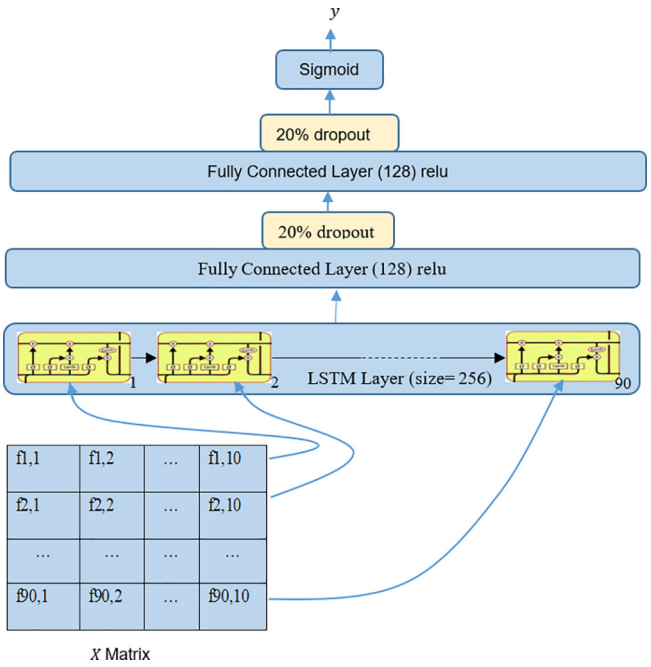

Fig. 5. LSTM-based Model Architecture.

from the previous time step and the current input in a tanh layer to implement recurrence. LSTMs also implement recurrence in a similar way, but instead of a single tanh layer, there are four layers (called gates) interacting in a very specific way. These gates constitute the LSTM cell (the yellow block in Fig. 5). The core concept of LSTM is the cell state, and its various gates. The cell state acts as a transport highway that transfers relative information all the way down the sequence chain. It is the memory of the network. The cell state can carry relevant information throughout the processing of the sequence. Therefore, even information from the earlier time steps can make its way to later time steps, reducing the effects of short-term memory. As the cell state goes on its journey, information is added or removed to the cell state via gates. The gates can learn what information is relevant to keep or forget during training. This learning mechanism is very suitable for churn prediction, where the LSTM remembers the recent and the old behavior of a customer and can decide the label based on that. LSTM is the best known type of deep neural networks to deal with sequential data and has demonstrated superior performance in natural language processing tasks. Whereas, in churn prediction domain, there are just few works that investigated this type of deep learning technique on monthly behavioral features. In that scenario, LSTM did not outperform the traditional machine learning models. However, in this paper we used LSTM and fed it with daily behavioral features. More precisely, we designed LSTM-based model that consists of four layers, LSTM layer and three fully connected layers as shown in Fig. 5. The LSTM layer processes ten behavioral features during 90 days. This LSTM layer consists of 256 LSTM units, each one has 90 cells that correspond to 90 days. In each time step (day), all behavioral features follow throughout the four gates of the LSTM cell. The LSTM layer followed by two fully connected layers of 128 units, with 0.2 dropout, and finally the output layer of one sigmoid unit.

While this model in its current architecture takes only temporal behavioral features, considering static features along with these features is feasible, and can be done by adding an input layer just before the first fully connected layer and concatenate this input layer with the input of the fully connected layer.

4. Experimental results

In this section, we present our experiments for validating the proposed models of daily churn prediction and comparing them with different approaches for daily and monthly churn prediction that have been discussed in Section 2. More precisely our models are compared against these approaches:

1) The direct approach that has been proposed by previous works to handle the monthly dynamic behavior. In this approach the temporal features are fed directly to a tradition machine learning such as Random Forest. Accordingly, we build a model named **RF-Daily.**
2) The daily behavior-based churn prediction model using the CNN model that has been presented in (Wangperawong et al., 2016). We named it **CNN-prev**.
3) The most commonly used model for the monthly churn prediction, Random forest based on monthly behavioral features. We named it **RF-Monthly.**
4) The LSTM approach based on the monthly behavioral features. We named it **LSTM-Monthly**.

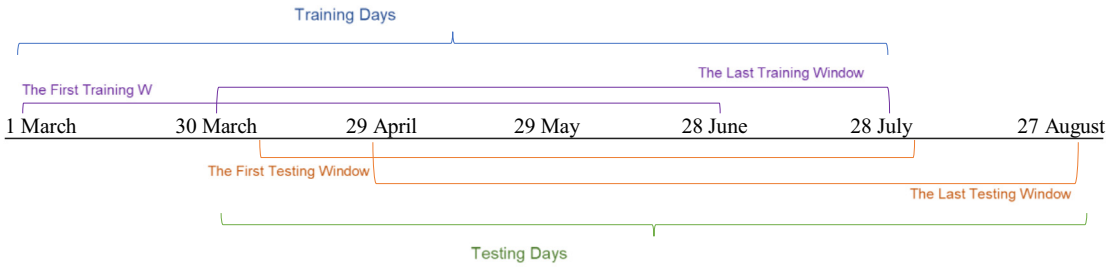

Fig. 6. Time windows for constructing the datasets.

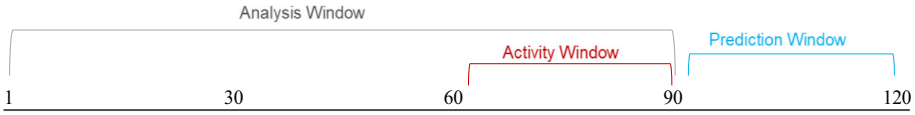

Fig. 7. The sub-windows of a 120-day time window.

Table 3
hyper-parameter space and best values.

Model Name	Explored hyper-parameters' values	The best hyper-parameters' values
RFM-based Model	n_estimator = {10, 50, 100, 200} max_depth = {10, 15, 20}	n_estimator = 100, max_depth = 10
Statistics-based Model	n_estimator = {10, 50, 100, 200} max_depth = {10, 15, 20}	n_estimator = 100, max_depth = 15
CNN-based Model	Kernels = {8, 16 ,32}, Stride = {1,7,30} fully-connected layers' size = {64, 128, 256}	Kernels = 32, Stride = 1 fully-connected layers' size = 128
LSTM-based Model	LSTM layer's size = {16, 64, 128, 256, 512} fully-connected layers' size = {64, 128, 256}	LSTM layer's size = 256, fully-connected layers' size = 128
RF-Monthly	n_estimator = {10, 50, 100, 200} max_depth = {10, 15, 20}	n_estimator = 100, max_depth = 10
LSTM-Monthly	LSTM layer's size = {32, 64, 128} fully-connected layers' size = {32, 64, 128}	LSTM layer's size = 32, fully-connected layers' size = 64
RF-Daily	n_estimator = {10, 50, 100, 200} max_depth = {10, 15, 20}	n_estimator = 100, max_depth = 20

Table 4
the mean values for the performance metrics the three best values for each metric are shown in boldface.

Frequency of Input	Features of Time Series	ML Algorithm	Model	AUC	F1-Score	Log Loss	Lift	EMPC
Daily	Raw	Random Forest	RF-Daily	0.894	0.466	0.228	4.511	3.017
		Deep Learning	CNN-based Model	**0.904**	**0.520**	**0.215**	**4.832**	**3.173**
			LSTM-based Model	**0.914**	**0.542**	**0.204**	**5.067**	**3.315**
			CNN_prev	0.885	0.452	0.23	4.478	2.905
	RFM	Random Forest	RFM-based Model	**0.906**	**0.525**	**0.212**	**4.922**	**3.186**
	Statistics		Statistics-based Model	0.877	0.491	0.233	4.661	2.752
Monthly	Raw	Random Forest	RF-Monthly	0.858	0.456	0.245	4.467	2.553
		Deep ML	LSTM-Monthly	0.852	0.447	0.248	4.413	2.536

Table 5
The standard deviation values for the performance metrics.

Frequency of Input	Features of Time Series	ML Algorithm	Model	AUC	F1-Score	Log Loss	Lift	EMPC
Daily	Raw	Random Forest	RF-Daily	0.003	0.011	0.004	0.076	0.212
		Deep Learning	CNN-based Model	0.004	0.013	0.003	0.067	0.227
			LSTM-based Model	0.004	0.014	0.003	0.077	0.24
			CNN_prev	0.004	0.01	0.004	0.077	0.218
	RFM	Random Forest	RFM-based Model	0.003	0.013	0.004	0.079	0.217
	Statistics		Statistics-based Model	0.005	0.016	0.004	0.088	0.218
Monthly	Raw	Random Forest	RF-Monthly	0.004	0.01	0.005	0.083	0.178
		Deep Learning	LSTM-Monthly	0.004	0.01	0.005	0.078	0.181

Table 6
Average ranks of the models measured by the five performance measures.

	AUC	F1-Score	Log Loss	Lift	EMPC	Overall Average Rank
LSTM-based Model	1.00	1.00	1.00	1.00	1.00	1.00
RFM-based Model	2.00	2.03	2.00	2.00	2.23	2.05
CNN-based Model	3.00	2.97	3.00	3.07	2.77	2.96
RF-Daily	4.00	5.07	4.33	5.63	4.00	4.61
Statistics-based Model	6.00	4.00	5.90	3.97	6.00	5.17
CNN_prev	5.00	6.87	4.77	6.30	5.00	5.59
RF-Monthly	7.03	6.23	7.00	6.37	7.03	6.73
LSTM-Monthly	8.00	7.83	8.00	7.67	7.97	7.89

Table 7
The values of performance metrics on the validation data.

Frequency of Input	Features of Time Series	ML Algorithm	Model	AUC	F1-Score	Log Loss	Lift	EMPC
Daily	Raw	Random Forest	RF-Daily	0.895	0.486	0.25	4.188	4.002
		Deep Learning	CNN-based Model	0.905	0.549	0.235	4.542	4.145
			LSTM-based Model	0.918	0.571	0.21	4.891	4.343
			CNN_prev	0.883	0.466	0.256	4.126	3.806
	RFM	Random Forest	RFM-based Model	0.908	0.56	0.232	4.663	4.166
	Statistics		Statistics-based Model	0.888	0.548	0.248	4.754	3.745
Monthly	Raw	Random Forest	RF-Monthly	0.853	0.474	0.274	4.199	3.308
		Deep Learning	LSTM-Monthly	0.844	0.463	0.279	4.146	3.235

Table 8
Training and prediction runtimes.

	RFM-based Model	Statistics-based Model	CNN-based Model	LSTM-based Model	RF-Monthly	LSTM-Monthly	RF-Daily	CNN_prev
Training	10.64 *m*	34.45 *m*	23.06 *m*	33.8 *m*	5.2 *m*	7.38 *m*	14.59 *m*	9.29 *m*
Prediction	43.42 s	340.22 s	12.28 s	86.63 s	24.99 s	17.9 s	72.17 s	11.04 s

Table 9
exact p-values of the statistical test on AUCs.

	RF-Monthly	RF-Daily	LSTM-Monthly	Statistics-based Model	RFM-based Model	LSTM-based Model	CNN-based Model
RF-Daily	1.76E−05						
LSTM-Monthly	0.120	5.38E−10					
Statistics-based Model	0.120	0.0124	0.0124				
RFM-based Model	7.97E−17	0.0124	1.21E−27	5.38E−10			
LSTM-based Model	1.21E−27	1.76E−05	2.01E−51	7.97E−17	0.1202		
CNN-based Model	5.38E−10	0.1202	7.97E−17	1.76E−05	0.1202	0.0124	
CNN_prev	0.0124	0.1202	1.76E−05	0.1202	1.76E−05	5.38E−10	0.0124

Table 10
exact p-values of the statistical test on F1-score.

	RF-Monthly	RF-Daily	LSTM-Monthly	Statistics-based Model	RFM-based Model	LSTM-based Model	CNN-based Model
RF-Daily	0.206						
LSTM-Monthly	0.095	0.0001					
Statistics-based Model	0.005	0.221	4.33E−09				
RFM-based Model	4.23E−11	1.50E−05	5.44E−25	0.018			
LSTM-based Model	7.20E−19	2.48E−10	1.66E−44	2.02E−05	0.221		
CNN-based Model	1.94E−06	0.009	9.94E−16	0.221	0.295	0.018	
CNN_prev	0.330	0.040	0.266	5.87E−05	1.83E−15	7.98E−26	2.01E−09

Table 11
exact p-values of the statistical test on Log Loss.

	RF-Monthly	RF-Daily	LSTM-Monthly	Statistics-based Model	RFM-based Model	LSTM-based Model	CNN-based Model
RF-Daily	0.0002						
LSTM-Monthly	0.24	3.06E−08					
Statistics-based Model	0.18	0.11	0.008				
RFM-based Model	8.32E−17	0.002	1.21E−27	2.11E−09			
LSTM-based Model	1.21E−27	1.01E−06	2.01E−51	5.36E−16	0.24		
CNN-based Model	6.23E−10	0.15	8.32E−17	4.74E−05	0.24	0.015	
CNN_prev	0.004	0.511	2.53E−06	0.18	0.0001	1.01E−08	0.048

Table 12
exact p-values of the statistical test on Lift.

	RF-Monthly	RF-Daily	LSTM-Monthly	Statistics-based Model	RFM-based Model	LSTM-based Model	CNN-based Model
RF-Daily	0.516						
LSTM-Monthly	0.280	0.012					
Statistics-based Model	0.001	0.078	2.20E−08				
RFM-based Model	3.80E−12	4.46E−08	2.30E−23	0.018			
LSTM-based Model	3.82E−20	5.90E−14	7.96E−40	2.69E−05	0.406		
CNN-based Model	1.34E−06	0.0006	1.02E−13	0.457	0.387	0.01	
CNN_prev	0.937	0.610	0.227	0.002	9.68E−12	1.66E−19	2.53E−06

Table 13
exact p-values of the statistical test on EMPC.

	RF-Monthly	RF-Daily	LSTM-Monthly	Statistics-based Model	RFM-based Model	LSTM-based Model	CNN-based Model
RF-Daily	1.41E−05						
LSTM-Monthly	0.295	8.68E−10					
Statistics-based Model	0.012	0.015	0.017				
RFM-based Model	3.33E−15	0.042	3.59E−24	9.61E−09			
LSTM-based Model	4.20E−28	1.89E−05	1.22E−49	8.32E−17	0.1968		
CNN-based Model	1.64E−11	0.196	1.47E−18	2.68E−06	0.415	0.042	
CNN_prev	0.012	0.240	2.52E−05	0.240	0.0001	5.66E−10	0.005

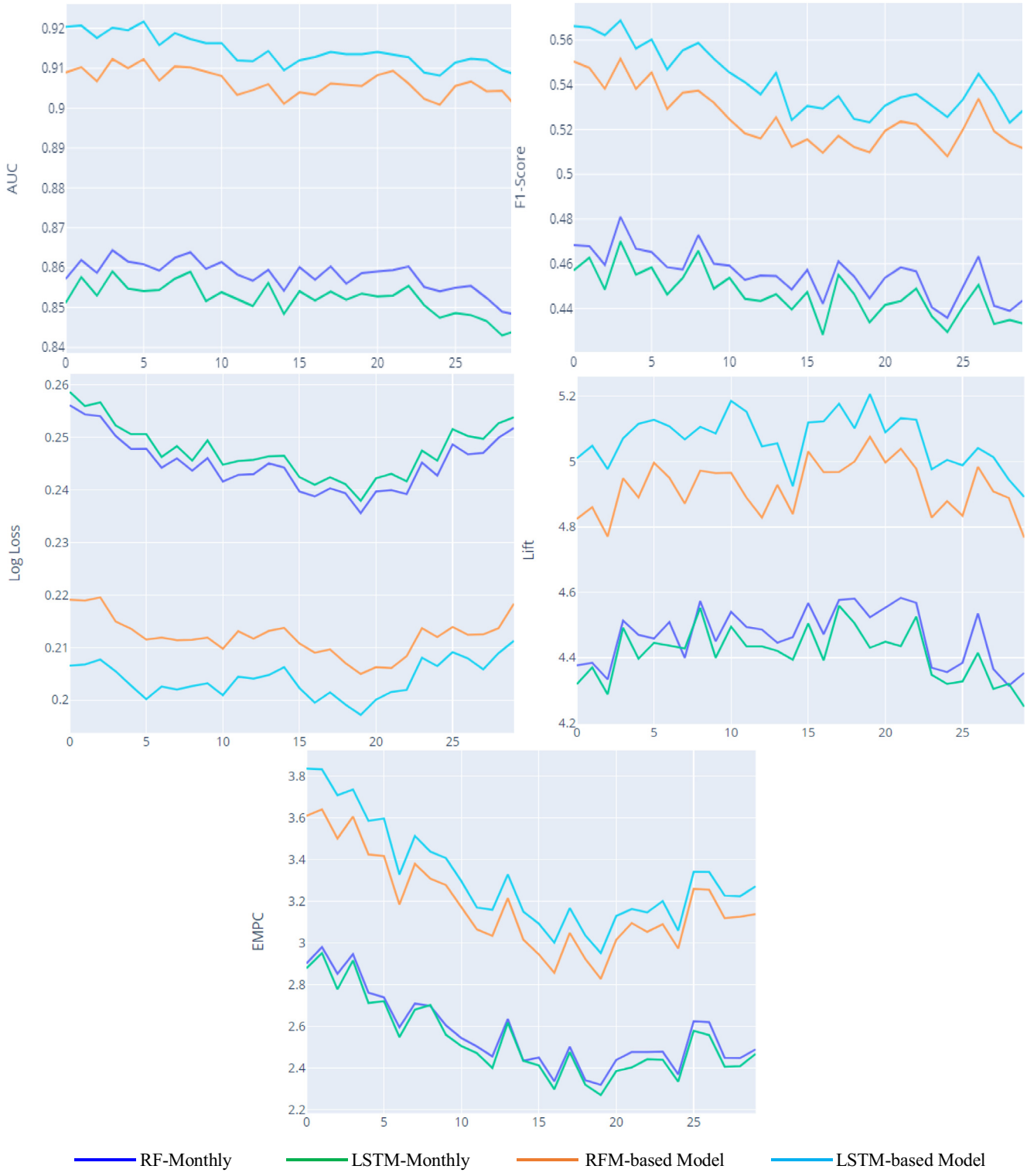

Fig. 8. Performance by day for daily and monthly models.

Furthermore, we provide an analysis of the performance of the daily and monthly models in the future time window to prove the merit of daily churn prediction in identifying churner earlier

For the evaluation and comparison of models, we have used five metrics: The Area under the Curve (AUC), the logarithmic likelihood of the classification (log loss), the weighted harmonic mean of the precision and recall of classification (F1-score). In addition to the two commonly used metrics: the top decile lift (Lift) which represents the ratio of churners in the 10% of the highest predicted probabilities to the ratio of churners in the actual customer base, and the expected maximum profit measure for churn (EMPC). The AUC metric is considered a more general metric because it summarizes the overall performance in all possible cutoffs. The EMPC metric is a recent performance measure designed specially to measure performance of churn prediction models while taking into account the cost and expected return of a retention campaign

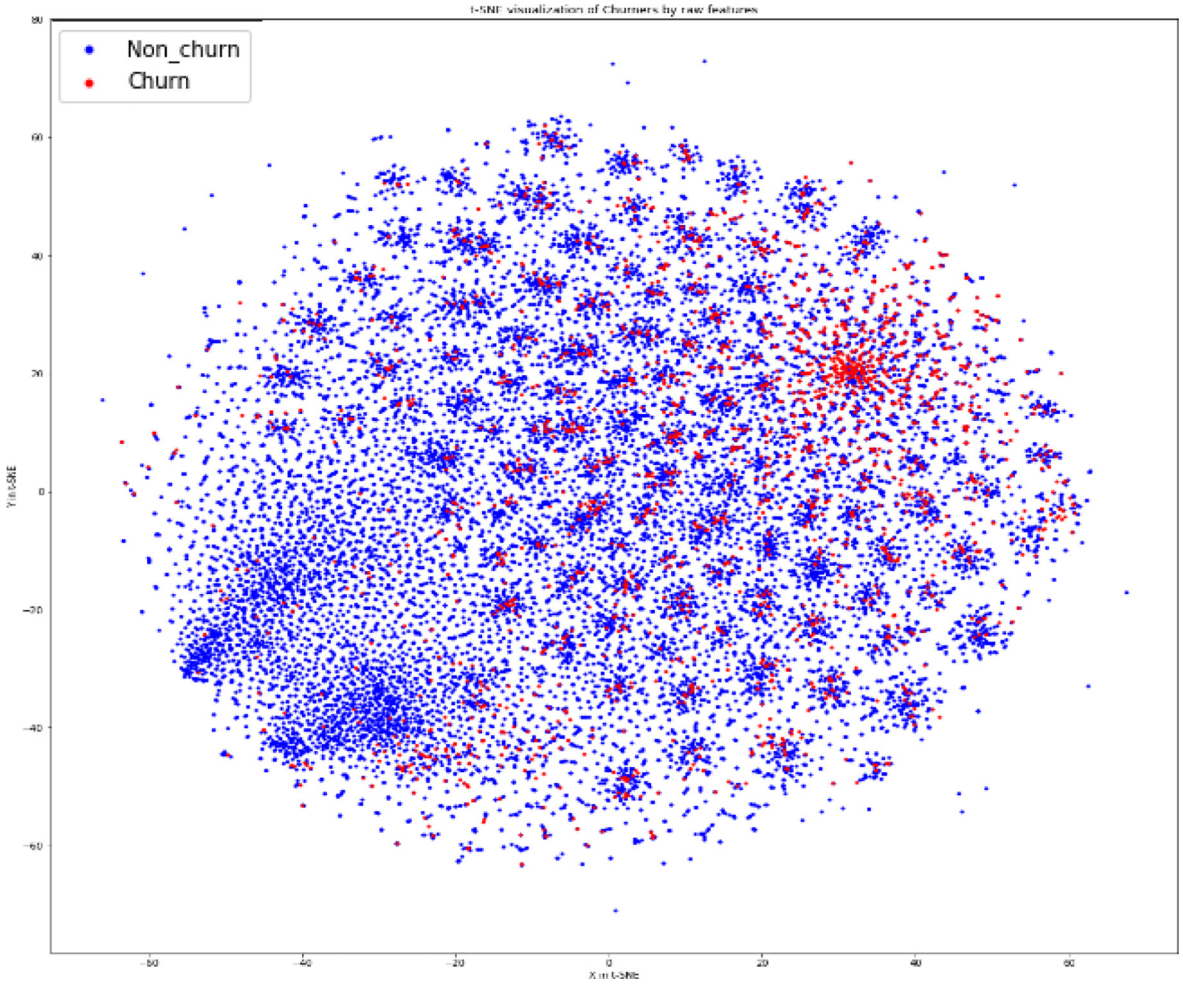

Fig. 9. t-SNE visualization of raw representation of customer daily behavior.

(Verbraken et al., 2013). We used the default parameter values ($\alpha = 6, \beta = 14, CLV = 200, d = 10, f = 1$) when computing the EMPC metric as suggested in Verbraken et al. (2013).

4.1. Dataset description

For the purpose of our research, real customer data have been provided by MTN Operator in our country. Our dataset consists of KPI and detailed usage tables from March 2019 to August 2019. KPI tables cover daily customers' spending in different usage types (including call, SMS, data, and services) and refill amounts. The detailed usage tables contain daily customers' usage amount of outgoing calls, SMS and data volume. The amount of customers in these tables for each day is more than 5 million. We use the first 150 days for generating training and validation datasets, while the last 150 days are used for generating the testing datasets. Training and testing days are further split into thirty time-shifted windows from 1st March to 30th March and from 31th March to 29th April, for the training and the testing days respectively, as shown in Fig. 6. Each time-shifted window is 120-day-length. Furthermore, these 120-day-length windows have been split into three sub windows: the analysis window, the activity window, and the prediction window as shown in Fig. 7. The analysis window is the first 90 days of the 120-day-length window, and it is used to construct the multivariate time series dataset as described in Section 3.1. The activity window is last 30 days of the analysis window. It is used to exclude the already churned customers (those that do not do any revenue generating event during this time window). The prediction window is the next 30 days to the analysis window. It is used to label the customers as churn or non-churn according to Eq. (3). In each 120-day-length window, we randomly choose 50 K customers from the total active customers during the activity window of that window. Finally, the generated observations from the training days have been concatenated in one dataset of 1.5 million observations; 12% of them are churners according to our definition in Eq. (3). This dataset has been split into two datasets: training dataset of 1.35 million observations (90%) and validation dataset of 150 thousand observations (10%). On the other hand, the generated observations from the thirty 120-day-length windows during the testing days constitute thirty testing datasets each of 50 thousands observations. In other words, testing the models has been done for 30 prediction windows with a rolling horizon from 29th July to 27th August in order to evaluate the robustness of the models.

4.2. Experimental setup

In order to effectively evaluate how well the proposed models performed, they were compared against the direct approach, the Random Forest on the daily behavioral features (**RF-Daily**). Moreover, they were compared against three models proposed by previous studies. These models are: Random Forest on a monthly behavior (**RF-Monthly**), LSTM also on a monthly behavior (**LSTM-Monthly**) and finally the **CNN_prev** model proposed in (Wangperawong et al., 2016). All these models were trained, validated and tested using the same datasets. These models and ours

were implemented in python 3 using scikit-learn 0.22.0, and keras 2.2.4 with tensorflow 2.1.0 back end. The deployment of the models was performed on a PC with an Intet Core i7 CPU@2.8 GHz, 1060 GPU and 16 GB of RAM. To build the RF-Daily model, the daily behavioral features represented by the multivariate time series X are flattened into a row vector and fed into Random forest classifier. For building the RF-Monthly and LSTM-Monthly models the daily behavioral features listed in Table 1 were aggregated monthly. Therefore the X matrix (which captures customer's behavior) for this model will be of size 3X10 instead of 90X10.

We tune the hyper-parameters of the models using the Grid Search technique. The hyper-parameters space and the best values of these parameters for every model are listed in Table 3.

On the other hand, we built the CNN model proposed in (Wangperawong et al., 2016). This model comprises of a $12 \times 7 \times 1$ convolutional layer with 0.25 dropout, followed by a 2×1 max pooling layer, a $7 \times 1 \times 12$ convolutional layer, a 2×1 max pooling layer, a fully-connected layer of 100 units with 0.2 dropout, a fully-connected layer of 40 units with 0.2 dropout, a fully-connected layer of 20 units with 0.2 dropout, and a softmax output of two units for the binary classification, as it is mentioned in their paper.

4.3. Result analysis

All the models were tested on multiple prediction time windows. More precisely, they were tested on 30 prediction time windows with a rolling horizon. The first test prediction window starts at 30th June and ends at 29th July, and the last test prediction window is from 29th July to 27th August. The mean and the standard deviation values of the five performance metrics for the models on the test prediction windows are listed in Tables 4 and 5. In addition, the average ranks of the models measured by the five performance measures and the overall average ranks are shown in Table 6. Furthermore, to check models' generalization ability, the values of the performance metrics on the validation dataset are presented in Table 7. In these tables, the models have been arranged according to three aspects: a) the frequency of the input data (monthly vs daily), b) the features of the time series data (raw, RFM and statistics), c) the machine-learning algorithm (Random Forest, Deep Learning-CNN and LSTM). On the other hand, to check models' operational efficiency, the training and prediction runtimes of the models, for the training and testing datasets described in Section 4.1, are listed in Table 8. The prediction runtimes of deep learning were recorded with batch size = 5000 samples.

When comparing the models' performance on the validation and testing datasets, we observe that the differences between the values of performance metrics listed in Table 4 and those listed in Table 7 are small. This proves that all models generalize well and there is no overfitting.

When looking at training and prediction runtimes listed in Table 8, we observe that the daily models are more time consuming than monthly ones in terms of training runtimes. This is due to the greater amount of data that has been handled in daily models, it is 30 times greater. The longest training and prediction runtimes are for the Statistics-based Model: 34.45 and 5.67 min respectively. That is due to the cost of extracting statistical features from the multivariate time series. On the other hand, the proposed daily

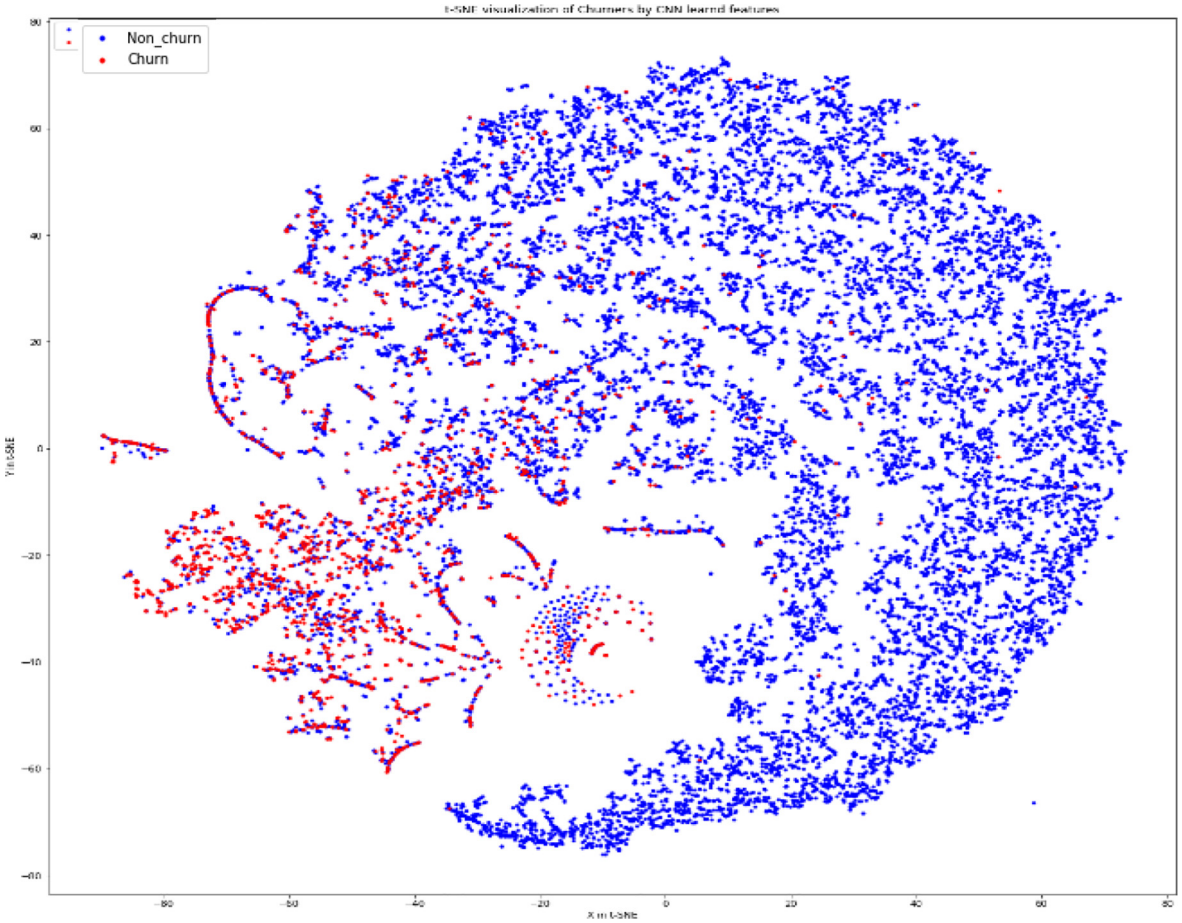

Fig. 10. t-SNE visualization of learned representation by CNN.

models are varied in terms of training and prediction runtimes. RFM-based model is faster to train and predict than Statistics-based model. In addition, CNN-based model is faster to train and predict than LSTM-based model. Even though daily models need more time to train than monthly models, and that they vary in terms of training and prediction runtimes, all of these models are operationally feasible and their training and prediction runtimes are very reasonable and acceptable for churn prediction purposes.

Many observations can be made when looking at Tables 4 and 6. First, the average ranks of the monthly models (RF-Monthly and LSTM-Monthly) measured by the five metrics are lower than the average ranks of the daily models. In other words, the models that consider the dynamic behavior on a daily level are more accurate than the models that do not. That proves the validity of our hypothesis that the dynamic changes in the customer behavior within the days of month should be considered to predict potential churners more accurately. Second, the three daily models: the LSTM-based model, the CNN-based and the RFM-based model have higher ranks, measured by all metrics, than the RF-Daily model. However, the overall average rank of the Statistics-based model is lower than the RF-Daily model's rank. As a result, feeding the raw features of the multivariate time series, that represents the customer's daily behavior, into a well-designed deep neural network (LSTM or CNN) or feeding the RFM features of that time series into a Random Forest classifier results in more accurate predictions than feeding the raw features as a flattened vector to a Random Forest. However, feeding the statistical features of the time series to a Random Forest is worse than feeding the flattened raw features. In conclusion, we can say that the daily frequency of the input is the first contributor to the accuracy, and the RFM features of the time series or the raw features with a well-designed deep learning model is the second contributor to it. However, statistical features don't contribute to the accuracy at all.

To test whether there is a significant difference between the daily and the monthly models on one hand, and between our proposed models on the other hand, we apply the statistical test for pairwise comparisons proposed in (Eisinga et al., 2017). The resulted pairwise p-values for the four performance metrics are listed in Tables 9–13. These p-values show statistical differences in performance measured in two metrics at least between any two daily and monthly models. Furthermore, there are statistical differences in performance measured in all metrics between the monthly models (RF-Monthly and LSTM-Monthly) and each of the CNN-based, the RFM-based and LSTM-based models. However, there is no statistical difference in performance neither between the RFM-based and the LSTM-based models, nor between the RFM-based and the CNN-based models. Nevertheless, the LSTM-based model significantly outperforms the CNN-based model in terms of all metrics. This can be explained by the fact that LSTM is more suitable to mimic the sequential nature of the customer's behavior (its learning mechanism enables it to remember the recent and oldest patterns). This feature makes it more suitable in the context of churn prediction, because the recent behavior of new churner is often very different from his behavior far in the past. We have investigated many hybrid architectures of CNN and LSTM by stacking LSTM and CNN layers, i.e. feeding the CNN with the returned sequence of LSTM or passing the input to CNN layer then process the returned sequence by LSTM layer, but no additional improvement has been gained. While the RFM-based and the LSTM-based models are equal in terms of prediction

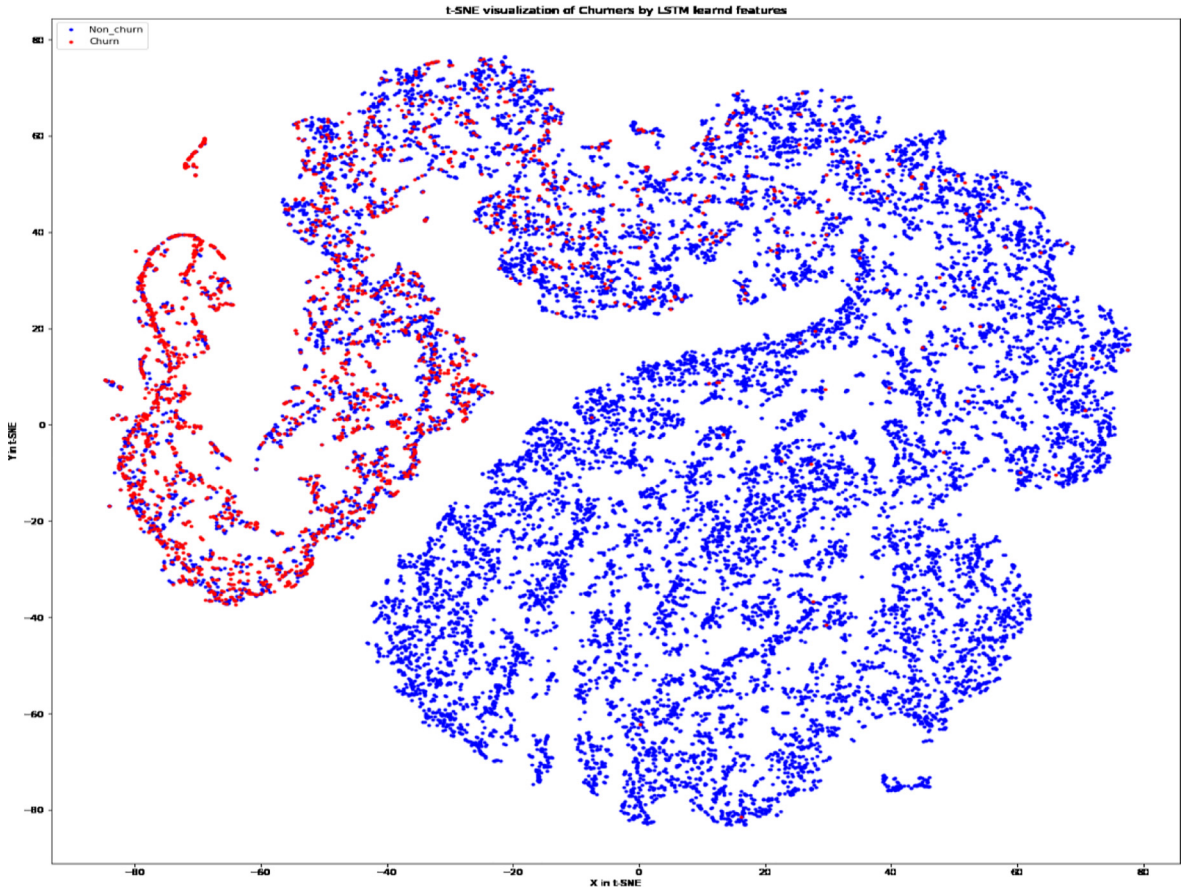

Fig. 11. t-SNE visualization of learned representation by LSTM.

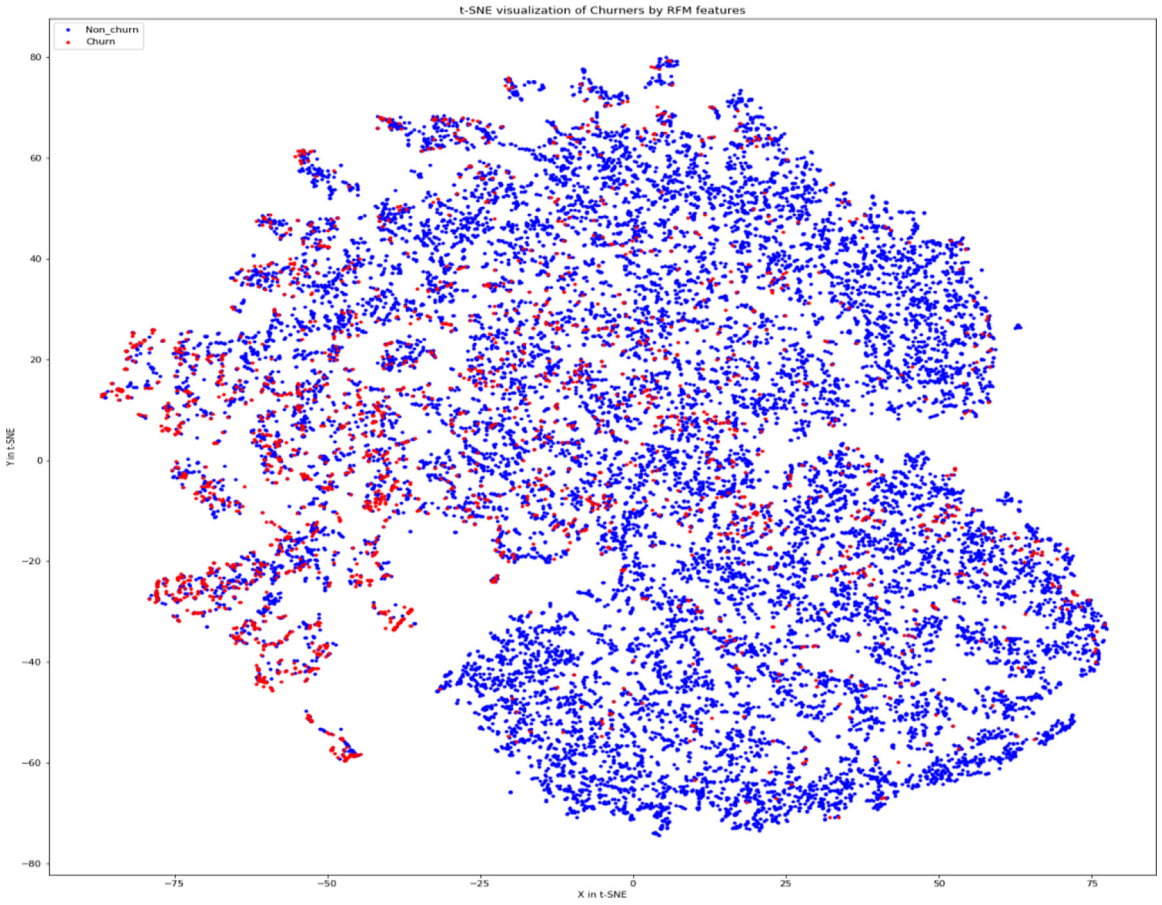

Fig. 12. t-SNE visualization of customers represented by RFM features.

performance, the RFM-base model is more preferred to be used in business, because it is simpler to build and its features (RFM features) are more interpretable than LSTM extracted features.

One the other hand, there are statistical differences in performance between each of the LSTM-based, RFM-based and the CNN-based models and the RF-Daily model. This emphasizes the importance of processing the daily behavior represented by a multivariate time series either by using a well-designed deep neural network or by extracting RFM features. While the Statistics-based model significantly outperforms the monthly models in terms of F1-Score, Lift and EMPC, it does not outperform the RF-Daily. The RF-Daily model significantly outperforms the Statistics-based model in terms of AUC and EMPC. Therefore, representing the time series by its statistical features is not an efficient way for daily churn modeling, and considering the raw features as a flattened vector is better than extracting their statistical features. Finally, our CNN-based model significantly outperforms the CNN-prev model proposed in Wangperawong et al. (2016) in terms of all metrics. This proves the effectiveness of our CNN design in which two parallel convolutional layers have been designed, one for extracting the weekly patterns in customer's behavior and the other for extracting the monthly ones. In conclusion, the dynamic daily behavior-based models always outperform the dynamic monthly ones, whether the features of the time series have been fed directly into a deep learning or a Random Forest model or have been aggregated well before. Furthermore, feeding the raw features into a well-designed LSTM or CNN model or aggregating them using the RFM features are better than feeding them directly into a Random Forest model. Moreover, feeding the raw daily features to a LSTM model is significantly better than feeding them to a CNN model.

While the previous analysis proves the merit of the proposed models in identifying churners more accurately than monthly models, it does not prove the early identification of churners. To do so, we compared the performance by day of the two best daily models versus the monthly models. More precisely, we predict churn using these models (RF-Monthly, LSTM-Monthly, RFM-based Model, and LSTM-based Model) in the following thirty-day prediction time window, and in each of these thirty days. These thirty days are following to the training days (from 29th July to 27th August in Fig. 6). The performance metrics were calculated for these models in each day. The results are shown in Fig. 8. It is clearly observed from the Fig. 8 that the performance of daily models is better than the performance of the monthly models in all days even when predictions are made further in the future.

In order to validate the quality of the learned representation visually, we used t-SNE to visualize the representation learned by our proposed models versus the raw representation (before passing to LSTM/CNN model or doing any transformation). While this visualization may not be helpful for model interpretation, it is helpful to validate visually the quality of the learned representation. t-Distributed Stochastic Neighbor Embedding (t-SNE) is an unsupervised, non-linear technique primarily used for exploring and visualizing high-dimensional data. In simpler terms, t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space. It was developed by Laurens van der Maatens and Geoffrey Hinton (Maaten & Hinton, 2008). The t-SNE algorithm calculates a similarity measure between pairs of instances in the high dimensional space and in the low dimensional space. It then tries to optimize these two similarity measures using a cost function. Accordingly, when we see two adjacent points in the

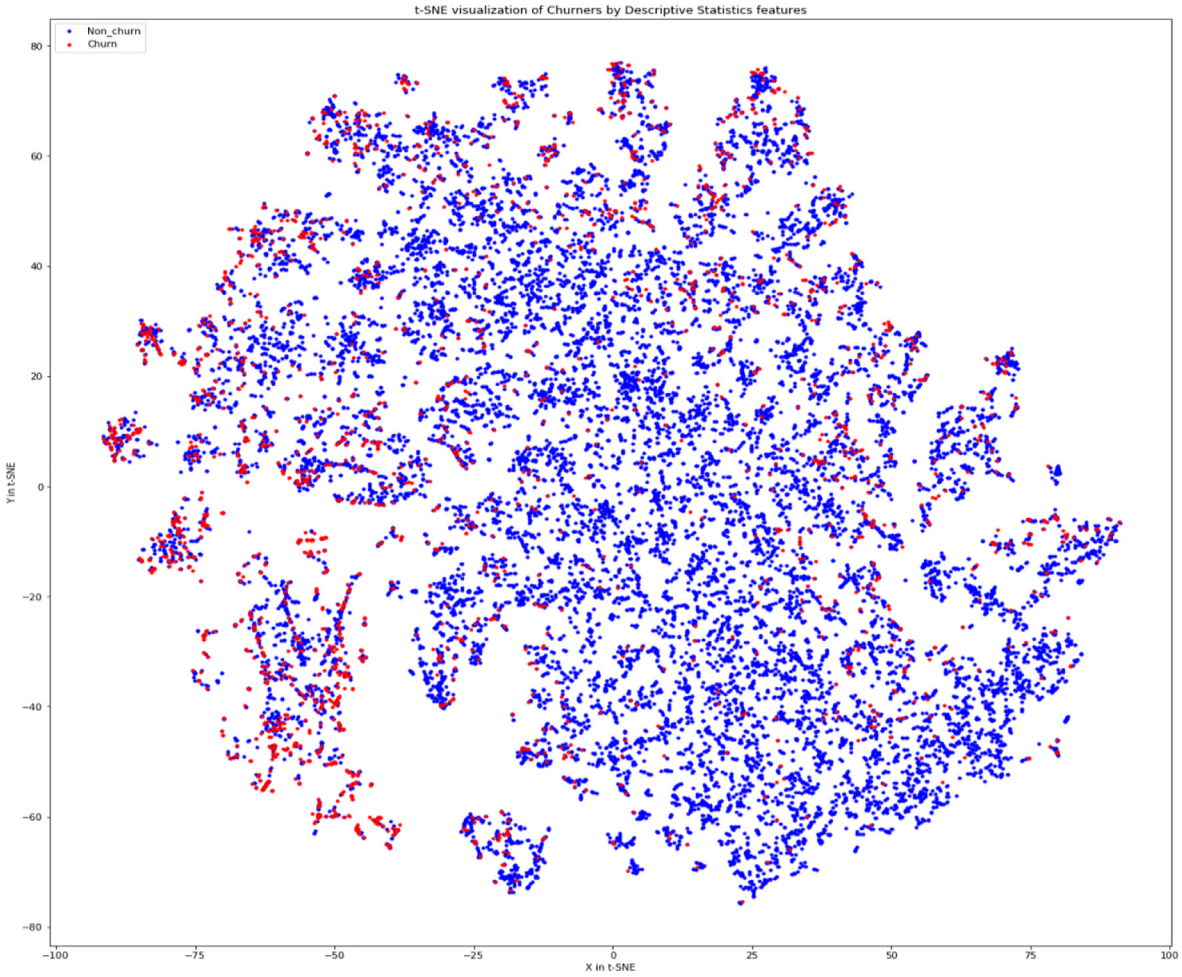

Fig. 13. t-SNE visualization of customers represented by the descriptive statistics features.

two-dimensional figure resulted from t-SNE algorithm we know that their representations in the multidimensional space are similar. Therefore, in this paper we utilized this algorithm to validate the quality the representation learned by the proposed model. We run this algorithm with its default parameters (perplexity = 30) on a random sample of 20 k customers from the validation dataset.

First, we visualized the raw representation of the customers by passing 20 K row vectors each has 900 values to the t-SNE algorithm. The output is shown in Fig. 9. For visualizing the learned representation by CNN and LSTM models, the output of the last hidden layer (fully connected layer with 128 neuron) in the CNN/LSTM model is passed to t-SNE, The corresponding outputs are shown in Figs. 10 and 11. The input passed to the t-SNE for visualizing the RFM-based/Statistics-based representation was the RFM feature vector of 120 values/the statistics feature vector of 360 values. The resulted output of the RFM representation and the statistics representation are shown in Figs. 12 and 13 respectively.

As Fig. 9 shows, the original distribution of customers has small sub groups that contains both types (churners and non-churners), in addition to two sub groups each contains one dominant type. CNN has somehow isolated some of churners in one area (the most left area of Fig. 10). RFM and Statistics representations is nonlinearly separable as Figs. 12 and 13 shows this ensures the need for nonlinear classifier to get good predictions. Whilst, LSTM has the clearest separation of churners and non-churners as Fig. 11 shows.

5. Conclusion

As discussed in the literature section, previous works have looked at churn as a static prediction problem and the studies that considered dynamic churn modeling focused on predicting churn at a monthly level. In this paper, our goal is to predict churn at a daily level based on the dynamic changes in customer's daily behavior. Therefore, we represented the customer's daily behavior by a multivariate time series and formulated the problem of daily churn prediction based on this representation. As a next step, we proposed two main approaches to predict churn at daily level using the multivariate time series. The first approach, namely the Feature-Based approach, suggests extracting meaningful features from the time series using our defined RFM functions (RFM-based model) or using some statistical functions (Statistics-based model), then feeding them to a traditional machine learning model, such as Random Forest, to make predictions. The second approach, namely the Deep Learning-Based approach, suggests designing deep learning models (CNN-based model and LSTM-based model) to learn the representative features from the multivariate time series and predict churn simultaneously.

In terms of the practical implication of this paper, our findings indicate that our proposed models are operationally efficient and they predict churners earlier and more accurately than the monthly models. This is very important, from business point of view, in order to improve the efficiency of retention marketing

campaigns. Furthermore, we found that the first contributor to the prediction accuracy is the frequency of the input where considering the daily behavior rather than monthly are better. The second contributor to the accuracy is using a well-designed LSTM/CNN model or using the RFM features of the time series with a Random Forest. Where, the predictive performance of these daily models (the LSTM-based, the CNN-based and the RFM-based models) are better than the predictive performance of the direct approach, in which the raw features of the time series are fed as a flattened vector into a Random Forest model (RF-Daily). However, statistical features of the time series does not contribute to the accuracy of daily churn prediction at all.

So far, we have only considered the behavioral features for the daily churn prediction. It would be useful, for future work, to consider another features such as the dynamic social features proposed in (Óskarsdóttir et al., 2018). Furthermore, there are two main issues that can be addressed in a future work, interpretability and retention effectiveness. As for interpretability, the proposed models lack causality, which targeting requires, so future work can be conducted to develop casual daily churn prediction model whose results are interpretable, and thus, enabling companies to target churning customers according to their causes of churn. This model might be attention-based LSTM. The second issue is related to the insufficiency of churn prediction output in terms of targeting, i.e. identifying customers at high risk of churning doesn't suffice to drive the company's targeting decisions because customers respond differently to retention interventions. Companies should not target those with the highest risk of churning, but rather those with the highest sensitivity to the intervention (Ascarza, 2018). To address this issue, future work can be done to extend the RFM-based and LSTM-based models by adding a dummy variable that denote customer's response to learn the corresponding churn uplift models. Furthermore, we can then compare the resulted uplift models with ones proposed in this paper using the maximum profit uplift (MPU) measure proposed in (Devriendt et al., 2019).

CRediT authorship contribution statement

Nadia Alboukaey: Conceptualization, Methodology, Software, Investigation, Writing - original draft. **Ammar Joukhadar:** Supervision. **Nada Ghneim:** Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Ahmad, A. K., Jafar, A., & Aljoumaa, K. (2019). Customer churn prediction in telecom using machine learning in big data platform. *Journal of Big Data, 6*(1). https://doi.org/10.1186/s40537-019-0191-6.

Ascarza, E. (2018). Retention futility: Targeting high-risk customers might be ineffective. *Journal of Marketing Research, 55*(1), 80–98. https://doi.org/10.1509/jmr.16.0163.

Buckinx, W., & Van den Poel, D. (2005). Customer base analysis: Partial defection of behaviourally loyal clients in a non-contractual FMCG retail setting. *European Journal of Operational Research, 164*(1), 252–268. https://doi.org/10.1016/j.ejor.2003.12.010.

Castanedo, F., Valverde, G., Zaratiegui, J., & Vazquez, A. (2014). Using Deep Learning to Predict Customer Churn in a Mobile Telecommunication Network Federico. 1–8.

Chen, Z. Y., Fan, Z. P., & Sun, M. (2012). A hierarchical multiple kernel support vector machine for customer churn prediction using longitudinal behavioral data. *European Journal of Operational Research, 223*(2), 461–472. https://doi.org/10.1016/j.ejor.2012.06.040.

Devriendt, F., Berrevoets, J., & Verbeke, W. (2019). Why you should stop predicting customer churn and start using uplift models. *Information Sciences*. https://doi.org/10.1016/j.ins.2019.12.075.

Eisinga, R., Heskes, T., Pelzer, B., & Te Grotenhuis, M. (2017). Exact p-values for pairwise comparison of Friedman rank sums, with application to comparing classifiers. *BMC Bioinformatics, 18*(1), 68. https://doi.org/10.1186/s12859-017-1486-2.

Gubela, R. M., Lessmann, S., & Jaroszewicz, S. (2020). Response transformation and profit decomposition for revenue uplift modeling. *European Journal of Operational Research, 283*(2), 647–661. https://doi.org/10.1016/j.ejor.2019.11.030.

Hadden, J., Tiwari, A., Roy, R., & Ruta, D. (2007). Computer assisted customer churn management: State-of-the-art and future trends. *Computers and Operations Research, 34*(10), 2902–2917 https://doi.org/10.1016/j.cor.2005.11.007.

Hochreiter, S., & Schmidhuber, J. (1997). LSTM can solve hard long time lag problems. *Advances in Neural Information Processing Systems*, 473–479.

Hu, J., Zhuang, Y., Yang, J., Lei, L., Huang, M., Zhu, R., & Dong, S. (2018). pRNN: A recurrent neural network based approach for customer churn prediction in telecommunication sector. *IEEE International Conference on Big Data (Big Data), 2018*, 4081–4085. https://doi.org/10.1109/BigData.2018.8622094.

Hung, S. Y., Yen, D. C., & Wang, H. Y. (2006). Applying data mining to telecom churn management. *Expert Systems with Applications, 31*(3), 515–524. https://doi.org/10.1016/j.eswa.2005.09.080.

Kane, K., Lo, V. S. Y., & Zheng, J. (2014). Mining for the truly responsive customers and prospects using true-lift modeling: Comparison of new and existing methods. *Journal of Marketing Analytics, 2*(4), 218–238. https://doi.org/10.1057/jma.2014.18.

Khan, F., & Kozat, S. S. (2017). Sequential churn prediction and analysis of cellular network users - A multi-class, multi-label perspective. 2017 25th Signal Processing and Communications Applications Conference, SIU 2017. https://doi.org/10.1109/SIU.2017.7960659

Khan, M. R., Manoj, J., Singh, A., & Blumenstock, J. (2015). Behavioral modeling for churn prediction: early indicators and accurate predictors of custom defection and loyalty. *Proceedings – 2015 IEEE International Congress on Big Data BigData Congress, 2015*, 677–680. https://doi.org/10.1109/BigDataCongress.2015.107.

van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE Laurens. *Journal of Machine Learning Research, 9*, 2579–2605.

Mitrović, S., Baesens, B., Lemahieu, W., & Weerdt, J. D. (2019). tcc2vec: RFM-informed representation learning on call graphs for churn prediction. *Information Sciences*. https://doi.org/10.1016/j.ins.2019.02.044.

Óskarsdóttir, M., Van Calster, T., Baesens, B., Lemahieu, W., & Vanthienen, J. (2018). Time series for early churn detection: Using similarity based classification for dynamic networks. *Expert Systems with Applications, 106*, 55–65. https://doi.org/10.1016/j.eswa.2018.04.003.

Phadke, C., Uzunalioglu, H., Mendiratta, V. B., Kushnir, D., & Doran, D. (2013). Prediction of subscriber churn using social network analysis. *Bell Labs Technical Journal, 17*(4), 63–75. https://doi.org/10.1002/bltj.21575.

Prashanth, R., Deepak, K., & Meher, A. K. (2017). High accuracy predictive modelling for customer churn prediction in telecom industry. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10358 LNAI, 391–402. https://doi.org/10.1007/978-3-319-62416-7_28

Sathe, S., & Aggarwal, C. C. (2017). Similarity forests. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Part F1296*, 395–403. https://doi.org/10.1145/3097983.3098046.

Vafeiadis, T., Diamantaras, K. I., Sarigiannidis, G., & Chatzisavvas, K. C. (2015). A comparison of machine learning techniques for customer churn prediction. *Simulation Modelling Practice and Theory, 55*, 1–9. https://doi.org/10.1016/j.simpat.2015.03.003.

Verbraken, T., Verbeke, W., & Baesens, B. (2013). A novel profit maximizing metric for measuring classification performance of customer churn prediction models. *IEEE Transactions on Knowledge and Data Engineering, 25*(5), 961–973. https://doi.org/10.1109/TKDE.2012.50.

Wangperawong, A., Brun, C., Laudy, O., & Pavasuthipaisit, R. (2016). Churn analysis using deep convolutional neural networks and autoencoders. http://arxiv.org/abs/1604.05377

Wei, C. P., & Chiu, I. T. (2002). Turning telecommunications call details to churn prediction: A data mining approach. *Expert Systems with Applications, 23*(2), 103–112. https://doi.org/10.1016/S0957-4174(02)00030-1.

Zaratiegui, J., Montoro, A., & Castanedo, F. (2015). Performing Highly Accurate Predictions Through Convolutional Networks for Actual Telecommunication Challenges. http://arxiv.org/abs/1511.04906

Zhang, J., Fu, J., Zhang, C., Ke, X., & Hu, Z. (2016). Not too late to identify potential churners: Early churn prediction in telecommunication industry. Proceedings - 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, BDCAT 2016, 194–199. https://doi.org/10.1145/3006299.3006324