Neural Computing and Applications
https://doi.org/10.1007/s00521-021-06744-7

S.I.: DEEP LEARNING FOR TIME SERIES DATA

Improving temporal predictions through time-series labeling using matrix profile and motifs

Pratik Saha[1] · Pritthijit Nath[2] · Asif Iqbal Middya[2] · Sarbani Roy[2]

Received: 13 April 2021 / Accepted: 10 November 2021

Abstract
One of the most challenging tasks in time-series prediction is a model's capability to accurately learn the repeating granular trends in the data's structure to generate effective predictions. Traditionally specially tuned statistical models and deep learning models like recurrent neural networks and long short-term memory networks are used to tackle such problem of sequence modeling. However in practice, factors like inadequate parameters in case of statistical models, random weight initializations, and data inadequacy in case of deep learning models affect the resulting final predictions. As a possible solution to these known problems, this paper introduces a novel method of time-series labeling (TSL) comprising a combination of encoding and decoding methodologies that not only takes into account the granular structure of a time-series data but also its underlying meta-learners for better predictive accuracy. To demonstrate the approach's effectiveness and capability of handling wide range of scenarios, comparisons are drawn first over different widely used statistical and deep learning models and then applying TSL to each of them in order to showcase the resulting performance improvement when implemented over a wide variety of real-world datasets. The experimental findings reflect an average of 25% increase in overall performance when using TSL along with mostly similar performance of different combinations regardless of model complexity thereby proving its efficacy in predicting periodic data.

Keywords Temporal predictions · Granularity · Matrix profile · Motifs

1 Introduction

Time-series prediction [1, 2] poses a difficult type of predictive modeling problem where along with classical regression dependencies, complexity of repeating or non-repeating dependence in its granular structure is also present. Traditional deep learning models like long short-term memory networks (LSTMs) [3] are used to predict time series as they are capable of naturally learning features from sequence data which makes them particularly suitable for the task. In addition, various statistical approaches like Auto-Regressive (AR) [4], Auto-Regressive Integrated Moving Average (ARIMA) [4] along with its seasonal extension (SARIMA) [5] when specifically tuned to take into account the periodicity and time lags of a time-series data performs the task of prediction practically well. However neither of these approaches takes into account the dependence among the repeating granular subsequences that could be used further for potentially increasing the model's prediction accuracy.

Granularity [6] of a time series refers to the partition of the entire temporal series into periodically repeating time intervals grouped by some functionality or similarity. Motifs [7] also know as time-series chains are approximately repeating subsequences which at its core are building blocks of a particular time series. Supervised learning models like LSTMs try to learn these trends of

✉ Sarbani Roy
sarbani.roy@jadavpuruniversity.in

Pratik Saha
pratiksaha198@gmail.com

Pritthijit Nath
pritthijit.nath@ieee.org

Asif Iqbal Middya
asifim.rs@jadavpuruniversity.in

[1] Department of Computer Science, SRM University, Kattankulathur, Chennai, India

[2] Department of Computer Science and Engineering, Jadavpur University, Kolkata, India

Published online: 10 January 2022

evolving granular subsequences which are then used to predict values over time. However, in certain circumstances, akin to any other learner method they are prone to overfitting [8], getting affected by randomness in weight initializations [3] and requiring large volumes data for accurate predictions [3]. As a possible solution to facilitate its initial learning process, the similarity between granular structure of the data can be used as label encoded time-series chains which can act as a pre-processing step before training a model, predictions of which can be further post-processed by utilizing the evolving trends (centrality measure) in those motifs, improving overall model prediction. These sequence of steps also prove to be quite effective for long-term predictions by taking into account the varying periodic features in due process.

Previous research work conducted in this field took into account the granularity of a time series or the meta-learners for better prediction accuracy. Leite et al. [9] proposed optimal-granular-fuzzy (OGF) rule-based models which took into account the variability and coverage of data during the process of modeling data streams along with statistical methods like ordered-weighted-averaging (OWA) for forecasting, coupled with model ensembles. Shao et al. [10–12] studied that impact of disturbance, noise, etc., can hamper the system's behaviour to a great extent. Afolabi et al. [13] investigated the use of meta-learners using the moving average (MA) method for achieving noise reduction during the forecasting of a time series resulting in a low signal-to-noise ratio and a scalable multi-cluster system. Although being crucial works in externally aided model training, these processes did not take into account the similarity of the repeating granular subsequences of a time series and how they could potentially help in the learning process.

Taking motivation from the recent advancements in this field, this paper proposes a novel time-series labeling (TSL) method which uses the granularity of a time series by taking into account the repeating nature of the underlying granular subsequences. This study investigates the usage of TSL when coupled with deep learning models like LSTMs or statistical methods like AR, ARIMA and SARIMA, to check whether it results in better prediction accuracy while also addressing numerous shortcomings as mentioned before. The approach undertaken uses different widely used deep learning and statistical models to predict a time series into the future. TSL is then applied to each of the methods and prediction is performed over the same period of time in the future. Finally comparison are drawn over the performance improvement with ordinary methods and ordinary methods + TSL for the same task. In TSL, time-series chains in the original temporal domain are label encoded and passed on as input to the models to obtain label encoded predictions, which are then decoded back to the original domain using the trend present in the centrality measure of subsequences belonging to the time-series chain of a particular type of label present in the training set. TLS could have significant applications in time-series forecasting [1, 14], temporal analysis of different city dynamics [15–17], and many other domains of research [18–24].

To summarize, the main contributions presented in this paper are as follows :

- A novel time-series labeling (TSL) method is proposed to improve time-series prediction that takes into account the similarity between repeating granular subsequences and the trend of the centrality measure of subsequences.
- Evaluation of model peformance gain on real world datasets in using TSL which feeds a label encoded time-series chains [7] as pre-processing features on a temporal dataset as input to different sequence modeling methods.
- Presenting evidence of similar overall performance in models of varying complexity and discussing the choice of using simpler models in situations when limited in terms of computational resources.

The rest of the paper is set out as follows: In Sect. 2, recent relevant studies conducted in this field are discussed in detail. Section 3 lays the groundwork of the different components that make up TSL. Section 4 demonstrates the problem formulation along with a detailed description of the method proposed for time-series labelling. The results of the study are laid out in great detail in Sect. 5, along with a detailed discussion on the performance of various models with and without using the proposed approach. The conclusions drawn from the findings are ultimately presented in Sect. 6.

2 Related works

This section deals with relevant research studies conducted in the field of time-series prediction utilizing the granularity of the data as an added feature engineered set for better performance.

Hmouz et al. [6] developed a time-series architecture, where the series was split into temporal windows leading to formation of information granules. Later they used clustering techniques to build a spatial representation of the data along with the use of swarm optimization during model prediction. Guo et al. [33] proposed an approach to transform the original data using principles of justifiable granularity. They used dynamic time warping (DTW) to adjust lengths of temporal sequences and exploited hidden markov models (HMM) to derive relations between

granular time series and performed comparative analysis through experiments on publicly available datasets. A fuzzy set-based granular evolving modeling (FBeM) approach was proposed by Leite et al. [34] for learning features from imprecise datasets. This led them to take into account endless flows of non-stationary data and structural adaptation of models, with the FBeM outperforming similar approaches when performed on classic Box–Jenkins and Mackey–Glass benchmarks. A previous study by Nath et al. [14] resampled the daily pollution data into monthly to better forecast the long-term trends.

The choice of choosing the most promising individual method or a combination of methods for time-series forecasting was tackled by Lemke et al. [35]. Using different meta-learning approaches with an initial investigation on which models worked best in which situations, they improved their forecasting performance thereby showing that a ranking-based combination of methods outperformed simple model selection approaches. Ali et al. [36] investigated the use of additional data for improving the performance of meta-learning systems with usage of cross-domain transfer of meta-knowledge. They demonstrated a similarity-based cluster analysis to discover homogeneous groups of time-series concerning performance, along with minimizing the risk of selecting the least appropriate base-learners. A. Abraham [37] presented a meta-learning evolutionary artificial neural network (MLEANN) in which an adaptive architecture comprising of activation functions, connection weights, learning algorithm was used. He explored the performance of backpropagation, conjugate gradient, quasi-Newton, Levenberg–Marquardt algorithm chaotic time series. Its effectiveness as a framework to design neural networks that were smaller, faster, and had a better-generalized performance was also described. Other recent scientific works related to this study are summarized in Table 1.

The merits of the novel method proposed in this paper distinguishes itself from the related works described in the following ways. First, all the motifs of a time series are found out through similarities between repeating granular subsequences using their Z-normalized euclidian distances with respect to each other. These motifs are then label encoded and passed on as input to models capable of time-series prediction. Second, to aid in model training and counter problems like varying number of hyper-parameters and requirement of domain expertise for best parameter selection [38], the usage of a combination of encoding and decoding time series in TSL is employed where the trend in the amplitude of similar subsequences in each time-series chain can act as a meta-learning feature during prediction. Third, a comparative study of the model performance using and not using TSL with deep learning models like stacked LSTMs and statistical models like AR, ARIMA and SARIMA on various real-world datasets is also presented to demonstrate the method's efficacy.

3 Background

In this section, the underlying methodologies that constitute TSL namely the usage of matrix profile and time-series chains have been explained.

3.1 Matrix profile

A matrix profile [39] provides us with the distances between all subsequences and their corresponding nearest neighbours which is used as a part of our methodology to label encode a time series.

To better explain matrix profile, a few terminologies have to be formally introduced. In a time series T having length n, the i^{th} subsequence $T_{i,m}$ is subset of continuous values of length m written as $T_{i,m} = \{t_i,\ t_{i+1}, \ldots,\ t_{i+m-1}\}$ where $1 \leq i \leq n - m + 1$. For each $T_{i,m}$, a distance profile D_i can be computed after taking their Z-normalized euclidean distances between $T_{i,m}$ with all subsequences in the same time series. Specifically, D_i is a vector of such distances between a particular subsequence $T_{i,m}$ and each susequences in time series T written as $D_i = [d_{i,1},\ d_{i,2}, \ldots\ d_{i,n-m+1}]$ where $d_{i,j}$ is the distance between $T_{i,m}$ and $T_{j,m}$ and $1 \leq i, j \leq n - m + 1$. A matrix profile M, can be formally represented by $M = [min(D_1), min(D_2), \ldots, min(D_{n-m+1})]$ where $min(D_i)$ is the minimum Z-normalized euclidean value in D_i. This entire process has been visually depicted in Fig. 1.

To store the location of the nearest neighbour or the closest match, a companion data structure called matrix profile index denoted by I is used. It is represented as a vector of integers $I = [I_1,\ I_2,\ \ldots,\ I_{n-m+1}]$ where $I_i = j$ if $d_{i,j} = min(D_i)$. This helps in retrieving the nearest neighbour for each $T_{i,m}$ query by accessing the matrix profile index's i^{th} element.

To implement the concept of matrix profile, stump() method from the stumpy package [40] in python is used. The stump() method takes as input a time series T and the subsequence length m and provides a modified matrix profile $M_{mod} = [A_1,\ A_2,\ \ldots,\ A_{n-m+1}]$ as output, where $A_{i \in [1,\ 2,\ \ldots,\ n-m+1]} = \{min(D_i),\ I_i,\ IL_i,\ IR_i\}$. Here IL and IR are the left and right matrix profile indices which are explained in Sect. 3.2.

3.2 Time-series chains

Time-series chains [41] are informally considered as motifs [7] that evolve or drift in some direction over time. They

Table 1 Summary of granular and meta-learning based methods of time-series prediction proposed by researchers in recent decades

Author	Year	Method	Description
Dong et al. [25]	2008	Fuzzy Clustering based granular time-series prediction	Used fuzzy clustering to construct information granules that captures relations among them to construct a forecasting mechanism with further performance analysis on other such models
Froelich et al. [26]	2017	Fuzzy Cognitive Map (FCM) based granulation	Developed a forecasting model using FCM using a sequence of vectors of maximal degree of activation on a fuzzy set described by its bounds and modal values
Deng et al. [27]	2016	Multi-granularity combined multi-factor prediction model	Proposed a clustering algorithm to forecast fuzzy trends in different granular spaces using particle swarm techniques on real world datasets
Jana et al. [28]	2020	Maximal Overlap Dis-crete Wavelet Transformation (MODWT)	Proposed a granular deep learning approach with evaluation using Boruta algorithm-based feature selection model. Final prediction was obtained by aggregating the decomposed components implemented over real-world energy consumption datasets
Wang et al. [29]	2009	Forecasting methods based on selection rule	Used the trend, seasonality, periodicity, skewness, etc., as a meta-learning feature set and a derived weighting schema for improving forecasting accuracy
Gordon et al. [30]	2019	Meta-Learning approximate Probabilistic Inference for Prediction (ML-PIP)	Extended upon the existing probabilistic interpretations of meta-learning to cover a broad class of methods with few-shot learning datasets as inputs during model training
Yao et al. [31]	2019	Spatio-Temporal Prediction	A meta-learning paradigm that learns a generalized initialization of a Spatio-temporal approach was used to demonstrate long term prediction
Zhou et al. [32]	2012	Back Propagation Neural Network Model (BPNN)	Improved an EMD learning rate based model with a rating method to identify the best neural network model for the prediction of gold prices

have a property of directionality in them, as demonstrated in Fig. 2. To better understand the underlying working of time-series chains a few terms need to be considered.

The i^{th} left distance profile DL_i and i^{th} right distance profile DR_i of a time series T are vectors of euclidean distances between a particular subsequence $T_{i,m}$ and each subsequences that appear before and after $T_{i,m}$, respectively. The left nearest neighbour of $T_{i,m}$, $LNN(T_{i,m})$ and a right nearest neighbour $RNN(T_{i,m})$ are subsequences that appear before and after $T_{i,m}$, respectively. Formally $LNN(T_{i,m}) = T_{j,m}$ if $d_{i,j} = min(DL_i)$ and $RNN(T_{i,m}) = T_{j,m}$ if $d_{i,j} = min(DR_i)$. Vectors called left matrix profile ML and right matrix profile MR can be used to represent the Z-normalized euclidean distances between all subsequences and their left and right nearest neighbours, respectively. Formally they can be written as $ML = [min(DL_1),\ min(DL_2),\ ...,\ min(DL_{n-m+1})]$ and $MR = [min(DR_1),\ min(DR_2),\ ...,\ min(DR_{n-m+1})]$ where DL_i and $DR_i (1 \leq i \leq n - m + 1)$. Companion vectors called left matrix profile index IL and right matrix profile index IR stores the location of the left and right nearest neighbour for each i^{th} element in ML and MR. They can be formally denoted as $IL = [IL_i,\ IL_2,\ ...,\ IL_{n-m+1}]$ and $IR = [IR_i,\ IR_2,\ ...,\ IR_{n-m+1}]$ where $IL_i = j$ and $IR_i = j$ if $LNN(T_{i,m}) = T_{j,m}$ and $RNN(T_{i,m}) = T_{j,m}$, respectively.

In a time series T, a time-series chain is an ordered set of subsequences $TSC = \{T_{C1,m},\ T_{C2,m},\ ...\ T_{Cw,m}\}$ where indices $C1 \leq C2 \leq ... \leq Cw$. For any $1 \leq k \leq w - l$, we have $LNN(T_{C(k+1),m}) = T_{Ck,m}$ and $RNN(T_{Ck,m}) = T_{C(k+1),m}$. Here w denotes the length of the time-series chain.

This is implemented using the allc() function from the stumpy package [40] in python. The allc() method takes in the left matrix profile indices IL and right matrix profile indices IR as input and provides an all chain set $S = [S_1, S_2, ...S_r]$ which is a 2D-array as output. Here r is the total no. of time-series chains and for each i^{th} element $S_i = \{x_{j_1}, x_{j_2}, ...x_{j_{g_i}}\}$, here x_j is the starting index of a subsequence of i^{th} type of time-series chain and g_i is the total no. of elements in the i^{th} row.

4 Problem formulation and proposed approach

This section deals with the formulation of the problem statement along with the proposed approach used in this study.

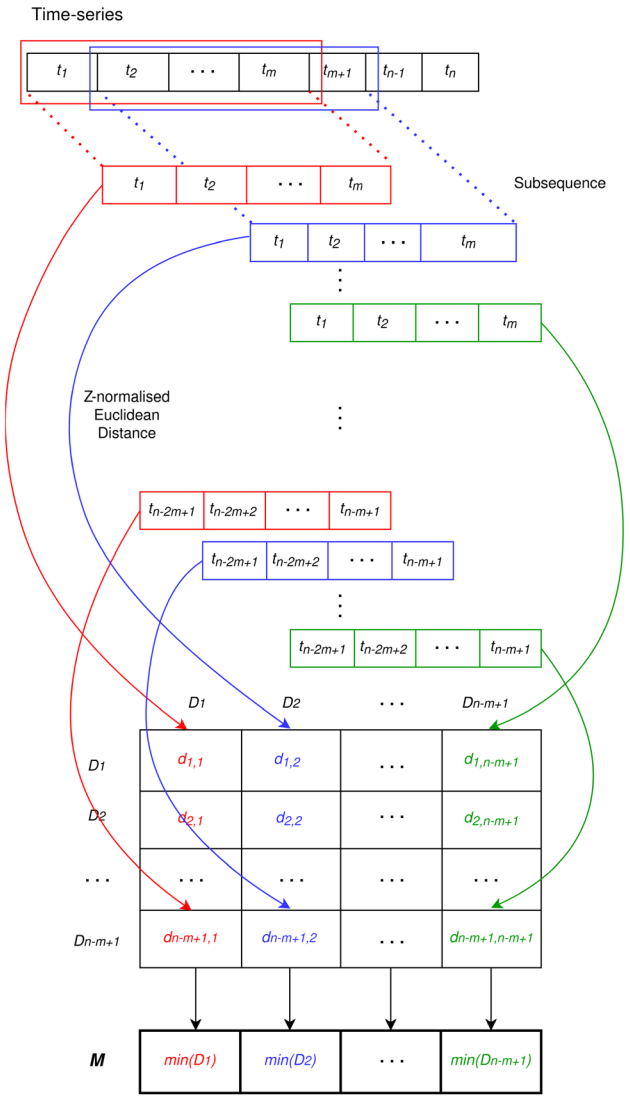

Fig. 1 Visual representation of matrix profile formation along with its relationship with the distance profile

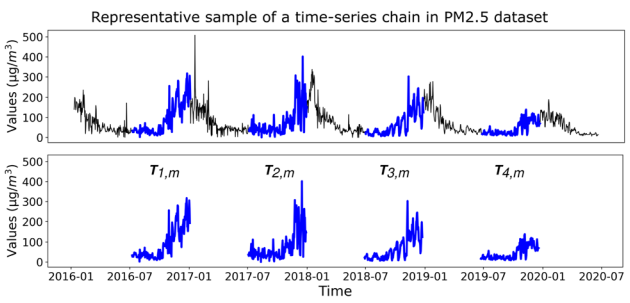

Fig. 2 Example of all subsequences falling under a particular type or label of time-series chain in PM2.5 dataset [42]. Here, different subsequences $T_{i,m}$ belongs to a specific label

4.1 Problem statement

Given a time series $P = \{v_1, v_2, \ldots, v_t\}$ over a duration of t timesteps in the past, our intention is to first label encode them from the original domain into $P_L = \{\omega_1, \omega_2, \ldots, \omega_{t/m}\}$ where corresponding to a subsequence of length m, label $\omega_k \in L = \{l_1, l_2, \ldots, l_n\}$ and L is the set of n different category of labels used. Based on the encoded time-series P_L prediction is made for next t'/m timesteps in the future to produce $P'_L = \{\omega'_{t/m+1}, \omega'_{t/m+2}, \ldots, \omega'_{(t+t)/m}\}$. Using $P_L{}'$ we intend to finally decode the time series into the original domain to produce predictions $P' = \{v'_{t+1}, v'_{t+2}, \ldots, v'_{t+t}\}$.

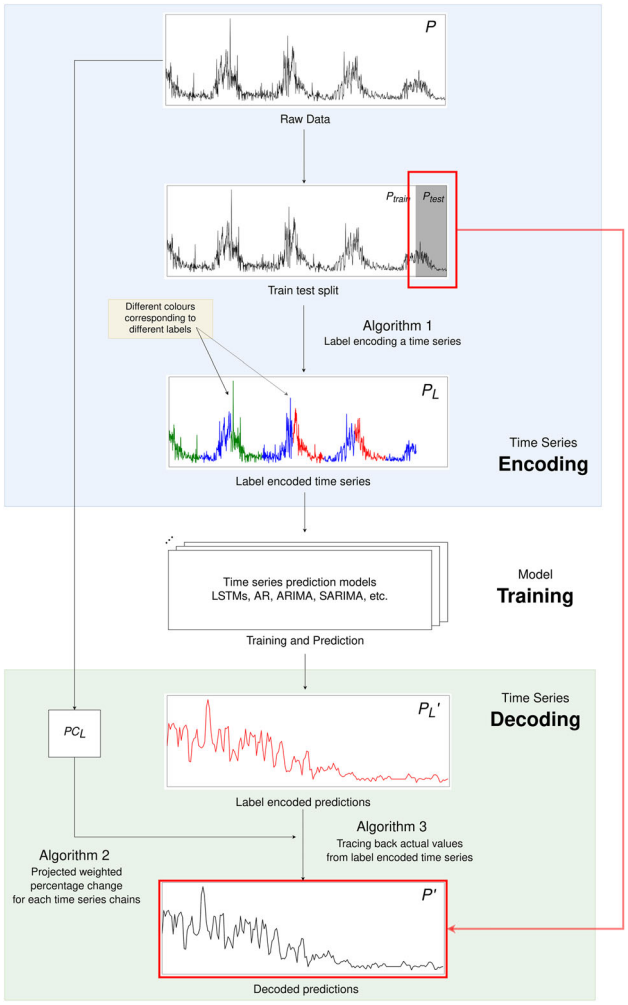

Fig. 3 General overview of the proposed approach using time-series labeling

4.2 Proposed approach

An overview of TSL is shown in Fig. 3. Here, the training part of the raw data is first encoded to a labeled time series and prediction is performed using various time-series prediction models. Decoding is then performed on labeled predictions 'P_L using the projected weighted percentage changes of previously occurred time-series chains corresponding to the predicted labels, to finally produce 'P the decoded predictions.

To perform the conversion from P to P_L, stump() [40] method is used for finding out the matrix profile of the time series. Its output is used by the allc() [40] method to categorize the different time-series chains formed along with finding the starting indices of subsequences of each label. This implementation is described in detail in Algorithm 1. Prediction is made using the label encoded time series P_L for the next t'/m timesteps by various time-series prediction models like LSTMs, AR, ARIMA and SARIMA to produce P'_L.

In order to decode P'_L, a two step process is used where first the projected weighted percentage change $PC_L = \{X_{l_1}, X_{l_2}, \ldots, X_{l_n}\}$ is found out for each category of labels where L represents the set of labels and X_i represents a particular label type. This is implemented using Algorithm 2. An assumption is undertaken that the predicted labeled subsequence will be most similar to the last occurring subsequence of the time-series chain corresponding to the same label, thus weights are distributed accordingly in a decreasing manner (Line 14 and 15 of Algorithm 2). For the second part, Algorithm 3 is used to decode P'_L by multiplying the projected weighted percentage change X_i with values from the last occurring subsequence in P corresponding to ω'_i. Finally to evaluate the performance of the method, metrics like root-mean-square error and mean absolute error are calculated and compared.

Algorithm 1: LABEL ENCODING A TIME-SERIES

Input: $P = \{v_1, v_2, \ldots, v_t\}, m$
Output: $P_L = \{\omega_1, \omega_2, \ldots, \omega_{t/m}\}$

1 $IL, IR \leftarrow stump(P, m)$
2 $S \leftarrow allc(IL, IR)$
3 $pred \leftarrow \{v_1, v_2, \ldots, v_t\}$
4 $r \leftarrow len(S)$
5 **for** $i \leftarrow 1$ ***to*** r **do**
6 $g_i \leftarrow len(S_i)$
7 **for** $j \leftarrow 1$ ***to*** g_i **do**
8 $x_{j_i} \leftarrow S_{i,j}$
9 **for** $k \leftarrow x_{j_i}$ ***to*** $x_{j_i} + m$ **do**
10 $v_k \leftarrow i$
11 **return** *pred*

Algorithm 2: PROJECTED WEIGHTED PERCENTAGE CHANGE FOR EACH TIME-SERIES CHAINS

Input: $S = [S_1, S_2, \ldots S_r], P = \{v_1, v_2, \ldots, v_t\}, m$
Output: $PC_L = \{X_{l_1}, X_{l_2}, \ldots X_{l_n}\}$

1 $finalChange \leftarrow \{\}$
2 $r \leftarrow len(S)$
3 **for** $i \leftarrow 1$ ***to*** r **do**
4 $sMed = \{\}$
5 $g_i \leftarrow len(S_i)$
6 **for** $j \leftarrow 1$ ***to*** g_i **do**
7 $med \leftarrow median(v_j, v_{j+1}, \ldots, v_{j+m})$
8 $sMed \leftarrow append(med)$
9 $percentChange = \{\}$
10 **for** $j \leftarrow 1$ ***to*** g_i **do**
11 $percent \leftarrow 100 \times (sMed_{j+1} - sMed_j)/sMed_{j+1}$
12 $percentChange \leftarrow append(percent)$
13 $n \leftarrow len(percentChange)$
14 $w = \frac{1}{\sum_{i=1}^{n} \frac{1}{2^{n-i}}}$
15 $weights \leftarrow \{w/2^{(n-1)}, w/2^{(n-2)}, \ldots w\}$
16 $wpe \leftarrow 0$
17 **for** $j \leftarrow 1$ ***to*** n **do**
18 $wpe \leftarrow wpe + weights_j * percentChange_j$
19 $finalChange \leftarrow append(wpe)$
20 **return** *finalChange*

Algorithm 3: TRACING ACTUAL VALUES FROM LABEL ENCODED TIME-SERIES

Input: $P_L{}' = \{\omega'_{t/m+1}, \omega'_{t/m+2}, \ldots, \omega'_{(t+t')/m}\}$,
$PC_L = \{X_{l_1}, X_{l_2}, \ldots X_{l_n}\}, P = \{v_1, v_2, \ldots, v_t\}$
Output: $P' = \{v'_{t+1}, v'_{t+2}, \ldots, v'_{t+t'}\}$

1 $predictions \leftarrow \{\}$
2 **for** $i \leftarrow t+1$ ***to*** $t+t'$ **do**
3 $label \leftarrow \omega'_{ceil(i/m)}$
4 $lastval \leftarrow getTSCLastVal(P, label)$
5 $prediction \leftarrow lastval \times X_{label}$ // $X_{label} \in PC_L$
6 $predictions \leftarrow append(prediction)$
7 **return** *predictions*

4.3 Models

To compare the performance of our proposed approach, different sequence learner models like LSTM, AR, ARIMA and SARIMA with and without time-series labeling (TSL) as a pre and post processing step are used to implement the same exact task of time-series prediction over a fixed period of time.

LSTM [3] networks are a special kind of recurrent neural networks (RNN) especially designed to remember information for longer periods of time. They are explicitly engineered to counter the problem of vanishing gradient, unlike RNNs which are very eaisily affected by it. LSTMs comprise of four interacting layers in their repeating modules compared to one in RNNs.

Mathematically the layers of an LSTM network can be expressed as follows: 1–6:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad (1)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \quad (2)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \quad (3)$$

$$\tilde{c_t} = \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \quad (4)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c_t} \quad (5)$$

$$h_t = o_t \circ \sigma_h(c_t) \quad (6)$$

where f_t, i_t, o_t and $\tilde{c_t}$ are the activation vectors of forget gate, input gate, output gate and cell input gate, respectively. c_t and h_t are the cell state and hidden state vectors. x_t is the input state vector. Matrices of the form W_q and U_q, respectively, contain the weights of the input and recurrent connections. In activation functions, σ_h denotes the hyperbolic tangent function while σ_c denotes the sigmoid function.

An AR model [4] is a statistical approach to time-series prediction which takes an input of the previous observations and predicts the values at the next time step. The model can be described as below:

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t \quad (7)$$

where $\phi_1, \phi_2, \ldots, \phi_p$ are the parameters of the model, c is the constant term and ϵ_i is the noise term.

An ARIMA Model [4] consists of an auto-regressive (AR), integrated (I) and a moving average (MA) component to better understand a time-series data or to predict a future time-series data. The AR component shows that the evolving variable of interest is regressed on its own lagged values. Replacement of the present values and their previous values are indicated by the I component. The MA component indicates that the regression error is a linear combination of error terms that occurred in the past. The ARIMA model can be expressed as following equation 8.

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right)(1 - L)^d X_t = \delta + \left(1 + \sum_{i=1}^{q} \theta_i L^i\right)\varepsilon_t \quad (8)$$

where p, d and q denote the time lags of the AR component, the degree of differencing and the order of the MA component, respectively.

The seasonal ARIMA Model (SARIMA) [5] is an extension of the ARIMA Model, with additional seasonal component along with the existing AR, I and MA terms as well a periodic term denoted by m.

4.4 Data modeling

For each model train-test split was performed on the data keeping the size of the test set uniform in all cases which was taken to be 180 timesteps, this was specifically ensured to provide a uniform base for comparison between different methods across various datasets.

The subsequence length m used to label the time series in Sect. 4.2 was chosen to be 180 after several experiments implementing a range of values from 30 to 360 timesteps. In the course of all the experiments which were performed, 180 timesteps provided an overall least RMSE over the final test set results throughout all the models.

5 Evaluation and results

In this section, the evaluation setup, description of datasets used along with the the model performance of different statistical and deep learning-based methods with and without TSL are compared and discussed in great detail.

5.1 Experimental setup

The entire study had been carried out on Google Colab [45, 46]. A virtual instance containing 2 single core hyper threaded Intel Xeon CPU Processor clocked at 2.3 Ghz coupled with 13 GB DDR4 RAM was used for carrying out the mathematical computations. For deep learning purposes, a 12 GB NVIDIA Tesla K80 GPU was also enabled by Google Colab as a hardware accelerator to speed up matrix related calculations in deep learning methodologies. The developmental code for this study was based on python [47], due to the presence of good high-end libraries like numpy [48], tensorflow [49], statsmodels [50] and scikit learn [51] to help in decreasing the overall complexity of the code without compromising in efficiency and performance.

5.2 Data description

Five real-world datasets had been used to demonstrate our proposed approach and its effectiveness to tackle different scenarios.

The particulate matter PM2.5 and PM10 datasets were of Kolkata, India and were provided by the central pollution control board (CPCB) [42], the statutory organisation responsible for providing field information regarding pollution of various places throughout the country. The monitoring stations were positioned at Victoria Memorial Hall (22.5448° N, 88.3426° E), supplemented with data procured from other nearby stations. Preliminary analysis

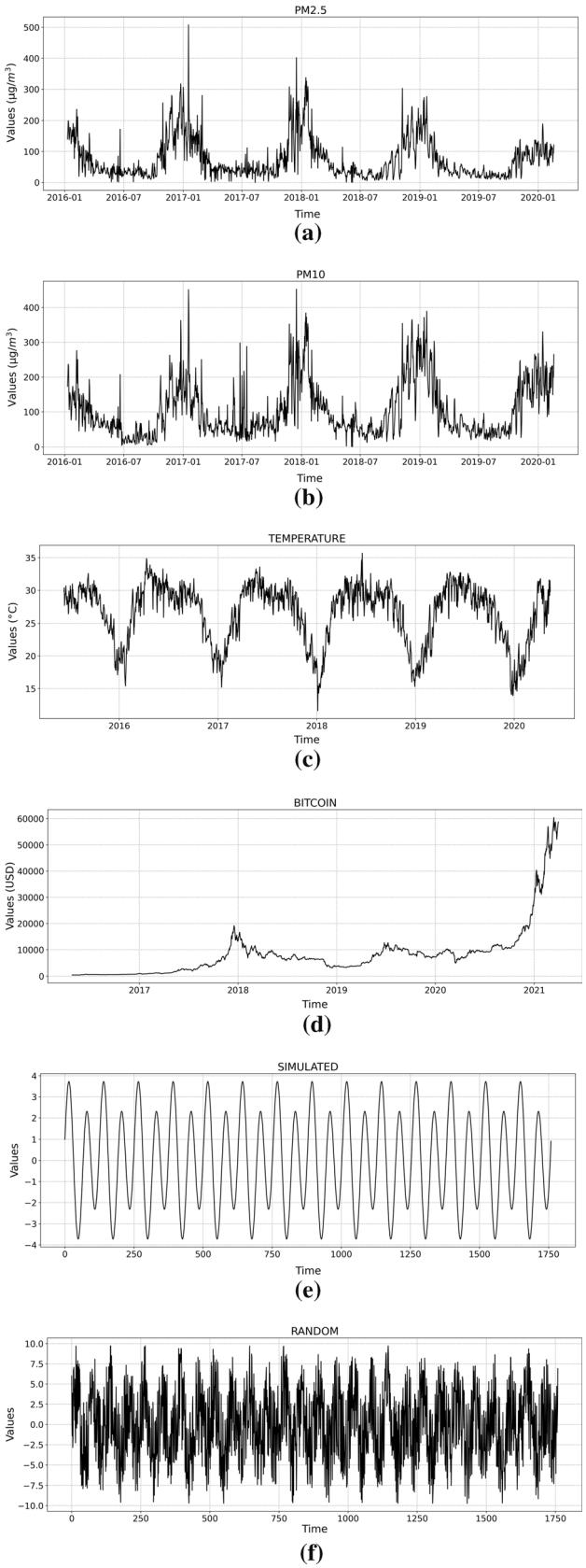

Fig. 4 **a** PM2.5 [42], **b** PM10 [42], **c** Temperature [43], **d** Bitcoin [44], **e** Simulated and **f** Random datasets used in the study

of the data led to finding it being daily in nature, spanning almost four years from 10th January 2016 to 18th February 2020 shown in Fig. 4a, b. Due to external factors such as hardware failure, maintenance operations, etc. Large chunks of the raw data extracted were found to be missing. Hence, these values were needed to be either found out from other external sources or have to be internally imputed using various techniques. These missing daily PM2.5 and PM10 values were extracted from the US Department of State's AirNow [52] web portal.

The Temperature dataset [43] used for the study was gathered from Kaggle [53] and was also of Kolkata, India. The daily data spanned over the time period from 11 June, 2015 to 13 May, 2020 as shown in Fig. 4c. This dataset was a subset of the global temperature values originally from the University of Dayton's Temperature [54] archive.

To test on how TSL methods perform on on-periodic datasets, historical price data of the most popular cryptocurrency Bitcoin [44] was used. The time-series data were gathered again from Kaggle [53] and it consists of the daily prices from a period of 27 April, 2016 to 31 March, 2021 in USD as shown in Fig. 4d. The dataset was originally derived from various exchange APIs for the OHCL (Open, High, Low, Close) prices of Bitcoin in a minute to minute update interval.

To gauge the efficacy when fed data having extreme characteristics, two additional datasets were also used. A perfect simulated periodic dataset generated using the equation $f(t) = 3\sin(t) + \cos(t/2)$ over a period of 56π and 0.1 interval as represented in Fig. 4e was used to signify perfect conditions. On the other hand, to simulate the worst possible condition, a randomly generated dataset using the equation $f(t) = \varepsilon_t$ was also included in our study as shown in Fig. 4f.

Figure 4 represents a visual depiction of the raw data for each of the datasets discussed above.

5.3 Parameter setting

In case of LSTMs, a stacked LSTM structure was used along with grid search on training epochs ranging till 150, batch size ranging till 64 and number of LSTM layers till 10 to find the best parameters according to the performance metric of least RMSE. To maintain a fair comparison of evaluation of models with/without TSL, same parameters were maintained for both. The hyper-parameters obtained during grid search on a limited search space of lags giving the least RMSE were implemented in the AR model.In case of AR, ARIMA and SARIMA, the statespace implementation of a python [47] package called statsmodels [55] was used. It took in parameters of maximum time lags of the AR auto-regressive component p, maximum degree of differencing d, maximum order of the (MA) moving-

average component q and m as the number of observations per seasonal cycle. Maximum value for each p, d and q was set to 7 based on trial and error to find the least possible RMSE values on the test set predictions. m too was also set to 7 as most of the datasets were daily in nature and to maintain uniformity while comparing. The seasonal component in the SARIMAX method was switched on and off while implementing ARIMA and SARIMA, respectively.

All the above parameter searching methodologies were implemented and repeated for each dataset individually (Fig. 5).

5.4 Evaluation metrics

Performance different deep learning and statistical methods used with/without TSL is compared on the basis of root-mean-squared error (RMSE) and mean average error (MAE) in contrast to the test set mean. Since each error is proportional to the size of the squared error, thereby causing larger errors to disproportionately have a larger effect on the RMSE. The test set mean is the mean of all the values present in the testing set, thus helping in providing a perspective of the errors made during prediction.

In addition, relative improvements in performance of all the methods on a particular dataset, were also studied and

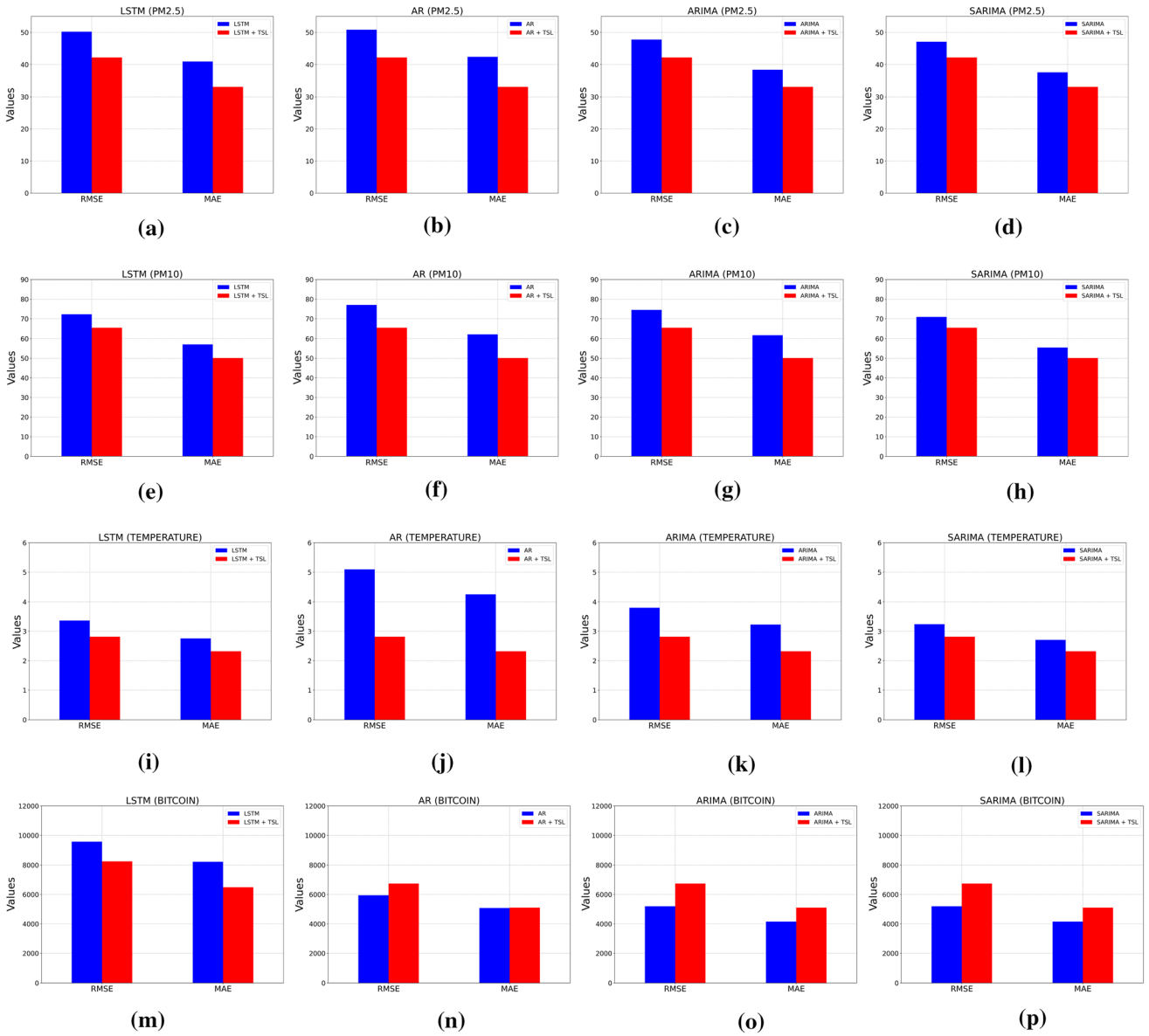

Fig. 5 Performance of LSTM, AR, ARIMA and SARIMA with and without TSL in terms of RMSE and MAE implemeted on **a–d** PM2.5 [42], **e–h** PM10 [42], **i–l** Temperature [43] and **m–p** Bitcoin [44] datasets

displayed in the form of bar plots in Fig. 13 to analyse the margin by which each method outperformed the others when compared with the worst performing ones.

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}}, \text{MAE} = \frac{\sum_{t=1}^{T}|\hat{y}_t - y_i|}{T} \quad (9)$$

where $\hat{y}_t$ is the prediction made by the model and y_t is the actual value at instant t. Here T denotes the count of the number of time-series samples (Fig. 6).

5.5 Results

To justify the effectiveness of TSL in improving the performance when coupled with various models like LSTMs, AR, ARIMA and SARIMA for time-series prediction, comparisons are drawn between model combinations with and without our proposed combination of pre and post processing steps. These series of steps are repeated for each dataset.

In case of the PM2.5 dataset, the largest increase in performance occurred on using TSL coupled with auto-regressive (AR) model (TSL + AR), decreasing the RMSE from 50.854 to 42.202 and decreasing the MAE from 42.404 to 33.131 shown in Fig. 5b. This resulted in a greater positive correlation between actual versus predicted values shown in Fig. 7f compared to Fig. 7b where solely AR model was used without TSL for the same task. The worst performance of sole AR can be attributed to it having only a single auto-regressive component which struggles to find periodicity in the dataset. Overall, the use of TSL resulted in almost 10–17% decrease in RMSE and 10–22% decrease in MAE across all the models compared to not using TSL as depicted in Fig. 13a. The performance of all the models in combination with TSL was similar.

For PM10 dataset, again TSL + AR showed the greatest improvement in performance compared to AR with RMSE value decreasing from 77.08 to 65.44 and MAE value decreasing from 62.213 to 50.138 depicted in Fig. 5h, across a test set mean of 136.642. Thereby leading to a greater positive correlation between actual and predicted values shown in Fig. 8f. An almost 7.5–15% decrease in RMSE and 8–19% decrease in MAE (as depicted in Fig. 13b) can be seen across all the models when coupled with TSL compared to not using TSL's pre and post processing steps. It can be inferred through Fig. 13b that, solely used AR model performed the worst and even though LSTM being a deep learning model having greater hyper-parameters, a statistical model like SARIMA outperformed it. Thus indicating the efficacy of simpler models that take into account parameters like seasonality, the time lag, degree of differentiation and order of the data. Lastly, here it can also be seen that combining TSL with different models show a similar performance.

In the case of the Temperature dataset that out our approach resulted in 10–45% decrease in both RMSE and MAE when implemented. It can be seen from Fig. 5i–l that SARIMA model beat LSTM in both the evaluated metrics, even though it having lesser components for data modeling. The deep learning and statistical models used without TSL could show a defined positive correlation of the actual versus predicted values as depicted through the scatter plots in Fig. 9a–d but combining with TSL lead to a much better, defined correlation, seen in Fig. 9e–h. The

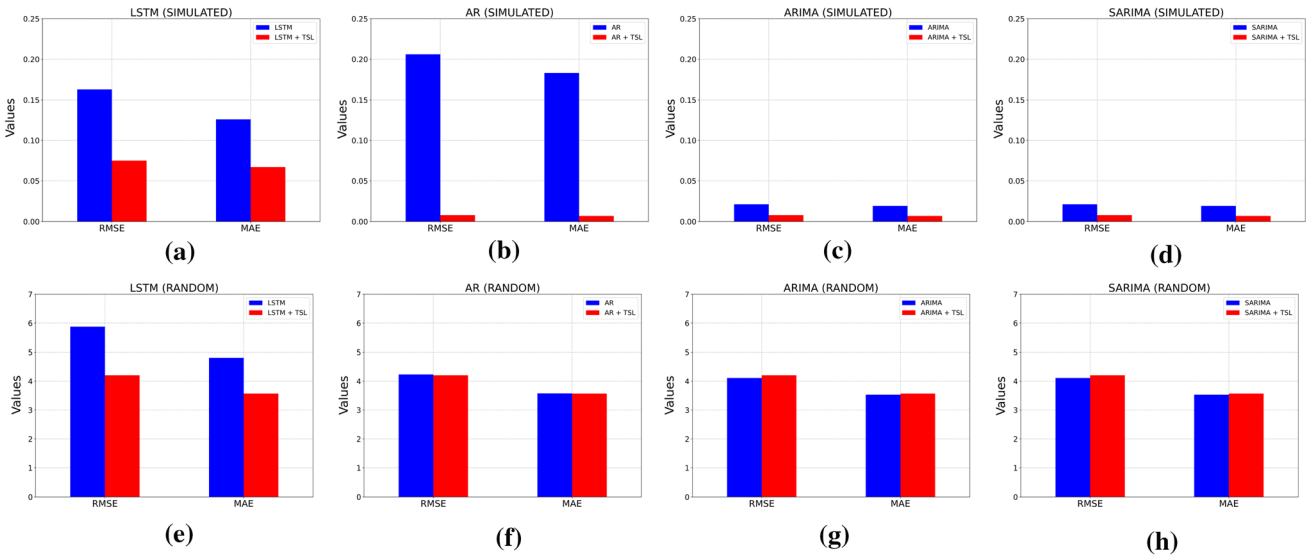

Fig. 6 Performance of LSTM, AR, ARIMA and SARIMA with and without TSL in terms of RMSE and MAE implemeted on Simulated (**a**–**d**) and Random (**e**–**h**) datasets

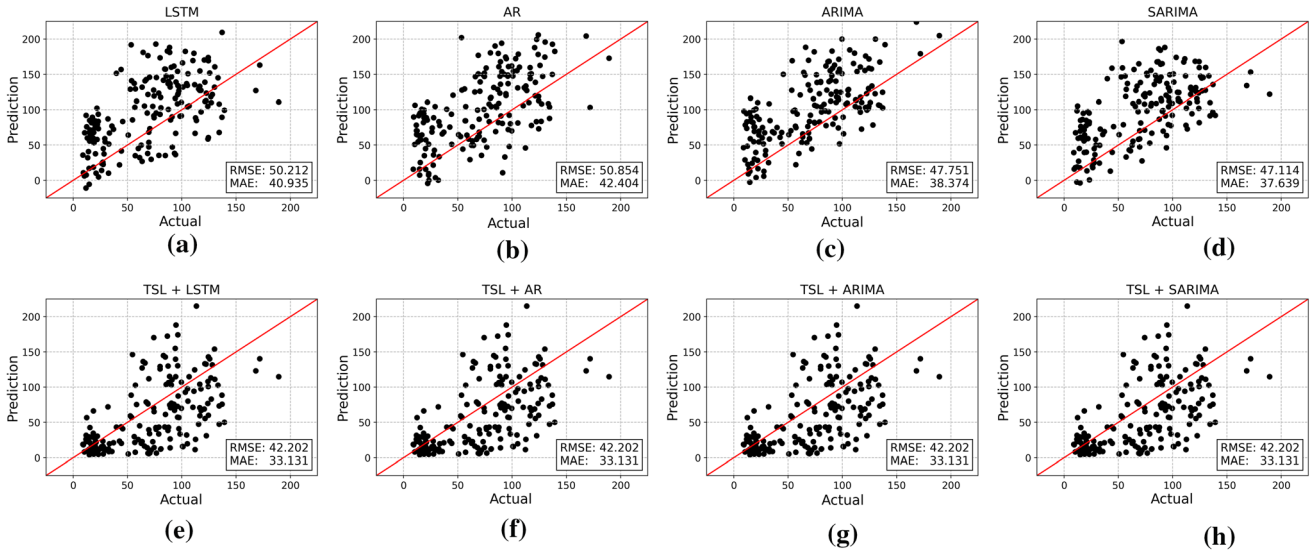

Fig. 7 Actual versus prediction correlation plots for different deep learning and statistical methods (**a**–**d**) without TSL and (**e**–**h**) with TSL on PM2.5 dataset [42]

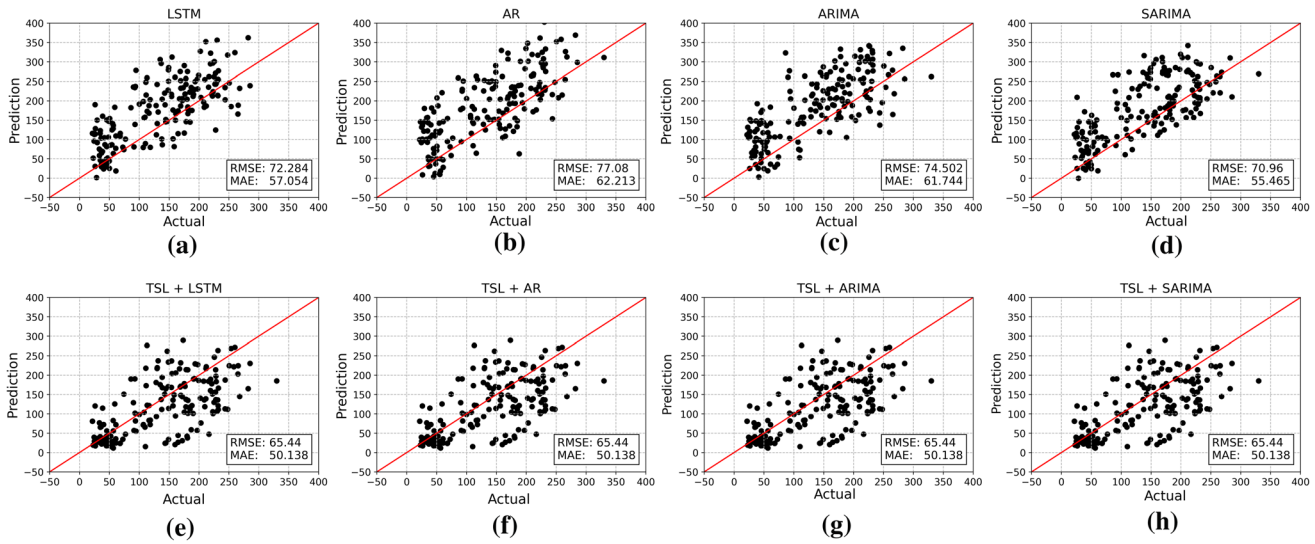

Fig. 8 Actual versus prediction correlation plots for different deep learning and statistical methods (**a**–**d**) without TSL and (**e**–**h**) with TSL on PM10 dataset [42]

performance of all the models in combination with TSL was similar.

In case of the Bitcoin dataset, a dissimilarity in performance of some of the models in combination with TSL can be seen. Even models performing solely without TSL can be seen performing better. ARIMA and SARIMA demonstrated the least RMSE and MAE values of 5198.992 and 4159.013, respectively, compared to all the other models. The same performance of ARIMA and SARIMA in Fig. 10 can be attributed to the lack of periodicity in the dataset. TSL was able to increase the performance only when coupled with LSTM as depicted in Fig. 10a, e. A relative performance improvement of 45% as depicted in Fig. 13d compared to the LSTM model, which performed the worst out of all the methods.

For the simulated periodic dataset, a massive increase in performance can be seen, as compared to the worst performing model (AR) through Fig. 6a–d where TSL is coupled with AR, ARIMA and SARIMA, led to a big jump in RMSE from 0.206 to 0.008 and MAE from 0.183 to 0.007 when put in comparison with sole AR. A significant increase in positive correlation can also be observed when comparing each scatter plot in Fig. 11a, b of models being used solely compared to plots in Fig. 11f–h where TSL is used in combination. This big increase in performance can solely be attributed to the simulated dataset being perfectly

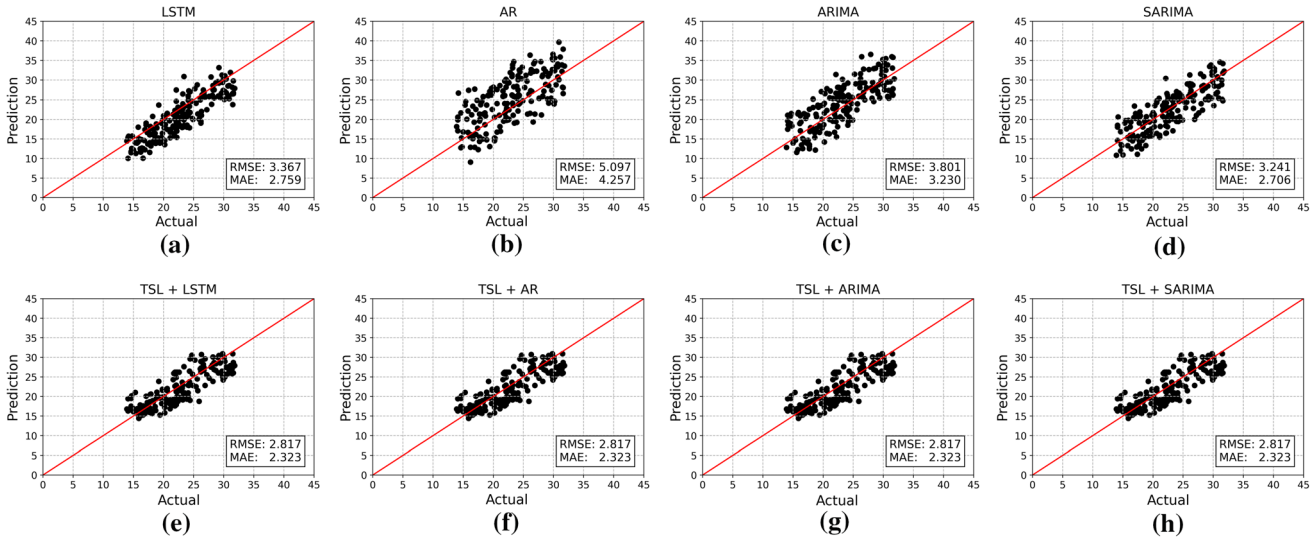

Fig. 9 Actual versus prediction correlation plots for different deep learning and statistical methods (**a**–**d**) without TSL and (**e**–**h**) with TSL on Temperature dataset [43]

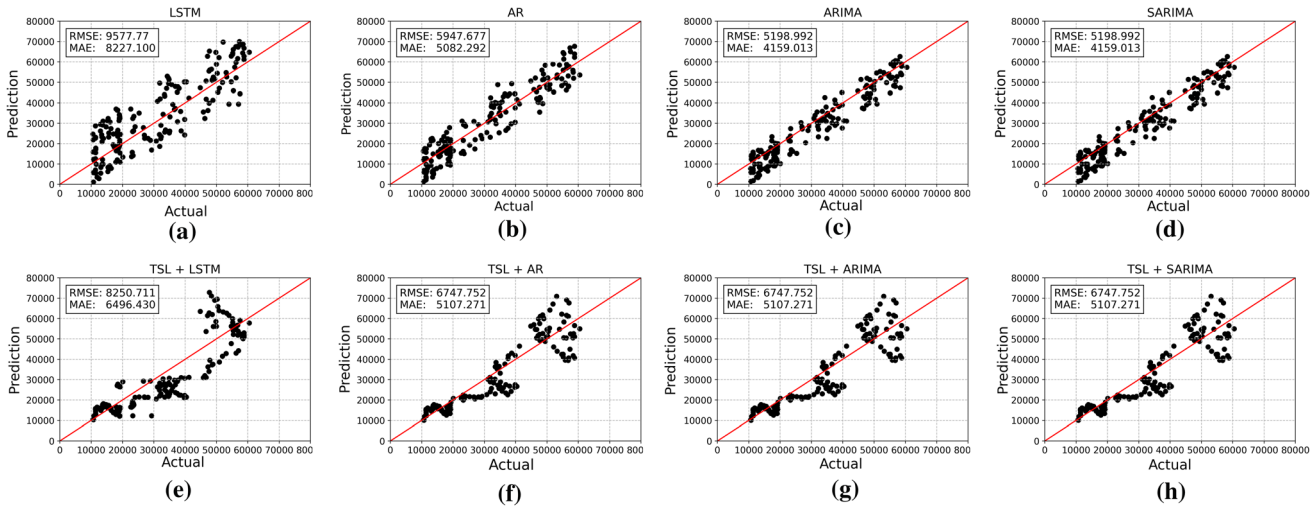

Fig. 10 Actual versus prediction correlation plots for different deep learning and statistical methods (**a**–**d**) without TSL and (**e**–**h**) with TSL on Bitcoin dataset [44]

periodic bt nature. In Fig. 6b the actual and predicted values over the test-set are phase-shifted, thus resulting in a peculiar radially distributed scatter plot. It can be observed through Fig. 13e that combining TSL with models, leads to a 5–92% increase in performance. Other than TSL + LSTM, the performance of all models in combination with TSL were found to be similar. ARIMA and SARIMA not only outperformed LSTM but (TSL + LSTM) too, thus showing their efficacy over perfectly periodic datasets.

In case of the pure Random dataset, although having erroneous intermediate encodings, an expected behaviour of convergence to the mean is witnessed in the models which are used without TSL. But a more defined correlation between actual and predicted values can be observed in all the TSL combination models show in Fig. 12e–h. Not all models used with TSL could perform better in terms of RMSE and MAE metrics, as shown in Fig. 13f.

5.6 Discussion

For each real world or generated dataset used in this study which had some degree of periodicity, the proposed method of TSL was successfully able to increase the performance (as evident from Figs. 5a–l and 6a–d), both in terms of RMSE and MAE metrics when coupled with time-series predictive models thus demonstrating its capability to predict time-series data more effectively. Therefore, combinations involving TSL which produce comparatively

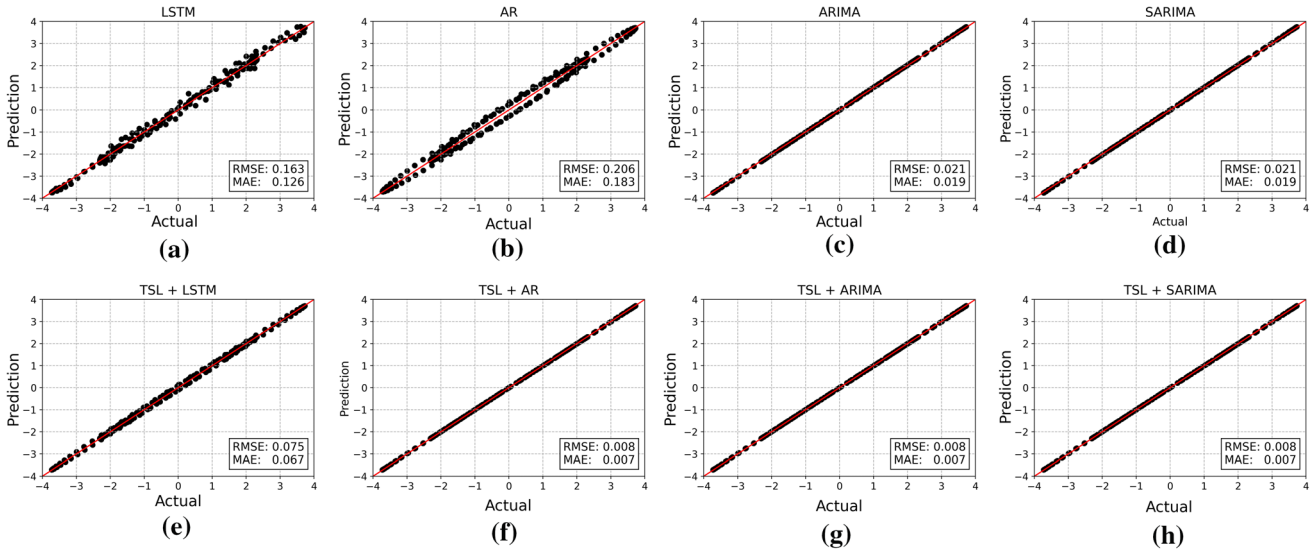

Fig. 11 Actual versus prediction correlation plots for different deep learning and statistical methods (**a**–**d**) without TSL and (**e**–**h**) with TSL on Simulated dataset

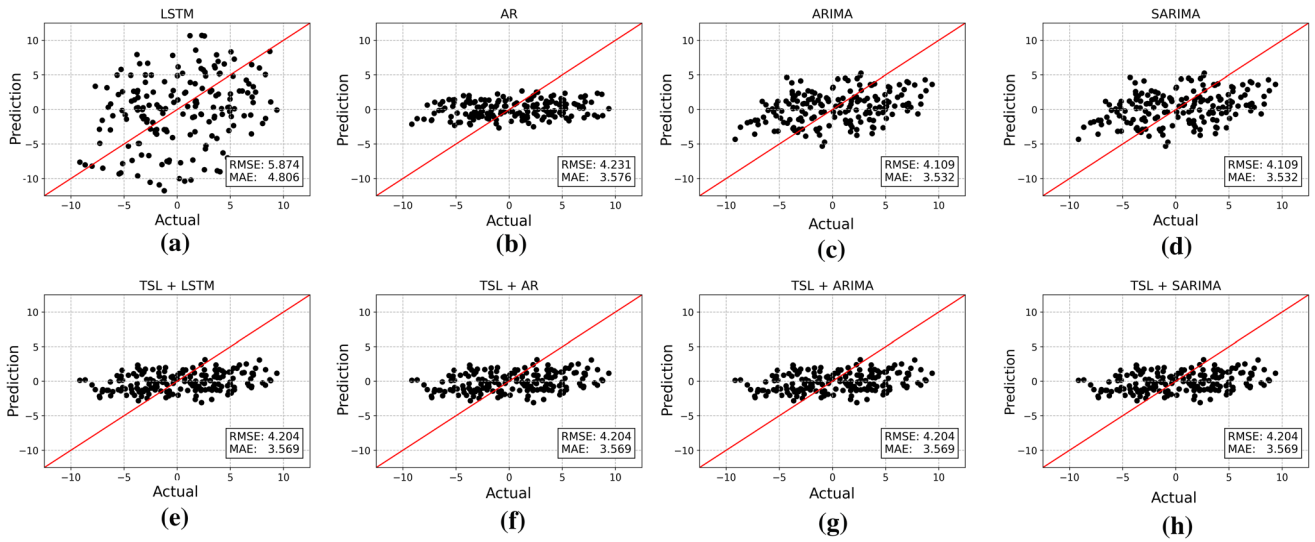

Fig. 12 Actual versus prediction correlation plots for different deep learning and statistical methods (**a**–**d**) without TSL and (**e**–**h**) with TSL on Random dataset

better performance showcase their capability in successfully learn repeating trends in data. This capability of producing effective predictions is also evident from the positive correlation of the actual versus predicted values in each of the scatter plots (e–h) for Figs. 7, 8, 9 and 11.

The Bitcoin dataset is a real-world dataset having non-periodic attributes, therefore TSL which takes in factors like periodicity and repeating trends in the data was not able to increase the performance of each model, compared to the models being used solely. Similarly, due the Random dataset being a generated random-walk, had no defined periodicity in it, therefore coupling TSL to models didn't lead to an improvement in performance in each model. Contrary to these two datasets, the Simulated dataset had a pre-defined periodicity in it, thus leading TSL to learn better from its trends. Moreover, resulting in a performance increase of maximum 95% shown in Fig. 13e which is the greatest increase in performance across all datasets compared to using a model solely.

Compared to using ordinary time-series prediction models, models coupled with TSL witness an increase in performance in the range of 7.5–85% and 11.5–85% for RMSE and MAE, respectively. This can be attributed to the feature of TSL effectively capturing the repeating trends in a time series, facilitating a model's learning process to capture information from previously occurred events, even

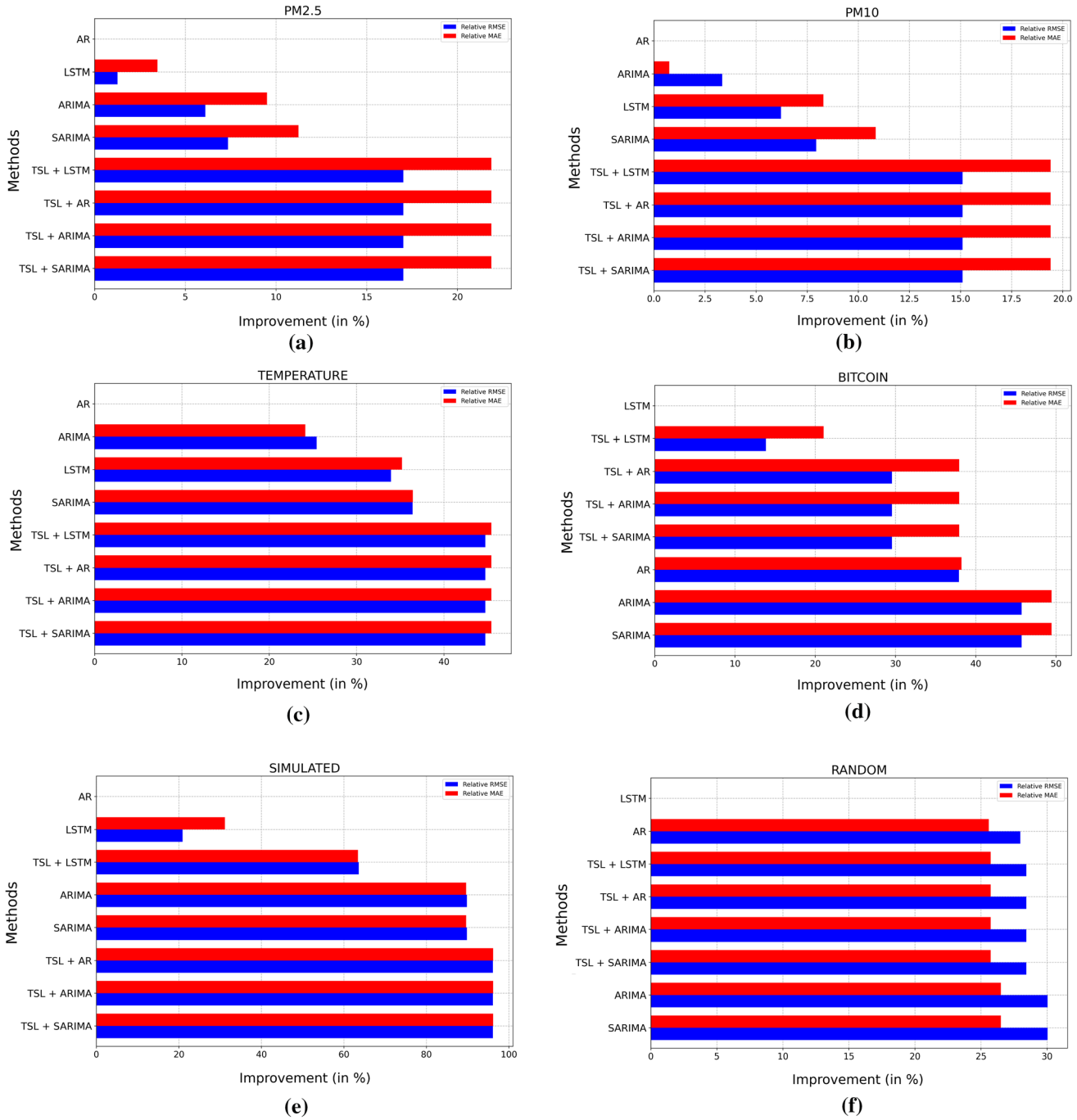

Fig. 13 Relative performance improvement in RMSE and MAE w.r.t. worst performing models over all the datasets used in the study

with external factors like chaotic nature of data and amount of data affecting it such as in case of real-world scenarios. From the experiments performed, TSL is found to effectively counter these points of failure along with the added benefit of finding varying types of periodically repeating trends (motifs) of granular time-series subsequences.

Upon carefully inspecting Figs. 5 and 6, an interesting observation in case of the versions of SARIMA without TSL can be noted. Even though having lesser parameters than its deep-learning peer like LSTM, SARIMA outperforms them in all datasets used, with an average of 25% increase in performance. This clearly shows that the added complexity of deep learning models seemed to have decreased the model efficiency and simpler statistical models are found to not only perform better than more complex deep learning ones but also has an added benefit of overcoming the chaotic nature of daily and hourly real-world datasets. Even simpler models like AR in

combination with TSL with lesser runtime and system requirements can be preferred compared to a deep learning models like LSTMs which are prone to overfitting [8], get affected by random weight initializations [3] and require huge amounts of data for accurate predictions.

Finally, the similar performance of LSTM, AR, ARIMA and SARIMA models when used in combination with TSL over PM2.5, temperature and simulated datasets can be attributed to them being able to correctly identify the label the test set falls under. This model behaviour is caused due to them accurately predicting the time-series labels thus resulting in similar RMSE and MAE uponon reconstruction.

6 Conclusion

In this study, a novel time-series labeling (TSL) method is proposed which takes into account the periodic trends in a time series, therefore, mimicking a model's learning process and when coupled with deep learning models like LSTMs and statistical models like AR, ARIMA and SARIMA, can improve their performance substantially. The use of a wide variety of real world and generated datasets with some being periodic and some non-periodic in nature makes the study more thorough and robust.

A certain shortcoming of this proposed method lies in it not being able to model non-periodic time-series data as efficiently compared to periodic time-series data. In such cases it may not always provide better results than using ordinary models without TSL for time-series prediction, which is demonstrated through the use of the Bitcoin and Random datasets.

As TSL is fundamentally a feature engineering step, it can be coupled with time-series models like auto-encoder, bidirectional and convolution LSTMs or even with different statistical models thus adding to its flexibility and providing room for further study in this field. Also, TSL is quite effective in democratising performance regardless of model complexity thus making simpler models like AR produce comparable predictions with respect to more complex ones like LSTMs.

This study in its current form has not taken into account the use of exogenous variables. Although models like AR, ARIMA and SARIMA does not have the required flexibility, its extended variants like ARIMAX and SARIMAX and LSTMs can be used in such a case to give much more accurate results.

Acknowledgements The research work of Asif Iqbal Middya is funded by "NET-JRF (National Eligibility Test-Junior Research Fellowship) scheme of the University Grants Commission, Government of India". This research work is also supported by the project entitled "Participatory and Realtime Pollution Monitoring System For Smart City, funded by Higher Education, Science & Technology and Biotechnology, Department of Science & Technology, Government of West Bengal, India".

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

References

1. Das R, Middya AI, Roy S (2021) High granular and short term time series forecasting of pm2.5 air pollutant - a comparative review. Artif Intell Rev. https://doi.org/10.1007/s10462-021-09991-1
2. Middya AI, Roy S, Das R (2021) Spatiotemporal variability analysis of air pollution data from iot based participatory sensing. J Ambient Intell Humaniz Comput. https://doi.org/10.1007/s12652-021-03536-8
3. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput. https://doi.org/10.1162/neco.1997.9.8.1735
4. Box GEP, Jenkins GM, Reinsel GC, Ljung GM (2015) Time series analysis: Forecasting and control, 5th edn. Wiley
5. Chang X, Gao M, Wang Y, Hou X (2012) Seasonal autoregressive integrated moving average model for precipitation time series. J Math Stat 8:500–505. https://doi.org/10.3844/jmssp.2012.500.505
6. Al-Hmouz R, Pedrycz W, Balamash A (2015) Description and prediction of time series: a general framework of granular computing. Expert Syst Appl 42:1. https://doi.org/10.1016/j.eswa.2015.01.060
7. Chiu B, Keogh E, Lonardi S (2003) Probabilistic discovery of time series motifs. KDD '03. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/956750.956808
8. Caruana R, Lawrence S, Giles L (2000) Overfitting in neural nets: backpropagation, conjugate gradient, and early stopping. In: Proceedings of the 13th international conference on neural information processing systems, NIPS'00. MIT Press, pp 381–387
9. Leite D, Škrjanc I (2019) Ensemble of evolving optimal granular experts, owa aggregation, and time series prediction. Inform Sci 504:95–112. https://doi.org/10.1016/j.ins.2019.07.053
10. Shao K, Zheng J, Wang H, Xu F, Wang X, Liang B (2021) Recursive sliding mode control with adaptive disturbance observer for a linear motor positioner. Mech Syst Signal Process 146:107,014. https://doi.org/10.1016/j.ymssp.2020.107014
11. Shao K, Zheng J, Wang H, Wang X, Lu R, Man Z (2021) Tracking control of a linear motor positioner based on barrier function adaptive sliding mode. IEEE Trans Ind Inf 17(11):7479–7488. https://doi.org/10.1109/TII.2021.3057832
12. Shao K (2021) Nested adaptive integral terminal sliding mode control for high-order uncertain nonlinear systems. Int J Robust Nonlinear Control 31(14):6668–6680. https://doi.org/10.1002/rnc.5631
13. Afolabi D, Guan SU, Man KL, Wong PWH, Zhao X (2017) Hierarchical meta-learning in time series forecasting for improved interference-less machine learning 9(11):1. https://doi.org/10.3390/sym9110283
14. Nath P, Saha P, Middya AI, Roy S (2021) Long-term time-series pollution forecast using statistical and deep learning methods. Neural Comput Appl 33(19):12551–12570. https://doi.org/10.1007/s00521-021-05901-2

15. Middya AI, Roy S, Dutta J, Das R (2020) JUSense: a unified framework for participatory-based urban sensing system. Mob Networks Appl 25(4):1249–1274. https://doi.org/10.1007/s11036-020-01539-x
16. Dutta J, Chowdhury C, Roy S, Middya AI, Gazi F (2017) Towards smart city: sensing air quality in city based on opportunistic crowd-sensing. In: Proceedings of the 18th international conference on distributed computing and networking, pp. 1–6. https://doi.org/10.1145/3007748.3018286
17. Middya AI, Roy S (2021) Spatial interpolation techniques on participatory sensing data. ACM Trans Spat Alg Syst 7(3):1–32
18. Kar D, Middya AI, Roy S (2019) An approach to detect travel patterns using smartphone sensing. In: IEEE international conference on advanced networks and telecommunications systems (ANTS). IEEE. https://doi.org/10.1109/ants47819.2019.9118073
19. Patra S, Middya AI, Roy S (2021) PotSpot: Participatory sensing based monitoring system for pothole detection using deep learning. Multimed Tools Appl 80(16):25171–25195. https://doi.org/10.1007/s11042-021-10874-4
20. Middya AI, Ray B, Roy S (2020) Auction based resource allocation mechanism in federated cloud environment: TARA. IEEE Trans Services Comput 1:1. https://doi.org/10.1109/tsc.2019.2952772
21. Bose B, Dutta J, Ghosh S, Pramanick P, Roy S (2018) D&RSense: detection of driving patterns and road anomalies. In: 3rd International conference on internet of things: smart innovation and usages (IoT-SIU). IEEE. https://doi.org/10.1109/iot-siu.2018.8519861
22. Rehena Z, Mukherjee R, Roy S, Mukherjee N (2014) Detection of node failure in wireless sensor networks. In: Applications and innovations in mobile computing (AIMoC). IEEE. https://doi.org/10.1109/aimoc.2014.6785531
23. Ghosh K, Roy S, Das PK (2009) An alternative approach to find the fermat point of a polygonal geographic region for energy efficient geocast routing protocols: global minima scheme. In: 1st International conference on networks & communications. IEEE https://doi.org/10.1109/netcom.2009.30
24. Middya AI, Roy S (2021) Geographically varying relationships of COVID-19 mortality with different factors in India. Sci Rep 11(1):1. https://doi.org/10.1038/s41598-021-86987-5
25. Dong R, Pedrycz W (2008) A granular time series approach to long-term forecasting and trend forecasting. Physica A Stat Mech Appl 387(13):3253–3270. https://doi.org/10.1016/j.physa.2008.01.095
26. Froelich W, Pedrycz W (2017) Fuzzy cognitive maps in the modeling of granular time series. Knowl-Based Syst 115:110–122. https://doi.org/10.1016/j.knosys.2016.10.017
27. Deng W, Wang G, Zhang X, Xu J, Li G (2016) A multi-granularity combined prediction model based on fuzzy trend forecasting and particle swarm techniques. Neurocomputing 173:1671–1682. https://doi.org/10.1016/j.neucom.2015.09.040
28. Jana RK, Ghosh I, Sanyal MK (2020) A granular deep learning approach for predicting energy consumption. Applied Soft Computing **89**, 106,091. https://doi.org/10.1016/j.asoc.2020.106091
29. Rule induction for forecasting method selection (2009) Meta-learning the characteristics of univariate time series. Neurocomputing 72(10):2581–2594. https://doi.org/10.1016/j.neucom.2008.10.017
30. Gordon J, Bronskill J, Bauer M, Nowozin S, Turner RE (2019) Meta-learning probabilistic inference for prediction. In: 7th International conference on learning representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019. OpenReview.net. https://openreview.net/forum?id=HkxStoC5F7
31. Yao H, Liu Y, Wei Y, Tang X, Li Z (2019) Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3308558.3313577
32. Zhou S, Lai KK, Yen J (2012) A dynamic meta-learning rate-based model for gold market forecasting. Expert Syst Appl 39(6):6168–6173. https://doi.org/10.1016/j.eswa.2011.11.115
33. Guo H, Pedrycz W, Liu X (2018) Hidden markov models based approaches to long-term prediction for granular time series. IEEE Trans Fuzzy Syst 26(5):2807–2817. https://doi.org/10.1109/TFUZZ.2018.2802924
34. Leite D, Gomide F, Ballini R, Costa P (2011) Fuzzy granular evolving modeling for time series prediction. In: IEEE international conference on fuzzy systems (FUZZ-IEEE 2011), pp 2794–2801 https://doi.org/10.1109/FUZZY.2011.6007452
35. Lemke C, Gabrys B (2010) Meta-learning for time series forecasting and forecast combination. Neurocomputing 73(10):2006–2016. https://doi.org/10.1016/j.neucom.2009.09.020
36. Ali AR, Gabrys B, Budka M (2018) Cross-domain meta-learning for time-series forecasting. Procedia Comput Sci 126:9–18. https://doi.org/10.1016/j.procs.2018.07.204
37. Abraham A (2004) Meta learning evolutionary artificial neural networks. Neurocomputing 56:1–38. https://doi.org/10.1016/S0925-2312(03)00369-2
38. Cecaj A, Lippi M, Mamei M, Zambonelli F (2020) Comparing deep learning and statistical methods in forecasting crowd distribution from aggregated mobile phone data. Appl Sci 10:1. https://doi.org/10.3390/app10186580
39. Yeh CM, Zhu Y, Ulanova L, Begum N, Ding Y, Dau HA, Silva DF, Mueen A, Keogh E (2016) Matrix profile i: All pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In: IEEE 16th International Conference on Data Mining (ICDM), pp 1317–1322. https://doi.org/10.1109/ICDM.2016.0179
40. Law SM (2019) Stumpy: a powerful and scalable python library for time series data mining. J Open Source Softw 4(39):1504. https://doi.org/10.21105/joss.01504
41. Zhu Y, Imamura M, Nikovski D, Keogh E (2017) Matrix profile vii: Time series chains: A new primitive for time series data mining (best student paper award). In: IEEE international conference on data mining (ICDM). https://doi.org/10.1109/ICDM.2017.79
42. Ministry of Environment, Forest and Climate Change, Govt. of India: Central Pollution Control Board. http://www.cpcb.nic.in/. Accessed: 31 March 2021
43. Sudalai Raj Kumar: Daily Temperature of Major Cities (2020). https://www.kaggle.com/sudalairajkumar/daily-temperature-of-major-cities
44. Zielak: Bitcoin Historical Data (2021). https://www.kaggle.com/mczielinski/bitcoin-historical-data
45. Bisong E (2019). Google Colaboratory. https://doi.org/10.1007/978-1-4842-4470-8_7
46. Google colaboratory. https://colab.research.google.com. Accessed 31 March 2021
47. Van Rossum G, Drake FL Jr (1995) Python tutorial. Centrum voor Wiskunde en Informatica Amsterdam
48. van der Walt S, Colbert SC, Varoquaux G (2011) The numpy array: a structure for efficient numerical computation. Comput Sci Eng 13(2):22–30
49. Martín Abadi et al. (2015) Tensorflow:large-scale machine learning on heterogeneous systems
50. Seasonal decomposition by moving averages. https://www.statsmodels.org/stable/_modules/statsmodels/tsa/seasonal.html#seasonal_decompose. Accessed 03 March 2021
51. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M,

Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
52. US Department of State: air now international US embassies and consulates. https://www.airnow.gov/international/us-embassies-and-consulates/. Accessed 31 March 2021
53. Kaggle. https://www.kaggle.com/. Accessed 31 March 2021
54. Kissock JK (2021) Unversity of Dayton average daily temperature archive. http://academic.udayton.edu/kissock/http/Weather/. Accessed 03 March 2021
55. Seabold S, Perktold J (2010) Statsmodels: Econometric and statistical modeling with python. In: Proceedings of the 9th python in science conference, p 1. https://conference.scipy.org/proceedings/scipy2010/pdfs/seabold.pdf