Contents lists available at ScienceDirect

Expert Systems With Applications

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

ATM cash demand forecasting in an Indian bank with chaos and hybrid deep learning networks

Vangala Sarveswararao [a,b], Vadlamani Ravi [a,*], Yelleti Vivek [a]

[a] Center of Excellence in Analytics, Institute for Development and Research in Banking Technology, Castle Hills Road 1, Masab Tank, Hyderabad 500057, India
[b] School of Computer and Information Sciences, University of Hyderabad 500046, India

A R T I C L E I N F O

Keywords:
ATM Cash Demand Forecasting
Deep Learning
Chaos
1D-CNN
LSTM
GRNN
GMDH

A B S T R A C T

This paper proposes to model chaos in the automated teller machine (ATM) cash withdrawal time series of a large Indian commercial bank and forecast the withdrawals using deep learning (DL) and hybrid DL methods. It also considers the influence of "day-of-the-week" on the results. We first modelled the chaos present in the withdrawal time series by reconstructing the state space of each series using the optimal lag and embedding dimension. This process converts the original univariate time series into a multi variate time series. The "day-of-the-week" dummy variable is converted into seven variables using one-hot encoding and augmented to the multivariate or univariate time series depending on whether chaos was present or absent. For forecasting the future cash withdrawals, we employed (i) statistical technique namely autoregressive integrated moving average (ARIMA), (ii) machine learning techniques such as random forest (RF), support vector regression (SVR), multi-layer perceptron (MLP), group method of data handling (GMDH), and general regression neural network (GRNN), and (iii) DL techniques such as long short term memory (LSTM) neural network, Gated Recurrent Unit (GRU) and 1-dimensional convolutional neural network (1D-CNN). We also explored hybrid DL techniques such as 1D-CNN + LSTM and 1D-CNN + GRU. We observed improvements in the forecasts for all techniques when "day-of-the-week" variable was included. It is observed that chaos was present in 28 ATMs, whereas in the remaining 22 ATMs chaos was absent. In both the cases, LSTM yielded the best Symmetric Mean Absolute Percentage Error (SMAPE) on the test data. However, LSTM showed statistically different performance than the 1D-CNN + LSTM in chaos category but equal performance with 1D-CNN in non– category yet statistically significant than 1D-CNN.

1. Introduction

Financial forecasting usually involves prediction of time series such as stock market price, gold price, crude oil price, foreign exchange (FOREX) rate, interest rate and macroeconomic variables (eg. consumer price index). Phenomenal work has been reported in all these areas, where increasingly accurate models, both of stand-alone and hybrid types, were proposed. This area spans the statistical family of techniques such as simple linear regression, ridge regression, least absolute shrinkage and selection operator (LASSO), multi-variate adaptive regression splines (MARS), Autoregressive Regressive Integrated Moving Average (ARIMA) and logistic regression on one hand and machine learning family of techniques such as different neural networks architectures, support vector regression, decision trees on the other (Pradeepkumar & Ravi, 2018). Of late, deep learning neural network architectures have also been employed to solve these problems quite successfully. Many other types of time series also occur naturally while a bank performs its operations. Although not so well researched, one such problem pertains to offering services through ATM.

Accurate forecasting of cash withdrawals in ATMs is beneficial to both customers and banks. If banks load excess cash in ATMs than what is required, then they will lose out on the interest earned on the idle cash that would have otherwise been realized had that cash been invested elsewhere. On the other hand, if less than required cash is loaded in ATMs, then it results in ATMs going out of cash entailing severe customer dissatisfaction. Therefore, loading ATMs with optimal amount of cash is an important problem critical to the success of banking operations. Further, the frequency of replenishment of ATMs with cash is also an associated difficult problem. However, it turns out that ATM cash withdrawals forms a time series. If only accurate predictions are made to

* Corresponding author.
E-mail addresses: sarveswararao.cs@gmail.com (V. Sarveswararao), vravi@idrbt.ac.in (V. Ravi), vivek.yelleti@gmail.com (Y. Vivek).

https://doi.org/10.1016/j.eswa.2022.118645
Received 31 December 2021; Received in revised form 17 August 2022; Accepted 19 August 2022
Available online 23 August 2022

Table 1
Acronyms used in the current study.

Acronym	Fullform
DL	Deep Learning
ARIMA	Autoregressive Integrated Moving Average
RF	Random Forest
SVR	Support Vector Regressor
MLP	Multi-Layer Perceptron
GMDH	Group Method Data Handling
GRNN	General Regression Neural Network
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
1D-CNN	1D Convolutional Neural Network
SMAPE	Symmetric Mean Absolute Percentage Error
Dstat	Directional Symmetry Statistics
XGBoost	Extreme Gradient Boosting
ATM	Automated Teller Machine
MARS	Multi-variate adaptive regression splines
MLFF	Multi-layer feed forward neural network
WNN	Wavelet Neural Network
TISEAN	Time series Analysis
VAR-MAX	Vector Auto Regression with Exogenous Variable model
FOREX	Foreign Exchange
RNN	Recurrent Neural Networks
MISO	Multi input single output
Dstat	Directional Symmetry
LASSO	Least absolute shrinkage and selection operator
CNN	Convolutional neural network
IQR	Interquartile range
ANN	Artificial neural network

this time series, it will have far reaching ramifications to the cash dispensing, which among others, is traditionally treated as a yardstick to measure the overall success of the banking operations. Consequently, both issues (i.e. the amount of cash to be replenished and frequency of replenishment) mentioned above will be resolved in one go.

Identification and modeling of chaos present in a financial time series is of recent phenomenon (Ravi et al., 2017), (Pradeepkumar & Ravi, 2014), (Pradeepkumar & Ravi, 2016) and (Pradeepkumar & Ravi, 2017). As regards ATM cash withdrawals forecasting, there exists only one work (Venkatesh et al., 2014) that exploited the power of identification and modeling of chaos before embarking on forecasting with a host of neural networks. Further, deep learning paradigm has not yet been fully exploited to solve this problem.

This paper precisely fills that gap. The main contributions of this paper are (i) Imputing the missing/null values with the novel imputation technique where imputed with the median of the K number of previous corresponding weekday. The parameter K is defined by the user. (ii) first modeling the chaos present in the ATM withdrawals time series (iii) employing two popular deep learning architectures along with traditional machine learning techniques for forecasting purpose (iv) studying the effect of the *day-of-the-week* dummy variable as an exogenous dummy variable on the accuracy of the forecasts (v) for the first time, the study of forecasting the ATM cash withdrawals for the Indian Banks is conducted. This last aspect was studied earlier in Venkatesh et al. (2014). In this study, we worked on a dataset taken from a big, well-known Indian Commercial bank, hereafter referred to as XYZ bank. The dataset contains daily cash withdrawals of 50 ATMs for two consecutive years. In this paper, the word models and techniques are used interchangeably. All the symbols used in the current research study are presented in Table 1.

Because it is a time series, we can use forecasting methods ranging from Autoregressive Regressive Integrated Moving Average (ARIMA) to complex techniques from the Machine learning & Deep Learning family. We considered three classes of techniques such as (i) statistical family (ARIMA), (ii) ML family (SVM, RF, MLP, GMDH, GRNN), (iii) DL family (1D-CNN, LSTM), and hybrid DL techniques (1D-CNN + LSTM, 1D-CNN + GRU) in order to investigate the effectiveness of those methods to solve the forecasting problem.

It is documented in several studies (Pradeepkumar & Ravi, 2018) that ML methods and their hybrids (a.k.a soft computing methods) significantly outperformed the traditional statistical methods. Further, it is reported that deep learning methods (Huang, Chai, & Cho, 2020) and (Sezer, Gudelek, & Ozbayoglu, 2020) also outperform the traditional statistical methods. In view of the foregoing, our intention was to employ methods from all the three families and contrast their performance on the problem at hand. The advantages of the above-mentioned models in the current study are discussed as follows:

- Among the above mentioned models, it is to be noted that we employed three popular deep learning architectures namely, 1dimensional-convolutional neural network (1D-CNN), long short term memory (LSTM), and gated recurrent unit (GRU) in pairwise combinations. These methods are chosen because of their demonstrated capability in solving time series forecasting problems (Jiang, 2021). Both GRU and LSTM (Sercan & Ugur, 2017, Ozbayoglu, Gudelek, & Sezer, 2020) capture the long-term dependencies in a given time series. CNN, (Liu, Zhang, & Ma, 2017, Ozbayoglu et al., 2020) by virtue of its design, yields latent variables, a feature which is immensely useful in extracting latent variables in a given time series. In view of the foregoing, these three deep learning architectures are chosen in our study.
- Further, ARIMA is chosen as a baseline because it captures the autoregressive and moving average portions of a given time series.
- MLP (Jiang, 2021, Pradeepkumar et al., 2020) is chosen because (i) it is a universal approximator. (ii) it has solved many time series forecasting problems more efficiently than the statistical methods. (iii) it captures the deterministic trend, underlying non-linearities and randomness in a given time series.
- SVR (Kavitha, Varuna, & Ramya, 2016; Pradeepkumar & Ravi, 2018; Sengupta et al., 2020), chosen because it provides a global optimal solution by virtue of its design (it collapses into a quadratic programming problem). SVR proven to be solving various classification and regression problems (Kavitha et al., 2016).
- GRNN (Pradeepkumar et al., 2020) is chosen because it performs non-parametric regression using Parzen window approach and is demonstrated to be very good at time series forecasting.
- GMDH (Ivakhnenko, 1966) is chosen because it is world's first deep learning architecture having self-organizing property (Pradeepkumar et al., 2020).
- RF being an ensemble of several decision trees provides very accurate predictions for classification (Gao & Liu, 2021) and regression problems (Manoj & Kumar, 2018).

The complete details of the aforementioned algorithms were discussed in detail in Section 3.4. Further, the reasons for not selecting a few of the other popular effective models are as follows:

- Exponential smoothing cannot be applied because it cannot capture the underlying non-linearities and randomness in the data. Therefore, Box-Jenkins methodology (ARIMA) was proposed in literature to address these issues (Shumway & Stoffer, 2017). The methods chosen in the current study outperformed ARIMA models in all ATMs.
- ELM, (Guang-Bin et al., 2006) consisting of a random set of weights between input and hidden layers and linear regression between hidden and output layers, is too simplistic to capture underlying non-linearities and randomness in the data. Further, ELM, by design, does not allow weight updation between input and hidden layers. It is for these reasons, we did not apply ELM in the current study.
- As regards DBN, it is an unsupervised network and hence cannot be used for forecasting purposes.

The popularity of this problem can be assessed by the fact that in 2005, Lancaster University conducted NN5 competition (Crone, 2008), which focused on forecasting daily cash withdrawals for 111 ATMs across the UK, and the participants proposed various methods to forecast cash withdrawals accurately.

The rest of the paper is organized as follows: Section 2 presents the literature review; Section 3 presents in detail our proposed model; Section 4 presents the dataset description and evaluation metrics; Section 5 presents a discussion of the results and finally Section 6 concludes the paper.

2. Literature survey

We first review the research works conducted on the dataset of NN5 Competition (Crone, 2008) as this competition is very much related to our work. The dataset in this competition contains two years of daily cash withdrawals for 111 ATMs in the UK. The task is to forecast ATM cash withdrawals for each ATM for the next 56 days. The best performing model from Andrawis, Atiya, and El-Shishiny (2011) is an ensemble of General Gaussian Regression, Neural network, and linear models and achieved a SMAPE of 18.95 %. Venkatesh et al. (2014) improved the forecast accuracy reported by Andrawis et al. (2011) by clustering similar ATMs and applying a host of popular neural networks. In another work, Venkatesh et al. (2014) further improved the results by modelling chaos in the cash withdrawal time series and invoking the popular neural networks architectures. None of the studies included the effect of exogenous dummy variable such as 'day_of_week' dummy variable and whether it is weekday or weekend as these features have a significant effect on the forecast accuracy.

Venkatesh et al. (2014) clustered the ATMs with the similar day of the week cash demand patterns and applied four neural networks namely General regression neural network (GRNN)) (Specht, 1991), multi-layer feed-forward neural network (MLFF) (Rumelhart, Hinton, & Williams, 1986), group method of data handling (GMDH), wavelet neural network (WNN) (Walter, Miao, Zhang, & Wayne Lee, 1995) and Auto-Regressive Integrated Moving Average (ARIMA). Out of their four networks, GRNN yielded an SMAPE of 18.44 %, which is better than the result of Andrawis et al. (2011). Venkatesh et al. (2014) found that chaos was present in the cash withdrawals data of NN5 Competition. In order to find and model chaos, they employed the time series analysis (TISEAN) tool for calculating the optimal lag and embedding dimension of each series. They employed a chaotic approach to the forecasting and achieved a SMAPE of 14.71 %, which was better than the results of Venkatesh et al. (2014). Later, Javanmard (2016) employed fuzzy logic to forecast cash demand, and did not compare the proposed method with ML methods. Bhandari and Gill (2016) proposed a hybrid of NN using Genetic Algorithm to predict cash demand and it outperformed NN with gradient descent. They did not mention the data used or comparison with other methods. Jadwal, Jain, Gupta, and Khanna (2017) proposed a clustering method for ATMs but it is more or less, same as that of Venkatesh et al. (2014). Arabani and Komleh (2019) proposed convolutional neural network (CNN) while Rafi, Wahab, Khan, and Raza (2020) employed Vector Auto Regression with Exogenous Variable model (VAR-MAX).

The use of machine learning and deep learning techniques for forecasting time series became a common practice nowadays as they tend to outperform the traditional statistical techniques and can learn intricate patterns in the series. Pradeepkumar and Ravi (2014; 2016) improved the FOREX rate predictions by modelling chaos before applying the hybrid of MLP + Evolutionary algorithms, Quantile regression random forest (Pradeepkumar & Ravi, 2016), and Multivariate Regression Splines (Pradeepkumar & Ravi, 2017). Recurrent neural networks (RNNs) are commonly used deep learning algorithms for sequence prediction and they have gained popularity in solving complex sequential problems such as natural language understanding, speech recognition, and language translation. However, unlike MLP, the feedback loop of the recurrent cells addresses the temporal order of the sequences (Schäfer & Zimmermann, 2006). The commonly used RNN cells used for sequence prediction are Elman RNN cell, Gated Recurrent Unit (GRU) (Cho et al., 2014), and Long Short-term Memory cell (Hochreiter & Schmidhuber, 1997). Because of the vanishing gradient problem in the RNN cell, LSTM is preferred for modelling sequence problems. Authors of (Tian & Pan, 2015), (Duan, Lv, & Wang, 2016) and (Lee, Xie, Gallagher, Zhang, & Tu, 2015) successfully applied LSTMs to time-series forecasting applications. LeCun and Bengio (1998) proposed the 1-dimensional convolutional neural networks (ID-CNN) for time series forecasting, but they are inefficient in capturing long term dependency in the time series. Papadopoulos (2018) proposed three layers dilated causal convolutions to time series forecasting based the WaveNet architecture for audio waveforms (van den Oord, Dieleman, Zen, Simonyan, Vinyals, Graves, Kalchbrenner, Senior, & Kavukcuoglu, 2016), which resulted in better performance compared to LSTM implementation. All these works on ATM cash withdrawals forecasting did not consider using exogenous dummy variables.

Our current proposed work is different from that of Venkatesh, Ravi, and Kumar (2014) and Venkatesh, Ravi, Prinzie, and Van Den Poel (2014) in the following ways:

- Venkatesh, Ravi, and Kumar (2014) clustered ATMs based on the daily withdrawal patterns and then applied four different forecasting methods to each cluster of ATMs. Effectively, that means training separate techniques for each one of the clusters. For example, GRNN model for cluster1 has different sets of hyperparameters compared to the GRNN model for cluster2. However, in our present study, we did not cluster ATMs into groups. Further, we modelled chaos present in the time series and included the day-of-the-week as a dummy feature, whereas chaos was not modelled at all in Venkatesh, Ravi, and Kumar (2014).
- Interestingly, while chaos was modelled in Venkatesh, Ravi, Prinzie, and Van Den Poel (2014), the day-of-the-week exogenous dummy variable was not included at all.
- More importantly, while we employed deep learning methods and their hybrids, neither of Venkatesh, Ravi, and Kumar (2014) and Venkatesh, Ravi, Prinzie, and Van Den Poel (2014) employed them.
- Our work analyzed the data related to a large commercial bank in India, while Venkatesh, Ravi, and Kumar (2014) and Venkatesh, Ravi, Prinzie, and Van Den Poel (2014) analyzed NN5 competition data.

3. Overview of the techniques used

3.1. Chaos theory

Poincare proposed the theory of chaos during the 1800s, and later it was extended by Lorenz (1963) in order to deal with complex non-linear systems (Dhanya & Nagesh Kumar, 2010). A chaotic system is dynamic, deterministic and evolves from initial conditions, and trajectories can describe the system in the state space. The governing equations for a chaotic system are known ahead, so the state space is represented by phase space, which we can reconstruct from original series using delay time and embedding dimension (Packard, Crutchfield, Farmer, & Shaw, 1980) as shown in Eq. (1):

$$Y_i = (x_i, x_{i+\tau}, x_{i+2\tau}, \cdots, x_{i+(m-1)\tau}) \tag{1}$$

where τ is the time delay or the lag of the system, m is the embedding dimension to reconstruct the phase space and Y_i is the i^{th} reconstructed vector. After reconstructing the phase space, the time series problem turns out to be a multi-input single-output (MISO) prediction problem, which can be modelled by methods ranging from linear models to deep neural networks.

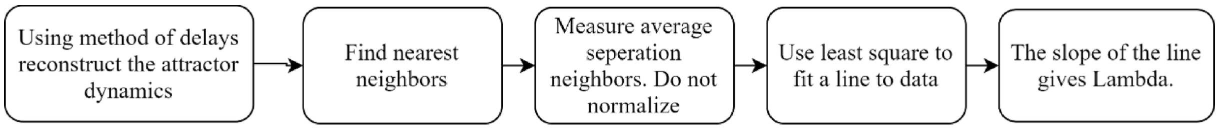
Using method of delays reconstruct the attractor dynamics
Find nearest neighbors
Measure average seperation neighbors. Do not normalize
Use least square to fit a line to data
The slope of the line gives Lambda.

Fig. 1. Rosenstein's method to calculate λ.

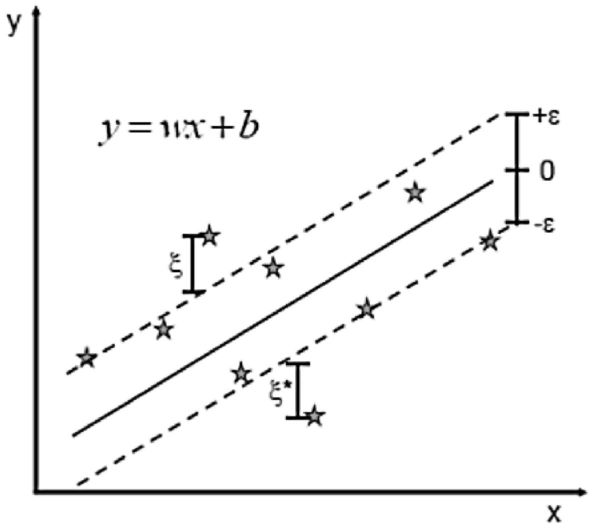

Fig. 2. Architecture of SVR.

3.2. *Rosenstein's method*

Rosenstein's algorithm (Rosenstein, Collins, & De Luca, 1993) estimates the largest Lyapunov Exponent *(λ)* (Lyapunov, 1907) from the given time series as shown as in Fig. 1. If $\lambda \geq 0$ then chaos is present; otherwise chaos is absent in given time series. Fig. 1 depicts the method.

3.3. *Cao's method*

Cao proposed a method (1997) to find out the minimum embedding dimension for a given time series. Let $X = (x_1, x_2, x_3, x_4, \ldots, x_N)$ be a time series of N points. In the phase space, the time series can be reconstructed as time delay vectors as given in Eq. (1).

3.4. *Methods employed in this study*

3.4.1. *Autoregressive integrated moving average (ARIMA)*

In time series analysis, non-seasonal ARIMA techniques are usually denoted as ARIMA (*p, d, q*) where *p* is the autoregressive model order, *d* is the degree of differencing and *q* is the moving-average model order. These techniques are fitted to the given time series data either to predict future points in the series or to understand the series better. ARIMA techniques are applied when the series shows non-stationarity, where differencing step (*d*) is applied to make the series stationary. The forecasting equation is constructed as follows:

Let *y* denote d^{th} difference of given time series *Y*.

If $d = 0$, then $y_t = Y_t$. If $d = 1$, then $y_t = Y_t - Y_{t-1}$. If $d = 2$, then $y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2})$ etc.

Then the ARMA (*p,q*) model is written as follows as presented in Eq. (2).

$$\widehat{y}_t = c + \varphi_1 y_{t-1} + \cdots + \varphi_1 y_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_1 e_{t-q} \tag{2}$$

where $\widehat{y}_t$ denotes the d^{th} difference of given time series *Y*, φ's are autoregressive parameters, θ's are moving average parameters and c is the intercept.

3.4.2. *Support vector regression Machines (SVR)*

Drucker, Burges, Kaufman, Smola, and Vapnik (1997) proposed a version of Support Vector Machines (Cortes & Vapnik, 1995) to deal with regression problems and the training phase of SVR involves minimizing the Eq.(3) while following the constraints as given in Eq.(4). The architecture of the SVR is depicted in the Fig. 2.

$$minimize : \frac{1}{2}*||w||^2 + C*\sum_{i=1}^{N}(\beta_i + \beta_i^*) \tag{3}$$

$$\begin{aligned} y_i - w^*x_i - b &< \epsilon + \beta_i \\ w^*x_i + b - y_i &< \epsilon + \beta_i^* \\ \beta_i, \beta_i^* &> 0 \end{aligned} \tag{4}$$

where w = weight vector, N = the number of training examples, y_i = target feature, β_i = bias, β_i^* = slack variable, C = regularization constant and ϵ = margin of tolerance.

3.4.3. *Random forest (RF)*

Random forests proposed by Ho (1995) is an ensemble method for both classification and regression problems. RF works by constructing a multitude of decision trees during the training phase and outputs the target value by calculating the mean of the individual decision tree outputs.

3.4.4. *XGBoost*

Extreme Gradient Boosting (XGBoost) proposed by Chen and Guestrin (2016) is a decision tree based ensemble algorithm that has been dominating Kaggle competitions and applied machine learning for tabular data. XGBoost is a robust implementation of gradient boosting trees (Friedman, 2001), which are designed for performance and speed. Boosting is an ensemble method to construct a strong model using many weak techniques by adding techniques on top of each other, i.e. the mistakes or errors made by previous techniques are corrected by next predictor, until the training set is correctly predicted.

3.4.5. *Multi-layer perceptron (MLP)*

Multilayer perceptron is a feed forward artificial neural network that is trained by backpropagation algorithm. MLP consists of 3 layers namely an input layer, a hidden layer, and an output layer. Hidden and the output layers have sigmoid or logistic function as the non-linear activation function. It is too popular architecture to be discussed here in greater detail.

3.4.6. *Group method of data handling (GMDH)*

GMDH (Ivakhnenko, 1966) is a self-organised feed-forward network based on polynomial transfer functions, called Ivakhnenko polynomials. The coefficients of these functions are determined using least square regression. It is the best suitable artificial neural network (ANN) for dealing with inaccurate, noisy, or small datasets. It can be used to solve classification & regression problems and solve the problem of overfitting. It is the earliest proposed deep learning neural network architecture, without using the present day jargon, where nodes in the hidden layers are dropped if they are found to be not having sufficient predictive power. The GMDH algorithm builds regression equations of high order (e.g., 2 or 3) for each pair of input variables x_i and x_j (e.g., $y = a + bx_i + cx_j + dx_i^2 + ex_j^2 + fx_ix_j$). This is called an Ivakhnenko polynomial. The best input variables (polynomials) selected as per pre-specified selection criteria are retained and the next layer accepts these best

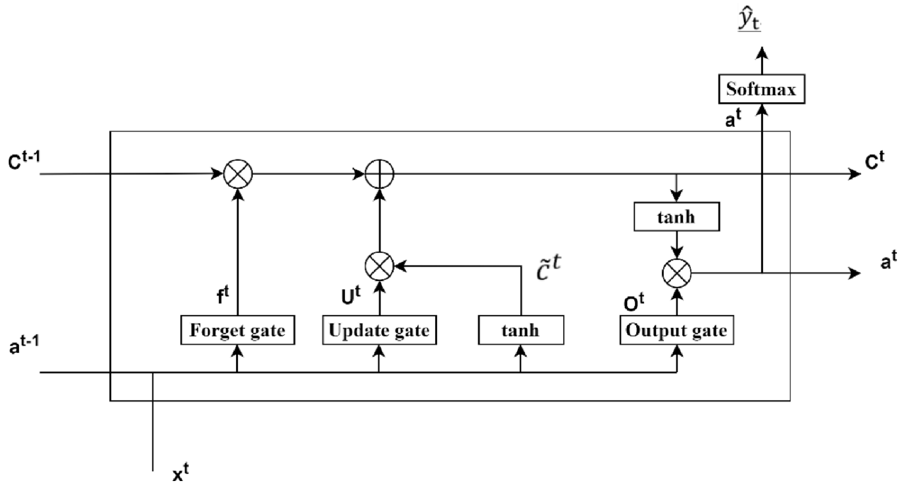

Fig. 3. Architecture of LSTM Unit.

variables to generate new input variables of higher-order (order of 4 if started with order 2). This process is continued as long as the prediction model does not satisfy the pre-specified error tolerance.

3.4.7. General regression neural network (GRNN)

Generalized regression neural network (GRNN) proposed by Specht (1991) is a feed forward neural network having roots in statistics. GRNN represents an advanced architecture in the neural networks that implements non-parametric regression with one-pass training. The topology of GRNN involves four layers of neurons, viz., input, pattern, summation, and the output in that order. The pattern layer comprises '*n*' training neurons. The test sample is fed to the input layer, which has all the variables, while the pattern layer consists of all the training samples. The distance, d_i, between the training sample present as a neuron in the pattern layer and the data point from the test set used for regression, is used to figure out how well each training neuron in pattern layer can represent the feature space of the test sample, X. This probability density is calculated by the Gaussian activation function. Thus, the summation of the product of target value and the result of activation function for each neuron i.e., $\sum_{i=1}^{n} Y_i * e^{\left(\frac{-d_i^2}{2\sigma^2}\right)}$ forms the numerator term in the summation layer and the summation of the term $e^{\left(\frac{-d_i^2}{2\sigma^2}\right)}$ i.e., $\sum_{i=1}^{n} e^{\left(\frac{-d_i^2}{2\sigma^2}\right)}$ forms the denominator term in the summation layer. Thus, the summation layer has two neurons. Then, in the output layer containing one neuron, the prediction is calculated as $\widehat{Y}(X) = \frac{\sum_{i=1}^{n} Y_i * e^{\left(\frac{-d_i^2}{2\sigma^2}\right)}}{\sum_{i=1}^{n} e^{\left(\frac{-d_i^2}{2\sigma^2}\right)}}$. Here $d_i^2 = (X - X_i)^T * (X - X_i)$. X is a test sample and X_i is the neuron present in the pattern layer.

3.4.8. Long short term memory neural network

Long short term memory (LSTM) is a variation of recurrent neural network proposed by Hochreiter and Schmidhuber (1997) that is trained using back propagation through time and overcomes vanishing gradient problem. So, it can be used to construct a large network to deal with difficult sequence problems mostly in natural language processing. But, LSTM's can be used as a forecasting algorithm as time series is a sequence of data points recorded over time. Instead of a neuron, LSTM's

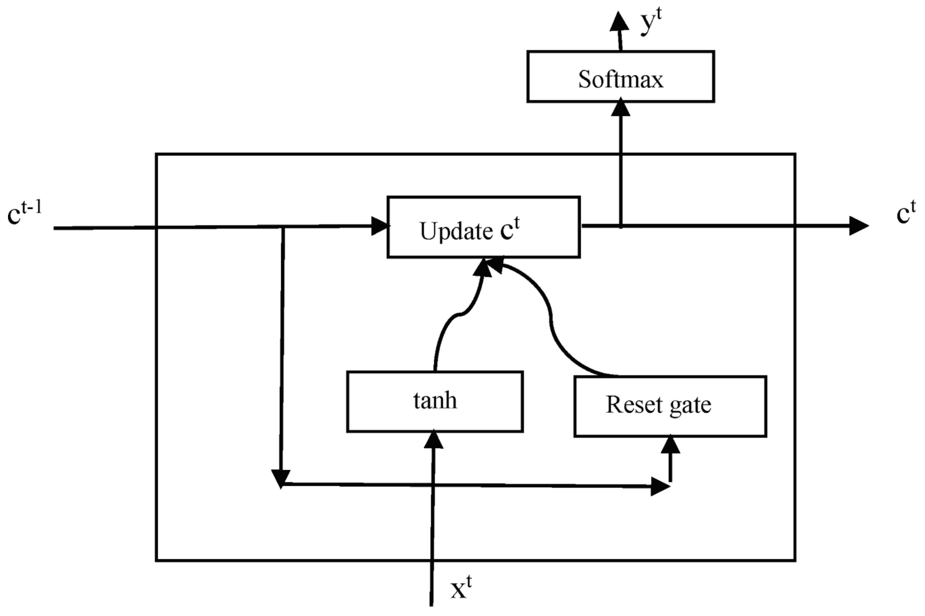

Fig. 4. Architecture of GRU Unit.

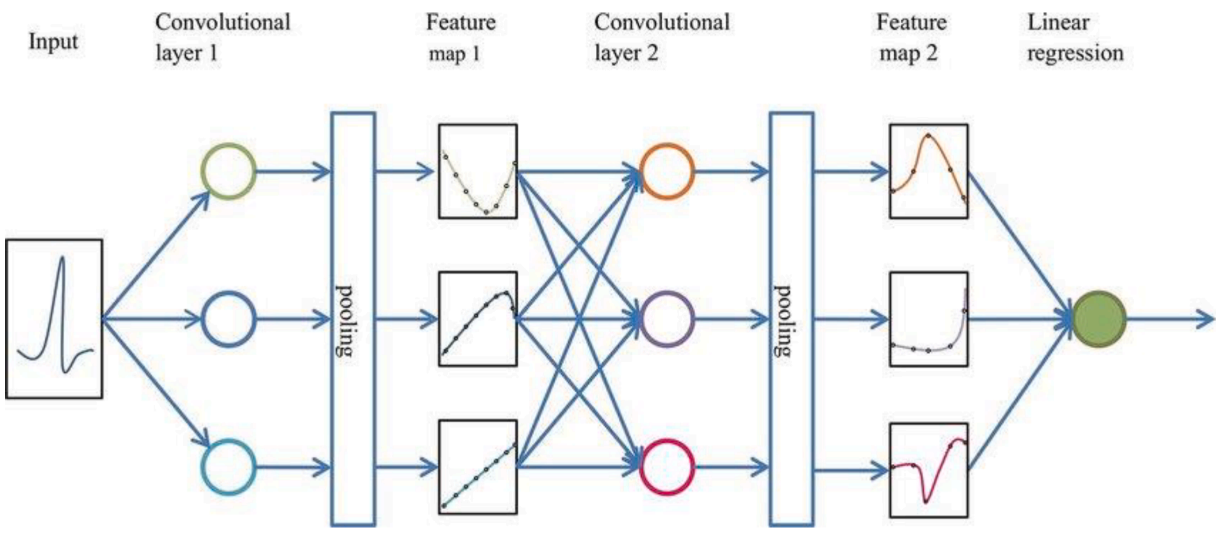

Fig. 5. Architecture of 1D-CNN.

contains memory block as nodes which are connected through layers. Each memory block has three types of gates in it. i) Forget Gate, which decides what information to throw away from the memory block, ii) Input Gate, that checks which values from the input to be used to update the memory block, iii) Output Gate, that decides what to output based on the input and memory of the block. Fig. 3 depicts the architecture of an LSTM unit. Input to the each LSTM block is processed sequentially by following the Eq. (5) to Eq. (10).

$$\widetilde{c}^t = tanh\left(W_c\left[a^{t-1}, x^t\right] + b_c\right) \tag{5}$$

$$UpdateGate : \Gamma_u = \sigma\left(W_u\left[a^{t-1}, x^t\right] + b_u\right) \tag{6}$$

$$ForgetGate : \Gamma_f = \sigma\left(W_f\left[a^{t-1}, x^t\right] + b_f\right) \tag{7}$$

$$OutputGate : \Gamma_o = \sigma\left(W_o\left[a^{t-1}, x^t\right] + b_o\right) \tag{8}$$

$$c^t = \Gamma_u * \widetilde{c}^t + \Gamma_f * c^{t-1} \tag{9}$$

$$a^t = \Gamma_o * tanhc^t \tag{10}$$

where $b_c, b_u, b_f, andb_o$ are biases for cell, update gate, forget gate and output gate respectively. (a^{t-1}, a^t) are previous cell and current activation values respectively. (W_f, W_o, W_u) are weights for forget, output and update gates respectively. $W_u[a^{t-1}, x^t]$ is the dot product between the weights of update gate and the concatenation of the previous cell a^{t-1} and current input x^t. Similarly, $W_f[a^{t-1}, x^t]$ and $W_o\left[a^{t-1}, x^t\right]$ is the dot product between the forget gate and output gate with the concatenation of previous cell a^{t-1} and current input x^t. (c^{t-1}, c^t) are previous and current cell memory values respectively ,$\widetilde{c}^t$ is temporary cell memory and $\widetilde{c}^t$ is temporary cell memory.

3.4.9. Gated Recurrent Unit neural network (GRU)

Gated Recurrent Unit (GRU) is another advancement variation of recurrent neural network introduced by Cho et al. (2014). It doesn't have any cell state c^t but it only have the hidden state h^t. This makes it faster to train when compared to LSTM and also offers the comparable

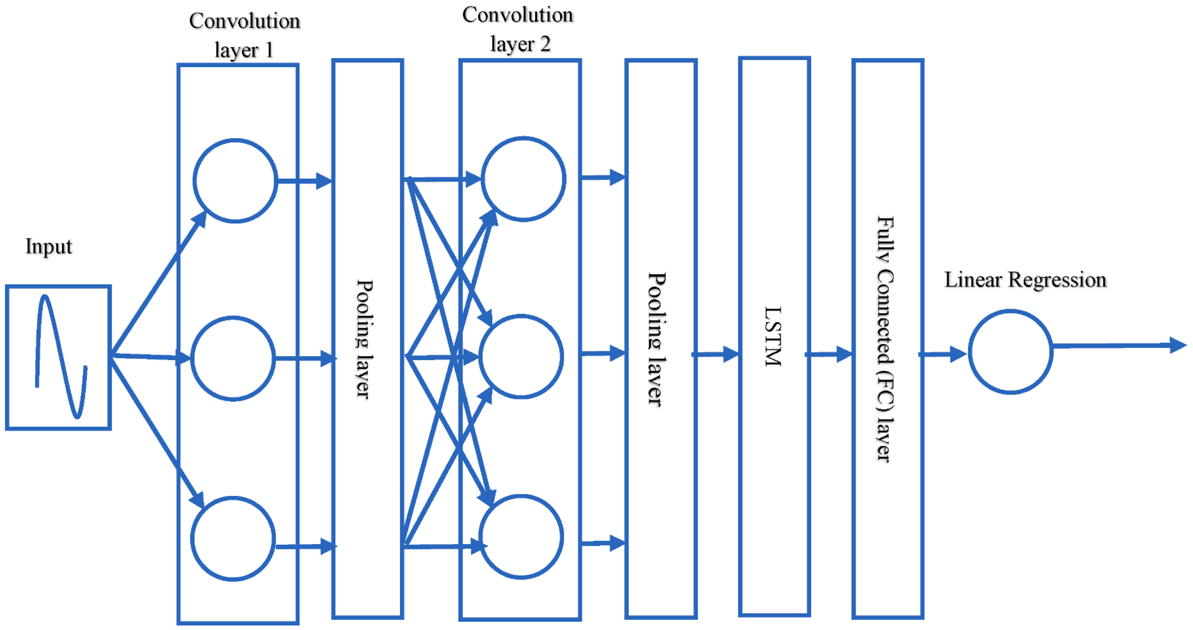

Fig. 6. Architecture of 1D CNN + LSTM.

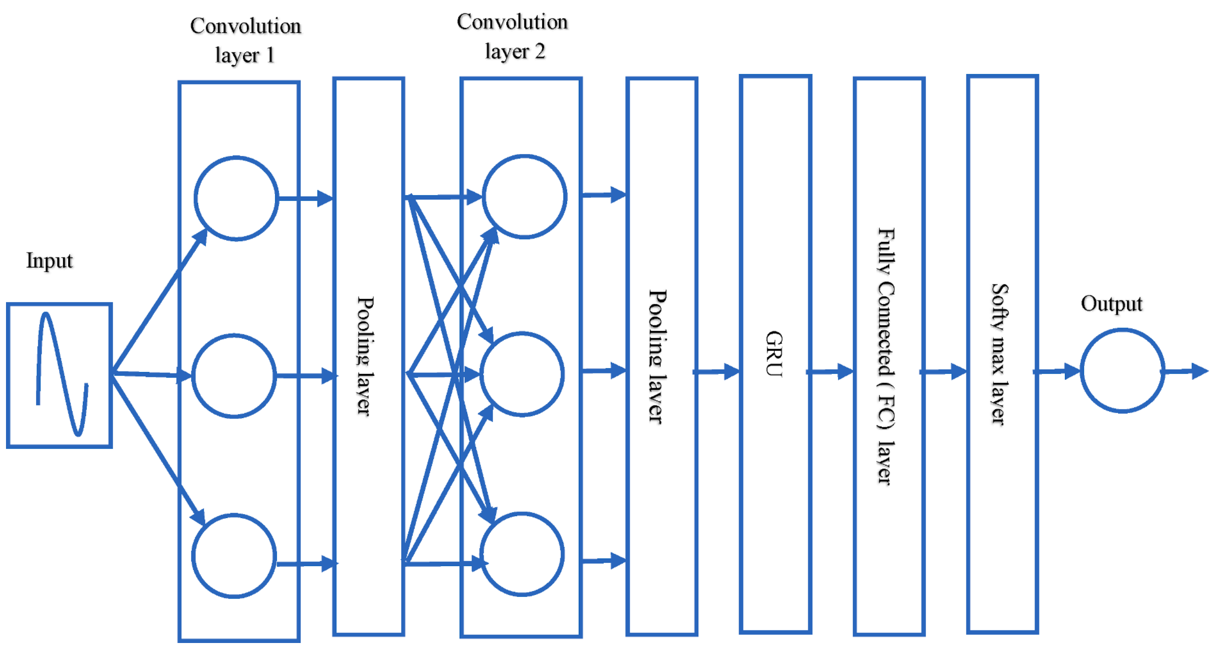

Fig. 7. Architecture of 1D CNN + GRU.

performance. Due to this similarity to LSTM, GRU can also be used in natural language processing and also as a forecasting algorithm. GRU's also contains memory block as nodes which are connected through layers. Each memory block has three types of gates in it. i)Update Gate (ii) Reset Gate. The architecture of GRU is depicted in Fig. 4. Input to the each GRU block is processed sequentially by following the Eq.(11) to Eq. (14).

$$\tilde{c}^t = tanh\left(W_c\left[c^{t-1}, x^t\right] + b_c\right) \tag{11}$$

$$ResetGate : \Gamma_u = \sigma\left(W_u\left[c^{t-1}, x^t\right] + b_u\right) \tag{12}$$

$$c^t = \Gamma_u * \tilde{c}^t + (1 - \Gamma_u) * c^{t-1} \tag{13}$$

$$a^t = \Gamma_o * tanhc^t \tag{14}$$

where b_c, b_u are biases for cell and update gate respectively. (a^{t-1}, a^t) are previous cell and current activation values respectively. (W_c, W_u) are

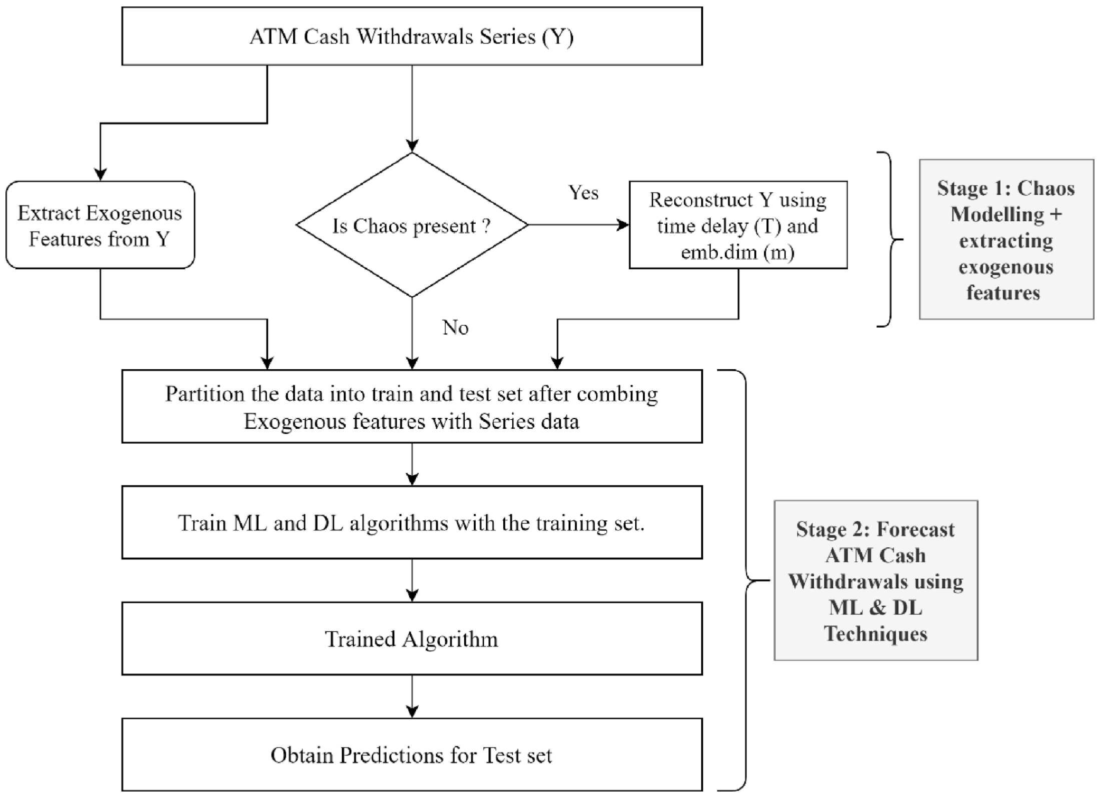
ATM Cash Withdrawals Series (Y)
Extract Exogenous Features from Y
Is Chaos present ?
Yes
Reconstruct Y using time delay (T) and emb.dim (m)
No
Stage 1: Chaos Modelling + extracting exogenous features
Partition the data into train and test set after combing Exogenous features with Series data
Train ML and DL algorithms with the training set.
Trained Algorithm
Obtain Predictions for Test set
Stage 2: Forecast ATM Cash Withdrawals using ML & DL Techniques

Fig. 8. Schematic of the proposed methodology.

weights for reset, update gates respectively. (c^{t-1}, c^t) are previous and current cell memory values respectively and $\widetilde{c}^t$ is temporary cell memory. $W_C[c^{t-1}, x^t]$ is the dot product between the weight of cell and the concatenation of the previous cell memory c^{t-1} and current input x^t. Similarly $W_u[c^{t-1}, x^t]$ is the dot product between the weight of update gate and the concatenation of the previous cell memory c^{t-1} and current input x^t.

3.4.10. One dimensional convolutional neural network (1D-CNN)

Convolutional Neural Network (CNN) was proposed by LeCun and Bengio (1998) for robust image classification. CNN techniques take images as two-dimensional input by representing each image as its pixel values and colour channels and output its corresponding predicted class. Each CNN layer from left to right learns most basic to complex features of the image, and the final layer takes complex features and outputs a target class. The one-dimensional CNN (1D-CNN) also follows the same procedure as 2D CNN but learns features from sequences of data (eg. Time Series). 1D-CNN extracts features from the time-series data, and it uses them to forecast future values. CNN works the same way for any number of dimensions except the structure input data and filters vary. Fig. 5 depicts the architecture of the 1D-CNN.

3.4.11. One dimensional convolutional neural Network + Long term short memory neural network (1D-CNN + LSTM)

As explained in the previous section both the 1D-CNN and LSTM has occupied its importance in various kinds of Classification and Regression problems. In order to leverage the advantages of both the 1D-CNN and LSTM, a Hybrid DL model is designed and implemented. The architecture of the 1D-CNN and LSTM is depicted in the Fig. 6. The architecture says that initially the Input data is given to the 1D-CNN model to extract the features. 1D-CNN which we used here also follows the same procedure (refer 3.4.10) to extract the most basic to complex features. Thus extracted features are given as input to the LSTM model, (refer 3.4.8) and the respective architecture is depicted in Fig. 3. This LSTM is added to extract the features further where the higher order complex features can also be extracted. Thus extracted features are given to the output layer to forecast the future values.

3.4.12. One dimensional convolutional neural Network + Gated Recurrent Unit neural network (1D-CNN + GRU)

As explained in the previous section both the 1D-CNN and GRU has occupied its importance in various kinds of Classification and Regression problems. In order to leverage the advantages of both the 1D-CNN and GRU, a Hybrid DL model is designed and implemented. The architecture of the 1D-CNN and GRU is depicted in the Fig. 7. The architecture says that initially the Input data is given to the 1D-CNN model to extract the features. 1D-CNN which we used here also follows the same procedure (refer 3.4.10) to extract the most basic to complex features. Thus extracted features are given as input to the GRU model, (refer 3.4.9) and the respective architecture is depicted in Fig. 4. This LSTM is added to extract the features further where the higher order complex features can also be extracted. Thus extracted features are given to the output layer to forecast the future values.

4. Proposed methodology

The proposed methodology (see Fig. 8) first extracts the exogenous dummy variable such as a week of the day, from the date index of each series and then appends this additional information with the series data to forecast the predictions. We applied One-Hot-Encoding to the week of the day feature, which resulted in a total of 7 exogenous dummy variables. We considered this extra information along with the series data while training machine learning and deep learning techniques.

4.1. Imputation technique

Several ATMs were observed to have missing/null values because of a cooling period where the ATM is under maintenance for some continuous period of time. This cooling period is observed to happened for some continuous period of time thereby saying a complete dip is referred to as a dip-period, as there are no transactions are happened in this particular time. Hence instead of going for imputing with median where the entire dip-period is imputed with a median value thereby converting the dip-period to a flat-period. This shows that the still the problem is unresolved and this stands as a barrier to mimic the original transaction trend. Hence, a novel imputation technique is proposed to avoid these flat-period as follows:

Let $Y = (y_1, y_2, y_3, .., y_N)$ be cash withdrawal series for a single ATM and the future cash demand is predicted using the following two-stage modelling. In this imputation technique, the missing value/null value is replaced by the median of K previous values corresponding to the same weekday. Here the value K is predefined by the user. Hence, suppose a Monday weekday value is observed to have a missing value, then it is replaced with the median of the withdrawals values of K previous Monday. Similar approach is followed for other days also. With this the flat-period is replaced with a trend which mimics the original trend or near-original trend. The mathematical representation of the imputation technique is presented in Eq.(15).

$$imputed_{value} = Median(y_{t-7}, y_{t-7*2}, .., y_{t-7*K}) \tag{15}$$

where, $imputed_{value}$ is the value that replaces the missing value and y_{t-7}, $y_{t-7*2}, .., y_{t-7*K}$ are the K values corresponding to the same weekday.

4.2. Description of the two-stage model

Stage 1: Chaos modelling

Reconstructing the phase space: Check series Y for the presence of chaos with the help of Lyapunov exponent value. If chaos is present, then reconstruct the phase space using lag/delay time (τ) & embedding dimension (m) found using autocorrelation function and Cao's method respectively. Partition the Y into $Y_{Train} = \{y_t; t = \tau m + 1, \tau m + 2, \cdots, k\}$ and $Y_{Test} = \{y_t; t = k + 1, k + 2, ..., N\}$, where N is the number of data points in the time series. The mathematical formulation of the chaotic time series along with exogenous dummy variables (refer to Eq. (16)).

$$\widehat{Y_t} = f(y_{t-\tau}, y_{t-2\tau}, y_{t-3\tau}, \cdots, y_{t-\tau m}, y_{is\_monday}, y_{is\_teusday}, y_{is\_wednesday}, y_{is\_thrusday}, y_{is\_friday}, y_{is\_saturday}, y_{is\_sunday}) \tag{16}$$

where $\widehat{Y}$ is forecasted value at time t, lag/delay time (τ), embedding dimension (m), $y_{is_{monday}}$, $y_{is\_teusday}$, $y_{is\_wednesday}$, $y_{is\_thrusday}$, y_{is_friday}, $y_{is\_saturday}$, y_{is_sunday} are the corresponding exogenous variables for the Monday, teusday, Wednesday, Thursday, Friday, Saturday, Sunday respectively.

However, if the chaos is absent, it is considered as non-chaos series. Hence, the time series depends only on the lag/delay time (τ). The mathematical formulation of the non-chaotic time series along with exogenous dummy variable is presented in Eq. (17).

$$\widehat{Y_t} = f(y_{t-\tau}, y_{is\_monday}, y_{is\_teusday}, y_{is\_wednesday}, y_{is\_thrusday}, y_{is\_friday}, y_{is\_saturday}, y_{is\_sunday} \tag{17}$$

where $\widehat{Y}$ is forecasted value at time t, lag/delay time (τ), $y_{is_{monday}}$, $y_{is\_teusday}$, $y_{is\_wednesday}$, $y_{is\_thrusday}$, y_{is_friday}, $y_{is\_saturday}$, y_{is_sunday} are the corresponding exogenous variables for the Monday, teusday, Wednesday, Thursday, Friday, Saturday, Sunday respectively.

- ***Extracting exogenous dummy variable from time index:*** Extract "*day_of_the_week*" the dummy features from the index and apply the one-hot-encoding to get a single feature for every week. Now we get seven extra features, namely, *is_monday, is_thuesday, is_wednesday, is_thursday, is_friday, is_saturday* and *is_sunday*. All these exogenous dummy variable are considered in the both chaotic and non-chaotic time series data before invoking any ML/DL technique.

Stage 2: Forecasting ATMs cash demand using ML/DL techniques

In a year, first 11 months data of each ATM series is considered as training data, and the last 30 days are considered as the test set. The model consists of the reconstructed phase space and the exogenous dummy variable as follows. Fig. 5 depicts the schematic of the proposed hybrid methodology.

5. Dataset description and evaluation metrics

We collected the dataset from a well-known Indian commercial bank for this study. The dataset contains daily cash withdrawals of the past two years, starting from 21-11-2017 to 20-11-2019 for 100 ATMs.

5.1. Dataset pre-processing

Because of the technical problems, a few ATMs were out of service in some days, which led to missing entries in the dataset. If the dataset contains too many missing entries, then the model may fit the imputed values. Therefore, we removed the data of 50 out of 100 ATMs as the number of missing entries are greater than 110 in those cases. We then imputed missing entries (withdrawals) for the rest 50 ATMs with the novel imputation technique as mentioned in the previous section. As discussed in the Section 4.1, the value K is a user defined parameter. In current research study we have maintained $K = 10$.

We employed the imputation method described in section 4.1 before starting any other step here. We then used the Augmented Dickey-Fuller (Mushtaq, 2012) test from the statsmodel library to check whether the series is stationary or not. The test indeed showed that the withdrawals in all the ATMs series are stationary. To deseasonalize the series, we employed the median-based deseasonalization method of (Andrawis et al., 2011). We then computed the Lyapunov Exponent (Lyapunov, 1907) using Rosenstein's method for all the ATMs to check for the presence of chaos in the cash withdrawals series and the for 28 ATMs, the Lyapunov Exponent turns out to be higher than zero for all the ATMs, which confirms that chaos is present in the cash withdrawals series for these ATMs. Whereas the rest of the 22 ATMs, the Lyapunov Exponent is less than zero, which confirms that chaos is absent in these particular ATMs. As the machine learning and deep learning techniques require the data to be normalized, we performed min–max normalization on the dataset for all the ATMs.

Using Cao's method (Cao, 1997), minimum embedding dimension for all the ATMs turned out to be seven and using the partial autocorrelation function, the lag turned out to be one for each ATM series. We considered first 11 months data to be the training set and the last 30 days data to be the test set.

The following discussion is needed for taking this decision of splitting the data into training and test sets. In tabular data, for regression problems, in the hold-out method, one resorts to simple random sampling to determine the training and test set composition. Further, in tabular data, for classification problems, we resort to stratified random sampling to form the training and test sets. However, in time series data, neither simple random sampling nor stratified random sampling is meaningful because while randomly shuffling the data points the meaning of time (temporal aspect of the data) is lost. Therefore, it is an established practice to consider the first 80 % (or 90 %) data points as training data and the subsequent 20 % (or 10 %) data points as test data as can be evidenced from many works (Andrawis et al., 2011, Taieb, Bontempi, Atiya, & Sorjamaa, 2012, Coyle, Prasad, & McGinnity, 2010, Yankov, DeCoste, & Keogh, 2006, Venkatesh, Ravi, Prinzie, & Van Den Poel, 2014, Ravi, Pradeepkumar, & Deb, 2017). Several studies were found in the literature which follows the same kind of train-test split mechanism (Wang, Smith, & Hyndman, 2006; Nag & Mitra, 2002; Montero-Manso, Athanasopoulos, Hyndman, & Talagala, 2019). Also, many of the competitors had used this kind of mechanism in various competitions such as NN3 (Crone, Hibbon, & Nikolopolous, 2011), and M4 competitions (Makridakis, Spiliotis, & Assimakopoulos, 2018). Hence, this forms as a baseline for us to follow the train-test split mechanism.

It goes without saying that different kinds of train-test split ratios produce distinct results because, without exception, all statistical techniques and machine learning/deep learning techniques are very sensitive to the composition of the training data. This is an established fact in machine learning literature often referred to as "no-free-lunch-theorem", a concept originated in evolutionary computation.

Despite this fact, in nowcasting (which is increasingly becoming significantly important in the domains such as finance, weather, and cybersecurity etc.,), one could follow a sliding window-based approach that divides the given data, such that a window of (n-1) months' data is taken as the training set and the nth month's data as the test set. Then, the window moves on in the forward direction. But nowcasting is out of scope for the current work.

5.2. Evaluation measures

We considered SMAPE to measure the prediction performance of all the techniques and across all ATMs because each ATMs follow different ranges of cash withdrawals. In addition to SMAPE, we also considered Directional Symmetry (Lawrence, 1991) statistic for comparing the movements of the forecasts with the actual values in both up and down directions and Theil's U Coefficient (Theil, 1966), (Gragner & Nebold, 1974) and (Bliemel, 1973) for measuring the closeness of the forecast values to the actual values.

5.2.1. Symmetric Mean absolute Percentage Error (SMAPE)

SMAPE is used as a primary evaluation measure in time series as it is invariant to the scale of the series data and is defined as given in Eq. (18).

$$SMAPE = \frac{100\%}{n}\sum_{t=1}^{n}\frac{|\widehat{Y}_t - Y_t|}{(|\widehat{Y}_t| + |Y_t|)/2} \tag{18}$$

where $\widehat{Y_t}$ = forecasted value at time t, Y_t is observed value at time t and n is the number of samples.

5.2.2. Theil's U Coefficient

Theil's U Coefficient is a relative accuracy measure that gives more weight to the larger errors, which can help eliminate techniques with large errors. If it is less than 1 then our forecast is better than random guessing whereas if it is greater than 1, then our forecast is worse than random guessing. It is defined as given in Eq.(19).

Table 2
Hyperparameters for all the techniques.

Model	Hyperparameters
ARIMA	(p, d, q) = (7,0,0)
MLP	2 layers, nodes in layer 1 = range(1, 64, step = 4), nodes in layer 2 = range(1, 8), lr = 0.1, momentum = 0.9
XGBOOST	n_estimators = range(100, 2000, step = 100), max_depth = range (2, 10)
RF	n_estimators = range(5, 50), depth = range(4, 15)
SVR	C = [0.01, 1, 10, 100, 200, 300, 400, 500, 1000], gamma = [0.0001, 0.0003, 0.0006, 0.001, 0.005, 0.01, 0.03, 0.06, 0.09], Kernel = ['linear', 'rbf','sigmoid', 'poly']
GRNN	Standard deviation = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7], smoothing factor=(0.01, 0.09)
GMDH	Max_layer_count = 50, ref_function_types = (linear, linear_cov, quardratic, cubic), alpha = 0.5, n_jobs = 4, admix_features = True.
1D CNN	N_filters = range(5, 70, step = 5), epochs = range(300, 2000, step = 200), kernel_size = 2, activation= 'relu', dense layer = range(2, 8), optimizer= 'adam', loss= 'mse'.
LSTM	Nodes range = (2, 10), epochs = range(300, 1000, step = 100), optimizer= "adam". Lr = 0.01.
GRU	Nodes range = (2, 10), epochs = range(300, 1000, step = 100), optimizer= "adam". Lr = 0.01
1D-CNN + LSTM	N_filters = range(5, 70, step = 5), epochs = range(300, 2000, step = 200), kernel_size = 2, activation= 'relu', dense layer = range(2, 8), optimizer= 'adam', loss= 'mse'. Nodes range = (2, 10), epochs = range(300, 1000, step = 100), optimizer= "adam". Lr = 0.01
1D-CNN + GRU	N_filters = range(5, 70, step = 5), epochs = range(300, 2000, step = 200), kernel_size = 2, activation= 'relu', dense layer = range(2, 8), optimizer= 'adam', loss= 'mse'. Nodes range = (2, 10), epochs = range(300, 1000, step = 100), optimizer= "adam". Lr = 0.01

$$Theil's\ U\ Coefficient = \sqrt{\frac{\sum_{t=1}^{n-1}\left(\frac{\widehat{y}_{t+1}-y_{t+1}}{y_t}\right)^2}{\sum_{t=1}^{n-1}\left(\frac{y_{t+1}-y_t}{y_t}\right)^2}} \tag{19}$$

where $\widehat{Y_t}$ = forecasted value at time t, Y_t is observed value at time t and n is the number of samples.

5.2.3. *Directional symmetry statistic (Dstat)*

Dstat is a statistical measure of a model's performance in predicting the direction of change, positive/negative, of a time series from a one-time period to the next. It is defined as given in Eq. (20).

$$Dstat(y_t, \widehat{y}_t) = \frac{100}{n-1}\sum_{i=2}^{n} d_i,\ where\ d_i = \begin{cases} 1, & if(y_i - y_{i-1})(\widehat{y}_i - \widehat{y}_{i-1})\rangle 0 \\ 0, & otherwise \end{cases} \tag{20}$$

where $\widehat{y}_t$ = forecasted value at time t, y_t is observed value at time t and n is the number of samples and direction of the value change is given by corresponding d_i.

6. Results and discussion

Table 2 presents the hyper-parameters and their ranges used for all the techniques across all 50 ATMs. After thorough experimentation, we presented that the ranges of the hyperparameters in Table 2, which yielded the best results for all techniques and all ATMs. In the experimentation, we found that the best hyperparameter combination varies from ATM to ATM because of the underlying differences in the withdrawal amounts. After employing the novel imputation technique, 28 ATMs were found to fall in the chaos-observed-ATMs category where the chaos is present, whereas chaos was not present in the rest 22 ATMs and hence they fall in non-chaos-observed-ATMs category. Table 3 presents the results of our study after tuning the hyper-parameters with and without an exogenous dummy variable namely day-of-the-week. It clearly shows that chaos modelling and including exogenous dummy variables in the data set will result in improved forecasts. We considered ARIMA as a baseline model for our study as it is a robust statistical model. All ML and DL techniques except XGBoost outperformed the ARIMA model in terms of SMAPE in both the categories. We avoided the risk of overfitting by following the early stopping criteria in all the

Table 3
Results of chaos-observed-ATMs.

chaos-observed-ATMs												
	Without Exogenous dummy variable						With Exogenous dummy variable					
	SMAPE		Dstat		Theils U		SMAPE		Dstat		Theils U	
Model	Mean (SD)	Median	Mean (SD)	Median	Mean (SD)	Median	Mean (SD)	Median	Mean (SD)	Median	Mean (SD)	Median
ARIMA	29.63 (7.80)	28.96	50.36 (5.66)	51.41	0.41 (0.12)	0.38	28.56 (6.51)	28.93	56.96 (2.47)	57.45	0.39 (0.17)	0.33
MLP	28.93 (7.30)	26.2	51.32 (6.73)	52.36	0.4 (0.12)	0.37	27.55 (7.32)	25.14	62.13 (4.62)	62.41	0.38 (0.12)	0.36
XGBOOST	30.9 (4.87)	28.8	53.26 (5.87)	52.78	0.37 (0.16)	0.36	29.45 (5.2)	27.3	61.89 (6.81)	62.78	0.36 (0.15)	0.35
RF	25.71 (7.52)	25.83	**61.56 (3.12)**	**62.88**	0.43 (0.21)	0.41	24.75 (7.38)	23.66	**64.12 (5.61)**	**64.89**	0.41 (0.13)	0.4
SVR	29.03 (7.43)	26.71	52.74 (2.45)	51.23	0.41 (0.13)	0.39	27.96 (6.95)	25.76	59.86 (2.63)	58.26	0.4 (0.16)	0.35
GRNN	25.69 (5.05)	25.83	54.96 (6.78)	55.12	0.39 (0.12)	0.42	25.03 (5.35)	23.93	59.13 (4.18)	58.15	0.42 (0.17)	0.41
GMDH	27.60 (7.51)	25.43	53.46 (4.29)	54.8	0.41 (0.14)	0.42	24.59 (5.58)	25.18	59.66 (3.63)	60.12	0.36 (0.13)	0.36
1D-CNN	24.70 (5.48)	24.86	53.61 (3.11)	53.14	0.38 (0.17)	0.39	24.59 (5.33)	24.82	62.13 (6.62)	63.14	0.37 (0.17)	0.39
LSTM	**24.17 (4.02)**	**24.48**	60.56 (4.44)	58.63	0.34 (0.13)	**0.32**	**22.75 (3.19)**	**23.06**	62.89 (7.52)	62.56	**0.32 (0.14)**	**0.32**
GRU	29.03 (7.44)	26.45	60.54 (2.18)	60.78	0.34 (0.14)	0.33	28.84 (7.44)	24.74	63.75 (4.73)	63.15	0.33 (0.16)	0.31
1D-CNN + LSTM	24.57 (3.07)	24.08	61.48 (2.64)	62.01	**0.33 (0.18)**	0.35	23.46 (3.67)	24.64	62.46 (5.87)	62.48	0.33 (0.14)	0.34
1D-CNN + GRU	28.70 (4.97)	28.56	58.89 (4.15)	57.45	0.36 (0.14)	0.34	27.6 (4.45)	27.66	61.12 (6.38)	62.09	0.33 (0.13)	0.32

Table 4
Results of the non-chaos-observed-ATMs.

non-chaos-observed ATMs												
	Without Exogenous dummy variable						With Exogenous dummy variable					
	SMAPE		Dstat		Theils U		SMAPE		Dstat		Theils U	
Model	Mean (SD)	Median	Mean (SD)	Median	Mean (SD)	Median	Mean (SD)	Median	Mean (SD)	Median	Mean (SD)	Median
ARIMA	28.49 (7.62)	27.45	53.36 (4.56)	52.89	0.4 (0.12)	0.38	27.45 (5.71)	27.01	57.16 (5.67)	57.12	0.4 (0.15)	0.39
MLP	28.05 (7.19)	26.24	56.45 (3.67)	55.64	0.42 (0.16)	0.41	28.26 (7.02)	29.09	61.45 (4.68)	61.89	0.39 (0.16)	0.36
XGBOOST	29.56 (5.27)	29.66	58.96 (7.47)	58.04	0.38 (0.14)	0.36	28.54 (5.70)	28.04	60.45 (5.12)	61.71	0.37 (0.14)	0.36
RF	27.24 (7.52)	25.76	61.12 (5.67)	61.56	0.44 (0.13)	0.42	27.24 (7.38)	25.76	63.12 (2.39)	**62.75**	0.42 (0.13)	0.41
SVR	29.03 (7.15)	26.71	56.12 (5.89)	57.45	0.36 (0.16)	0.34	27.96 (5.89)	25.76	58.75 (4.51)	59.12	0.39 (0.15)	0.38
GRNN	25.89 (4.57)	25.06	59.64 (6.12)	58.96	0.37 (0.16)	0.36	24.8 (5.13)	24.86	59.12 (4.55)	57.45	0.37 (0.14)	0.37
GMDH	24.91 (7.51)	24.78	60.12 (3.45)	59.67	0.39 (0.16)	0.39	24.78 (5.58)	24.56	60.45 (3.37)	61.72	0.38 (0.16)	0.36
1D-CNN	23.67 (4.00)	24.48	60.12 (4.56)	61.26	0.36 (0.17)	0.36	23.48 (3.51)	23.35	62.81 (4.09)	63.48	0.36 (0.18)	0.34
LSTM	**22.17 (3.08)**	**23.65**	**62.45 (2.67)**	63.87	0.36 (0.14)	0.37	**22.08 (2.57)**	**23.01**	**62.79 (2.36)**	60.73	0.34 (0.12)	0.31
GRU	24.66 (3.57)	25.01	61.23 (3.45)	60.89	0.35 (0.15)	0.33	23.94 (3.04)	24.07	61.45 (2.01)	63.12	0.35 (0.13)	0.35
1D-CNN + LSTM	24.54 (3.94)	24.88	62.41 (5.67)	**64.12**	**0.33 (0.13)**	**0.31**	23.18 (3.19)	23.67	62.71 (1.87)	60.15	**0.32 (0.12)**	**0.31**
1D-CNN + GRU	24.56 (3.67)	24.18	62.1	61.05	0.33 (0.14)	0.31	23.826 (3.02)	24.85	62.19	60.78	0.32 (0.13)	0.33

Table 5
t-test for comparing top three performing models for chaos-observed-ATMs.

t test significance level 5 %	p-value
RF vs 1D-CNN + LSTM	0.177
RF vs LSTM	0.0332
LSTM vs 1D-CNN + LSTM	0.0025

Table 6
t-test for comparing top three performing models non-chaos-observed ATMs.

t test significance level 5 %	p value
1D-CNN vs LSTM	0.114
LSTM vs 1D-CNN + LSTM	0.037
1D-CNN vs 1D-CNN + LSTM	0.4328

neural network techniques (DL techniques subsumed).

From Table 3, one can observe that in the chaos-observed-ATMs category, LSTM yielded the best performance in terms of SMAPE and Dstat with exogenous dummy variable, whereas without using exogenous dummy variable, 1D-CNN + LSTM turned out to be the best technique in terms of SMAPE. But, according to Dstat, RF turned out to be the best with or without exogenous dummy variable. When Theils U statistic is considered, LSTM is the best without considering the exogenous dummy variable and 1D-CNN + LSTM is the best while considering exogenous dummy variable. Therefore, in order to have a categorical inference from the study, we performed paired *t*-test on the mean SMAPE of the top three models on the test data at 50 + 50–2 = 98 degrees of freedom at a 5 % level of significance to check whether they are statistically performing the same or not. The results are presented in Table 5. It turned out that the LSTM and 1D-CNN + LSTM produced similar performance when compared with RF. However, LSTM turned out to be significant than the 1D-CNN + LSTM with or without exogenous dummy variable.

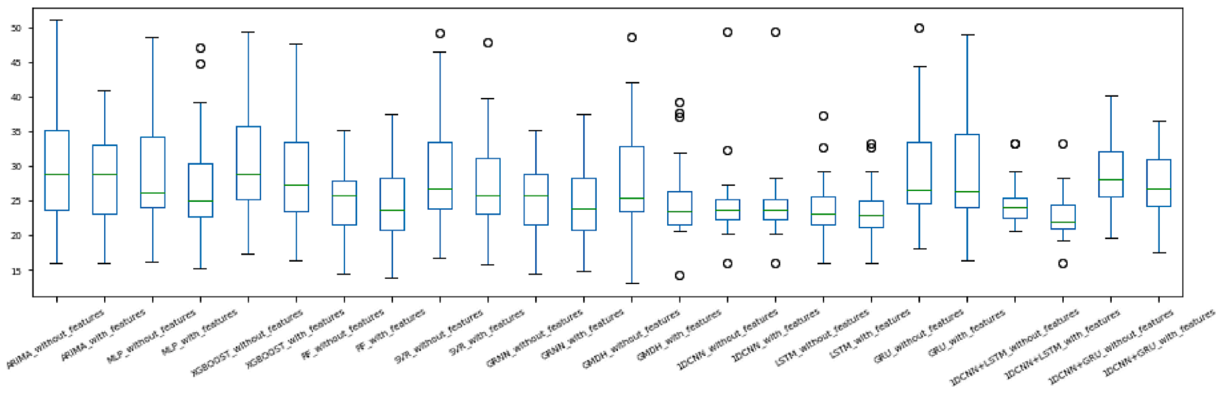

Fig. 9. Box plots for both ML and DL Techniques on the chaos-observed-ATMs.

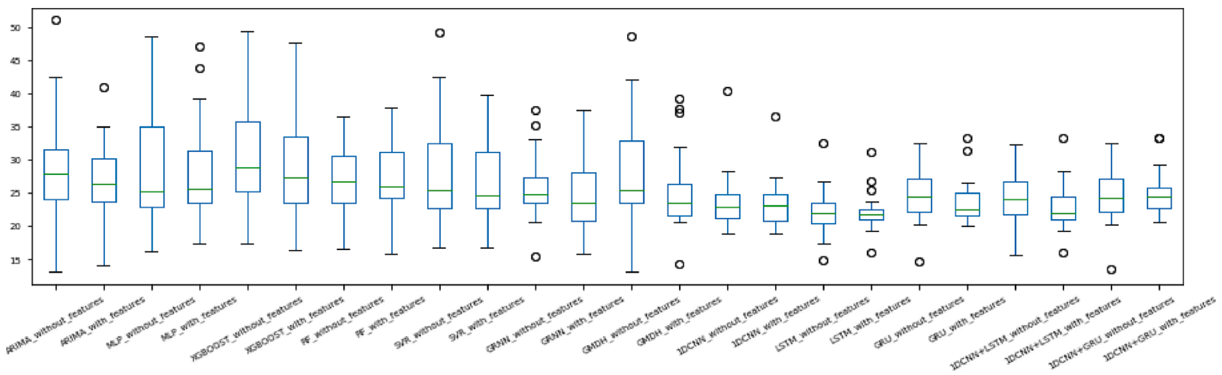

Fig. 10. Box plots for both ML and DL Techniques on the non-chaos-observed-ATMs.

From Table 4, one can observe that in the non-chaos ATM category, LSTM yielded the best performance in terms of SMAPE and Dstat with or without exogenous dummy variable. But, according to Theils U statistic, 1D-CNN + LSTM turned out to be the best with or without exogenous dummy variable. In this category also, we performed paired *t*-test on the mean SMAPE of the top three models on the test data at 50 + 50–2 = 98 degrees of freedom at a 5 % level of significance to check whether they are statistically performing the same or not. The results are presented in Table 6. It turned out that the LSTM and 1D-CNN + LSTM produced statistically similar performance when compared with 1D-CNN.

Fig. 9 and Fig. 10 depict the Boxplots of the SMAPE values on the test data for each technique with and without exogenous dummy variable in the case of chaos-observed-ATMs and non-chaos-observed ATMs respectively. It clearly shows adding an exogenous dummy variable leads to an increase in the interquartile range (IQR) for each algorithm while lowering the median SMAPE. Here, we have to two choices, (i) Do not consider extra features if getting smaller IQR is objective and be ready to sacrifice the improvements in median SMAPE of all methods. (ii) Consider extra features to get improvement in the median SMAPE coupled with an increment in the IQR. Thus there is a trade-off. So, the decision to include these features is left to the domain expert. The ML models did not perform well as they are not able to effectively learn sequential information like LSTM or 1D CNN.

The limitation of the current work is as follows: the proposed models could not yield significantly lower SMAPE values primarily because all of them without exception failed to completely capture the underlying non-linearity and randomness in the data. One possible solution would be to design weighted mean based ensemble of the top five methods where the weights are chosen in proportion to the performance of the individual techniques on the test data. Alternatively, taking cue from Ravi et al. (2017), one could model the forecasting problem as a bi-objective optimization problem where SMAPE and Dstat would be considered as two objective functions and an evolutionary algorithm guides the forecasting process, where the coefficients corresponding to the features in equations (16) and (17) turn out to be decision variables. These two strategies are left for future work. The fact that hybrid deep learning techniques also could not achieve spectacular median SMAPE values and Dstat indicates that further research is still warranted by devising novel methods. In the future, we would like to see how transfer learning can be applied to time series, which might improve the performance on smaller time-series datasets.

7. Conclusions

In this study, we developed a hybrid model comprising chaos modeling and machine learning/deep learning in tandem for forecasting cash withdrawals in ATMs of an Indian bank. This is an important operational problem for any bank, which would like to optimize its ATM replenishment strategies. We also studied the impact of the exogenous dummy variable such as day-of-the-week variables on the time series predictions instead of just using the raw daily time series data. In the case of chaos-observed-ATMs, all the forecasting techniques showed an improvement in the SMAPE after adding mentioned exogenous dummy variable. Out of all the techniques, LSTM turns out to be the best model in terms of mean SMAPE and mean Thiel's U coefficient both in exogenous and non-exogenous categories with 5.8 % and 5.9 % improvement respectively. While RF performed the best with respect to the mean Dstat in both categories and it showed 4 % improvement in exogenous category. However, in the case of non-chaos-observed ATMs, there is a very slight improvement in all metrics even after considering exogenous dummy variable. After analyzing the box plots, we noticed that the models with exogenous dummy variable displayed less variability in predictions. The less variability indicates that the underlying model yielded stable predictions in all the chaos-observed-ATMs. Specifically, even though LSTM outperformed GRU with respect to SMAPE the latter displayed less variability. We noticed identical behaviour in the box plots of the non-chaos-observed ATMs category.

Interestingly, the practical utility of the proposed methodology is immense and not just limited to the current research problem. It can also be applied seamlessly to effectively solve various problems such as: (i) cash transferred from one account to others account daily – a well-known proxy for nowcasting the GDP. (ii) amount spent in digital payment channels as a proxy for quantification of digital economy. (iii) transportation sector– got an example from NN3 competition. (iv) daily electricity load forecasting. (v) daily footfalls in a metro system to assess its viability etc. This fact demonstrates the utility of the proposed methodology.

CRediT authorship contribution statement

Vangala Sarveswararao: Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Vadlamani Ravi:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Yelleti Vivek:** Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, 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.

Data availability

The authors do not have permission to share data.

Acknowledgement

No funding was received for this research.

References

Pradeepkumar, D., & Ravi, V. (2018). Soft computing hybrids for FOREX rate prediction: A comprehensive review. *Computers & Operations Research, 99*, 262–284. https://doi.org/10.1016/j.cor.2018.05.020

Venkatesh, K., Ravi, V., Kumar D.N. (2014). Chaotic time series analysis with neural networks to forecast cash demand in ATMs, in:2014 IEEE Int. Conf. Comput. Intell. Comput. Res., Coimbatore, Tamilnadu, India.

Crone, S. (2008). Results of the NN5 time series forecasting competition, in: WCCI 2008, IEEE World Congr. Comput. Intell., Hong Kong, China.

Andrawis, R. R., Atiya, A. F., & El-Shishiny, H. (2011). Forecast combinations of computational intelligence and linear models for the NN5 time series forecasting competition. *International Journal of Forecasting, 27*(672–688).

Venkatesh, K., Ravi, V., Prinzie, A., & Van Den Poel, D. (2014). Cash demand forecasting in ATMs by clustering and neural networks. *European Journal of Operational Research, 232*, 383–392.

Shumway, R. H., & Stoffer, D. S. (2017). *Time Series Analysis and Its Applications With R Examples* (4th ed.). Cham: Springer.

Specht, D. F. (1991). A general regression neural network. *IEEE Transactions on Neural Networks, 2*, 568–576.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536. https://doi.org/10.1038/323533a0

Walter, G. G., Miao, Y., Zhang, J., & Wayne Lee, W. N. (1995). Wavelet neural networks for function learning. *IEEE Transactions on Signal Processing, 43*, 1485–1497. https://doi.org/10.1109/78.388860

Ivakhnenko, A. G. (1966). The group method of data handling- a rival of the method of stochastic approximation. *Soviet Automatic Control, 13*(3), 43–55.

Javanmard, M. (2016). An approach to improve forecasting cash demand at ATMs using fuzzy logic. *International Journal of Computer Science and Information Security, 14*, 404.

Bhandari, R., & Gill, J. (2016). An artificial intelligence ATM forecasting system for hybrid neural networks. *International Journal of Computers and Applications, 133*, 13–16.

Jadwal, P.K., Jain, S., Gupta, U., Khanna, P. (2017). K-Means clustering with neural networks for ATM cash repository prediction, in: Int. Conf. Inf. Commun. Technol. Intell. Syst., Ahmedabad, India, (pp. 588–596).

Arabani, S. P., & Komleh, H. E. (2019). The improvement of forecasting ATMs cash demand of IranBanking network using convolutional neural network. *Arab. J. Sci. Eng., 44*, 3733–3743.

Rafi, M., Wahab, M.T., Khan, M.B., Raza, H. (2020). ATM Cash Prediction Using Time SeriesApproach, in: 2020 3rd Int. Conf. Comput. Math. Eng. Technol., Sukkur, Pakistan, (pp. 1–6).

Ravi, V., Pradeepkumar, D., & Deb, K. (2017). Financial time series prediction using hybrids of chaostheory, multi-layer perceptron and multi-objective evolutionary algorithms. *Swarm and Evolutionary Computation, 36*, 136–149. https://doi.org/10.1016/j.swevo.2017.05.003

Pradeepkumar, D., Ravi, V. (2014). Forex rate prediction using chaos, neural network and particle swarm optimization, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), Springer Verlag, (pp. 363–375).

Pradeepkumar, D., Ravi, V. (2016). FOREX Rate prediction using Chaos and Quantile Regression Random Forest, in: 2016 3rd Int. Conf. Recent Adv. Inf. Technol. RAIT 2016, Institute of Electrical and Electronics Engineers Inc., (pp. 517–522).

Pradeepkumar, D., Ravi, V. (2017). FOREX rate prediction: A hybrid approach using chaos theory and multivariate adaptive regression splines, in: Adv. Intell. Syst. Comput., Springer Verlag, (pp. 219–227). 10.1007/978-981-10-3153-3_22.

Schäfer, A.M., Zimmermann, H.G. (2006). Recurrent neural networks are universal approximators, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), Springer Verlag, Athens, Greece, (pp. 632–640). 10.1007/11840817_66.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation, in: EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf., Association for Computational Linguistics (ACL), (pp. 1724–1734). 10.3115/v1/d14-1179.

Gragner, C. W. J., & Nebold, P. (1974). Spurious regressions in econometrics. *Journal of Econometrics, 2*(2), 111–120. https://doi.org/10.1016/0304-4076(74)90034-7

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Tian, Y., Pan, L. (2015). Predicting Short-Term Traffic Flow by Long Short-Term Memory Recurrent Neural Network, in: 2015 IEEE Int. Conf. Smart City/SocialCom/ SustainCom, Chengdu, China, (pp. 153–158). 10.1109/SmartCity.2015.63.

Duan, Y., Lv, Y., Wang, F.Y. (2016). Travel time prediction with LSTM neural network, in: IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC, Rio de Janeiro, Brazil, (pp. 1053–1058).

Lee, C.Y., Xie, S., Gallagher, P.W., Zhang, Z., Tu, Z. (2015). Deeply-Supervised Nets, in: Artif. Intell. Stat., (pp. 562–570).

LeCun, Y., Bengio, Y. (1998). Convolutional Networks for Images, Speech, and Time-Series, Handb. Brain Theory Neural Networks. 3361.

Papadopoulos, K. (2018). SeriesNet: A Dilated Causal Convolutional Neural Network for Forecasting, https://github.com/kristpapadopoulos/seriesnet (accessed December 5, 2019).

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K. (2016) WaveNet: A Generative Model for Raw Audio. http://arxiv.org/abs/1609.03499 (accessed December 5, 2019).

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *J. Atmos. Sci., 20*, 130–141.

Dhanya, C. T., & Nagesh Kumar, D. (2010). Nonlinear ensemble prediction of chaotic daily rainfall. *Adv. Water Resour., 33*, 327–347. https://doi.org/10.1016/j.advwatres.2010.01.001

Packard, N. H., Crutchfield, J. P., Farmer, J. D., & Shaw, R. S. (1980). Geometry from a time series. *Phys. Rev. Lett., 45*, 712–716. https://doi.org/10.1103/PhysRevLett.45.712

Rosenstein, M. T., Collins, J. J., & De Luca, C. J. (1993). A practical method for calculating largest Lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena, 65*, 117–134.

Lyapunov, A. (1907). Problème général de la stabilité du mouvement, in: Ann. La Fac. Des Sci. Toulouse Mathématiques, (pp. 203–474).

Cao, L. (1997). Practical method for determining the minimum embedding dimension of a scalar time series. *Physica D: Nonlinear Phenomena, 110*, 43–50.

Drucker, H. Burges, C.J.C., Kaufman, L., Smola, A.J., Vapnik, V. (1997). Support Vector Regression Machines, in: M.C. Mozer, M.I. Jordan, T. Petsche (Eds.), Adv. Neural Inf. Process. Syst. 9, MIT Press, (pp. 155–161).

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*, 273–297.

Ho, T.K. (1995). Random decision forests, in: Proc. Int. Conf. Doc. Anal. Recognition, ICDAR, Montreal, Canada. 10.1109/ICDAR.1995.598994.

Chen, T., Guestrin, C. (2016). XGBoost: A scalable tree boosting system, in: Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., San Francisco, California, USA, (pp. 785–794).

Friedman, J.H. (2001). Greedy function approximation: a gradient boosting machine, Ann. Stat. (pp. 1189–1232).

Mushtaq, R. (2012). Augmented Dickey Fuller Test, SSRN Electron. J. Lawrance, A.J. (1991). Directionality and Reversibility in Time Series, Int. Stat. Rev./Rev. Int. Stat. 59. 10.2307/1403575.

Theil, H. (1966). Applied economic forecasting, North-Holland Pub. Co, Amsterdam. Granger, C.W.J., Newbold, P. (1973). Some comments on the evaluation of economic forecasts, Appl. Econ. 5 (pp. 35–47). 10.1080/00036847300000003.

Bliemel, F. (1973). Theil's forecast accuracy coefficient: A clarification. *J. Mark. Res., 10*, 444.

Cho, K., van, M.B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio. Y. (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". arXiv:1406.1078.

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing, 90*, Article 106181. https://doi.org/10.1016/j.asoc.2020.106181

Huang, J., Chai, J., & Cho, S. (2020). Deep learning in finance and banking: A literature review and classification. *Frontiers of Business Research in China, 14*, 13. https://doi.org/10.1186/s11782-020-00082-6

Jiang, W. (2021). Applications of deep learning in stock marketing prediction: Recent progress. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2021.115537

Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020). Deep learning for financial applications: A survey. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2020.106384

Sengupta, S., Basak, S., Saikia, P., Paul, S., Tsalavoutis, V., Atiah, F., … Peters, A. (2020). A review of deep learning with special emphasis on architectures, applications and recent trends. *Knowledge-Based Systems*. https://doi.org/10.1016/j.knosys.2020.105596

Taieb, S. B., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications, 39*(8), 7067–7083. https://doi.org/10.1016/j.eswa.2012.01.039

Coyle, D., Prasad, G., & McGinnity, T. M. (2010). On utilizing self-organizing fuzzy neural networks for financial forecasts in the NN5 forecasting competition. In *The 2010 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). https://doi.org/10.1109/IJCNN.2010.5596955

Wang, X., Smith, K., & Hyndman, R. (2006). Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery, 13*, 335–364. https://doi.org/10.1007/s10618-005-0039-x

Nag, A. K., & Mitra, A. (2002). Forecasting daily foreign exchange rates using genetically optimized neural networks. *Journal of Forecasting, 21*, 501–511. https://doi.org/10.1002/for.838

Montero-Manso, P., Athanasopoulos, G., Hyndman, R., & Talagala, T. (2019). FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting., 36*. https://doi.org/10.1016/j.ijforecast.2019.02.011

Makridakis, S., Spiliotis, E., Assimakopoulos, V. (2018). The M4 Competition: Results, findings, conclusion and way forward, International Journal of Forecasting, 34(4), 802-808, 10.1016/j.ijforecast.2018.06.001.

Yankov, D., DeCoste, D., Keogh, E. (2006). Ensembles of nearest neighbor forecasts. Machine learning: ECML 2006. Lecture Notes in Computer Science, vol 4212. Springer, Berlin, Heidelberg. 10.1007/11871842_51.

Crone, S. F., Hibbon, M., & Nikolopolous, K. (2011). Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting, 27*(3), 635–660. https://doi.org/10.1016/j.ijforecast.2011.04.001

Manoj, T., Kumar, D. (2018). A hybrid financial trading support system using multi-category classifiers and random forest, Applied Soft Computing, 67, pp. 337-349, 10.1016/j.asoc.2018.03.006.

Gao, Y., Liu, J. (2021). Potential User Prediction for Financial APP Based on Random Forest Model," 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2021, pp. 180-185, 10.1109/CSCWD49262.2021.9437776.

Liu, S., Zhang, C., Ma, J. (2017). CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets. In: Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, ES. (eds) Neural Information Processing. ICONIP 2017. Lecture Notes in Computer Science, vol 10635. Springer, Cham. 10.1007/978-3-319-70096-0_21.

Sercan, K., Ugur A. (2017). A deep learning approach for optimization of systematic signal detection in financial trading systems with big data, International Journal of Intelligent Systems and Applications in Engineering. Special Issue (pg.31–36).

Kavitha, S., Varuna, S., Ramya R. (2016). A comparative analysis on linear regression and support vector Regression, 2016 Online International Conference on Green Engineering and Technologies (IC-GET), 2016, pp. 1-5, 10.1109/GET.2016.7916627.