Physica A 655 (2024) 130176

Contents lists available at ScienceDirect

Physica A

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

Explainable Artificial Intelligence methods for financial time series

Paolo Giudici [a,*], Alessandro Piergallini [b], Maria Cristina Recchioni [b], Emanuela Raffinetti [a]

[a] Department of Economics and Management, University of Pavia, Pavia, Italy
[b] Department of Economics and Social Sciences, University Politecnica Marche, Ancona, Italy

ARTICLE INFO

Keywords:
Explainable Artificial Intelligence
LSTM and GRU
Shapley–Lorenz values
Financial time series
Bitcoin prices

ABSTRACT

We consider the problem of developing explainable Artificial Intelligence methods to interpret the results of Artificial Intelligence models for time series data, taking time dependency into account. To this end, we extend the Shapley–Lorenz method, normalised by construction, to Artificial Intelligence for time series, such as neural networks and recurrent neural networks. We illustrate the application of our proposal to a time series of Bitcoin prices, which acts as the response variable, along with time series of classical financial prices, which act as explanatory variables.

Three main findings emerge from the analysis. First, recurrent neural networks lead to a better performance, in terms of accuracy and robustness, with respect to classic neural networks. Second, the best performing models indicate that Bitcoin prices are affected mostly by their lagged values, and that their explainability, in terms of classical financial assets, is limited. Third, although limited, the contribution of classical assets to Bitcoin price prediction is well captured by recurrent neural networks.

1. Introduction

Finance, along with health care and robotics, is one of the areas in which Artificial Intelligence (AI) is having a leading role in generating predictions of future scenarios.

Time series data, common in financial models and typically analysed using econometric models, are now also being examined using generative AI models, based on recurrent neural networks. This enables the generation of high-frequency price forecasting by treating price time series as input (see, e.g. [1]). More traditional machine learning models, such as feed forward networks are also used, the functioning and application of which was analysed, for example, in the work by [2].

Neural network models allow for the creation of more accurate long-term forecasts, as they can be integrated with multiple steps ahead forecast models. Although classic recurrent neural networks cannot be used, due to the Gradient Descent problem, specific architectures can, such as the LSTM and GRU (see, e.g. [3] for more details). These networks have been used for both univariate and multivariate high-frequency time series data, obtaining excellent performances, as illustrated in the analysis carried out by [4].

The application of Artificial Intelligence to finance has led to the development of financial technologies (*FinTech*), and has generated new econometric models for the analysis of time dependent time series, more complex than the classical ones (*ARIMA* and *GARCH* models), such as artificial neural networks and recurrent neural networks. While typically more accurate than classical models, machine learning models for time series are often considered black-boxes: their results are not explainable in terms of their drivers. In classical models, the sign and magnitude of each autoregressive parameter explain the direction and strength of

* Corresponding author.
E-mail address: paolo.giudici@unipv.it (P. Giudici).

https://doi.org/10.1016/j.physa.2024.130176
Received 23 June 2024; Received in revised form 16 September 2024
Available online 21 October 2024

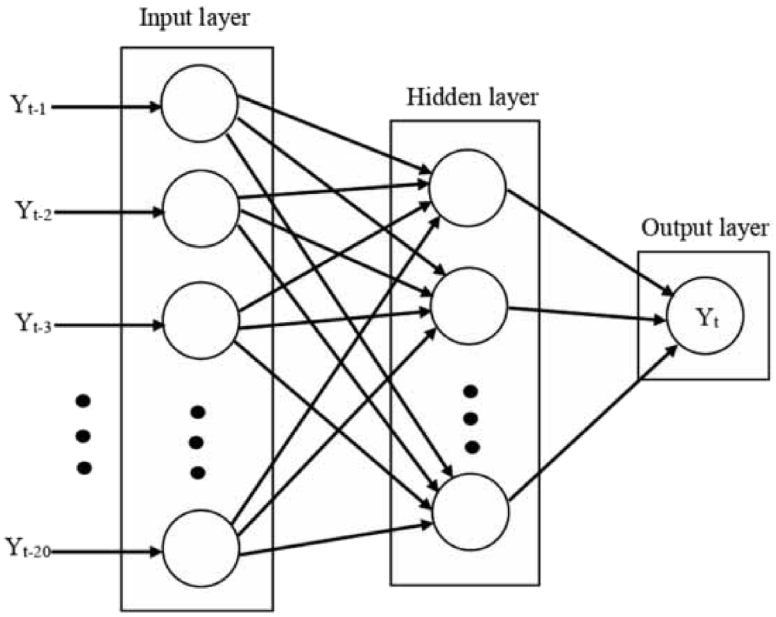


Fig. 1. Architecture of a NNAR network.

the relationship between a lagged predictor and the response variable. Whereas in machine learning models it is not possible to re-conduct the many fitted parameters to any relationship. However, further models can be run on the model predictions obtained with machine learning, such as Shapley values (see [5]) or their normalised version, the Shapley–Lorenz values (see [6]). The contribution of this paper is to show how to obtain Shapley–Lorenz values for the more complex models used in generative AI models for time series: recurrent neural networks.

In the next paragraph, we will introduce and describe how different neural network architectures can be used in generative AI for time series modelling. Additionally, we will explain how Shapley values, along with their normalised version, Shapley–Lorenz values, can be exploited to explain the output (generations, predictions, recommendations) obtained with such models.

2. Artificial Intelligence models for time series

In this section we consider two types of neural networks that can take into account the time dependency embedded the time series data to generate forecasts and scenarios: the classic forward networks, introduced with the advent of machine learning models, and the recurrent networks, which are also the basis of the modern generative AI models.

Specifically, we will focus on an autoregressive neural network model (NNAR), whereas as recurrent network models we will consider a Long-Short-Term Memory model (LSTM) and a Gated Recurrent Unit network model (GRU). For robustness, we will also consider two "classic" feed forward neural networks: a multilayer perceptron network (MLP) and a radial basis function network (RBF).

We will compare the different neural network models not only in terms of predictive accuracy, as it is typically done in the literature, but also in terms of explainability. To this aim, given the "black-box" nature of neural networks, in the next section we will also propose to evaluate time series Artificial Intelligence models by means of Shapley–Lorenz values, extending what done by [6] for classic machine learning models.

2.1. Autoregressive neural networks

Autoregressive neural networks (NNAR) are described by the architecture in Fig. 1. For more details see, e.g. [2].

From Fig. 1 note that the neural network has, in the input layer, p lagged values of the response variable, similarly as in the classic AR(p) model. Differently from what occurs in the AR(p) model, however, the input is not connected directly with the response, but it is processed by an intermediate hidden layer, a computational unit which transforms the signals from the input before employing them to determine the output values. It follows that a NNAR(p, s) model is a network characterised by p input nodes, the lags of the response variable $(y_{t-1}, y_{t-2}, \dots, y_{t-p})$, and the s neurons in the hidden layer.

In the model specification phase, various architectures of NNAR can be compared. These architectures may differ in terms of the number of lagged variables, nodes in the hidden layer, or additional input nodes, such as seasonal components or exogenous covariates. The optimal structure can be selected based on appropriate criteria, such as Mean Square Error (MSE) or the Akaike Information Criterion (AIC).

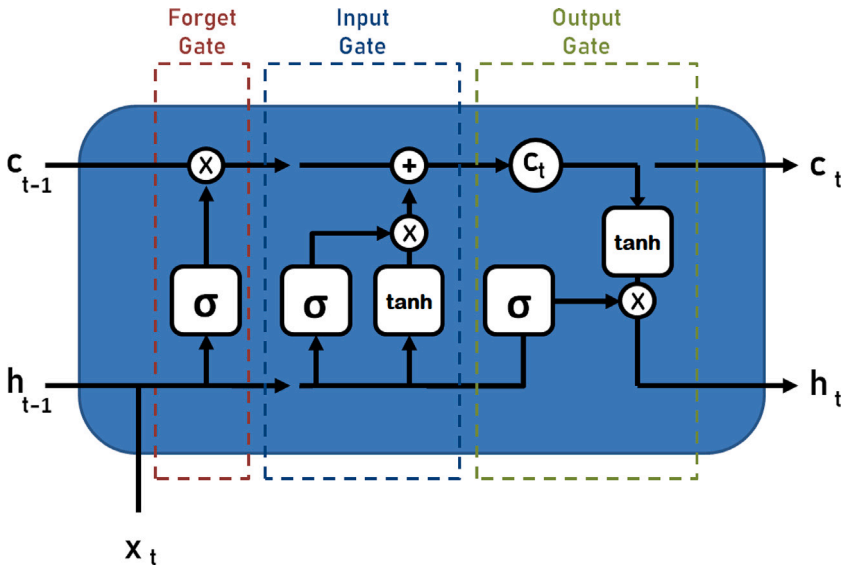


Fig. 2. Processing unit of a LSTM network.

In the prediction phase, the network needs to be applied iteratively. For instance, if data is collected daily and the aim is to predict the value for the following day, the available historical inputs up to the current date will be used for analysis. If the forecasting time horizon extends further, such as predicting the value for the second day, the forecast for the next day, along with the historical data, will be utilised as input, and so on.

2.2. Long Short Term Memory networks

Long Short-Term Memory (LSTM) networks can be described by a sequence of recursive steps, in which each input x_t is processed as in Fig. 2. For more details see, e.g. [3].

From Fig. 2 note that, during the processing of each unit x_t, there are two flows of information, represented by the two horizontal lines that enter and exit the LSTM: the cell state (c_t), known as Long Term Memory (LTM), and the hidden state (h_t), known as Short Term Memory (STM).

Each processing unit in the LSTM is characterised by three processing phases, as follows:

1. Forget Gate: this phase determines the percentage of information to forget. Looking at Fig. 2, the information received from the previous hidden state (h_{t-1}) and from the current input (x_t) are jointly processed by the sigmoid function $\sigma(z) = 1/(1+e^{-z})$, with $z \in (-\infty, +\infty)$, determining the update of the forget gate, $f_t \in [0, 1]$: $f_t = \sigma(h_{t-1}w_1 + x_t w_2 + b_1)$, which will then multiply the previous cell state (c_{t-1});
2. Input Gate: in this phase the long term memory LTM is updated, by means of two activation functions. A hyperbolic tangent function $\widetilde{c}_t$, with values in $[-1, 1]$, uses the previous hidden state and the current input, but with different weights and biases: $\widetilde{c}_t = tanh(h_{t-1}w_3 + x_t w_4 + b_2)$.
   A sigmoid activation function is instead employed to determine the memory potential, i_t: $i_t = \sigma(h_{t-1}w_5 + x_t w_6 + b_3)$.
   The LTM (cell) state is then updated summing the result of the forget gate with the product between $\widetilde{c}_t$ and i_t: $c_t = f_t \times c_{t-1} + \widetilde{c}_t \times i_t$;
3. Output Gate: in the third phase, the short-term memory (STM) is updated, using a tangent hyperbolic activation function: $v_t = tanh(c_t)$.
   Additionally, a sigmoid function (o_t) is calculated, taking as arguments the previous hidden state and the current input: $o_t = \sigma(h_{t-1}w_7 + x_t w_8 + b_4)$.
   The new STM (hidden state) is thus calculated by multiplying the two results: $h_t = v_t \times o_t$.

2.3. Gated Recurrent Unit networks

Gated Recurrent Unit (GRU) neural networks are characterised by processing units with the structure displayed in Fig. 3. For more details, see e.g. [3].

From Fig. 3, note that each processing unit in a GRU is characterised by two phases, as follows:

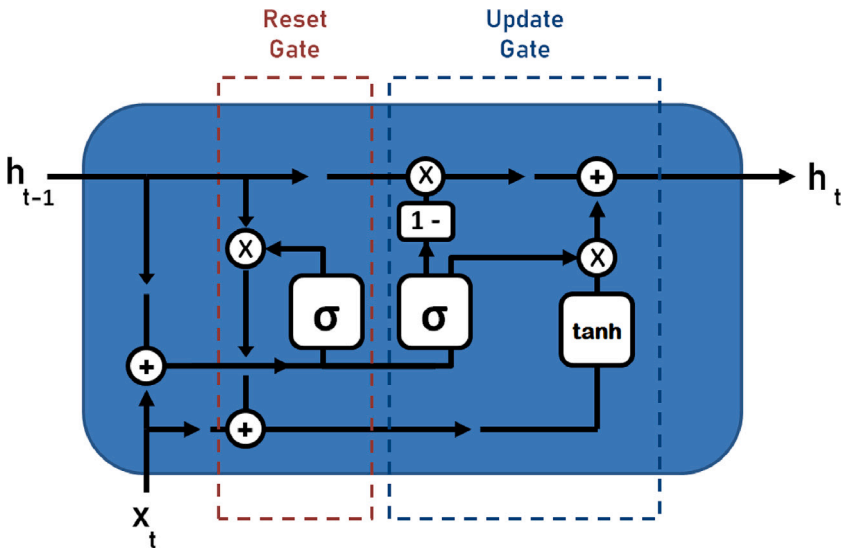


Fig. 3. Processing unit of a GRU network.

1. Reset Gate: it determines how much past information should be remembered. The reset value for a specific time step (r_t) is calculated through the sigmoid function:
   $r_t = \sigma(h_{t-1}w_1 + x_t w_2 + b_1)$.
   Next, r_t is multiplied with the previous hidden state (h_{t-1}) to determine how much past information should be retained for that particular time step t;
2. Update Gate: it determines which new information should be added. First, a candidate hidden state ($\widetilde{h}_t$) is determined through the *tanh* function, which will have as arguments the current input and the product between the reset and the previous hidden state: $\widetilde{h}_t = tanh((h_{t-1} \times r_t)w_3 + x_t w_4 + b_2)$.
   Thus, to determine the candidate, not all the information from the previous time instant $(t-1)$ is considered, but only what must be remembered, expressed by $h_{t-1} \times r_t$. Second, the information to be transferred from the previous hidden state is calculated as: $z_t = \sigma(h_{t-1}w_5 + x_t w_6 + b_3)$.
   It follows that from the current hidden state candidate we keep a percentage equal to z_t, and for the remaining $1 - z_t$ we keep the information of the previous hidden state. It follows that the update will be determined as: $h_t = (1 - z_t)h_{t-1} + z_t\widetilde{h}_t$.

In summary, a GRU network employs fewer calculations than the LSTM and is, in fact, faster in the forecasting phase.

2.4. Multilayer perceptron networks

A multilayer perceptron (MLP) is a feed forward neural network that consists of multiple layers of interconnected nodes: an input layer, one or more hidden layers, and an output layer. Each neuron in the network is connected to every neuron in the adjacent layers.

The input layer receives the initial data, which is then passed through the hidden layers, where nonlinear transformations occur, and finally, the output layer produces the network's predictions or classifications. or more details, see e.g. [7].

2.5. Radial basis function networks

A Radial Basis Function Network (RBFN) is a feedforward neural network that uses radial basis functions as activation functions. It typically consists of three layers: an input layer, a hidden layer with radial basis function neurons, and an output layer.

The input layer consists of neurons that receive the input data. Each neuron represents a feature or input variable. The hidden layer contains radial basis function neurons. These neurons compute their output based on the distance between their input and a center point, typically called a prototype or centroid. The output of each neuron in this layer is a radial basis function, such as the Gaussian function or the multi quadratic function. The output layer usually consists of a single neuron or multiple neurons depending on the task (e.g., regression or classification). It combines the outputs of the hidden layer neurons to produce the final output of the network. For more details, see e.g. [7].

2.6. Shapley values and Shapley–Lorenz values

Shapley values (SV) were introduced in game theory (see, e.g. [5]) with the purpose of dividing the value of a game between the various participants, depending on their contribution. In the recent years, they have been applied in the Artificial Intelligence context, to mitigate the lack of interpretability of complex non-linear machine learning models which, while improving the accuracy of the predictions, lose their explainability (see, for example, [8,9]).

In the machine learning context, Shapley values are exploited in a model post processing step, and aim to divide the model's predictions between the different feature variables, depending on their importance. In other words, the Shapley values approach is not a model selection technique, but it is technique that is applied to the output of a model to determine which variables features have mostly contributed to it. In this way, Shapley values can help the interpretation of a machine learning model, even when it is a black-box model.

From a more formal viewpoint, the Shapley value for the ith prediction ($i = 1, \dots, n$) of a feature variable k ($k = 1, \dots, K$) is the following:

$$\phi_i^k = \sum_{X' \subseteq C(X) \setminus X_k} \frac{|X'|!(K - |X'| - 1)!}{K!} \cdot [\hat{Y}_{(X'_{\cup} X_k)_i} - \hat{Y}_{(X')_i}], \tag{1}$$

where $\hat{Y}_{(X'_{\cup} X_k)_i}$ and $\hat{Y}_{(X')_i}$ denote the ith predicted value provided, respectively, by a model with and without the kth predictor; $C(X) \setminus X_k$ is the set of all the possible model configurations which can be obtained excluding variable X_k; $|X'|$ is the number of variables included in a given model configuration.

Thus, for each prediction, the Shapley values of each variable k, expressing its importance, is obtained as the average, over all possible models, of the difference between two predictions: that of a model in which the kth variable is used, and that in which it is not used.

What described so far are the Shapley values for any individual predictions, also known as Local Shapley values. The Global Shapley values of a variable k can be obtained taking the mean, or the sum, of the Shapley values of k over all predictions $i = 1, \dots, n$.

Shapley values have gained much importance in the Artificial Intelligence community, thanks to their capability of making machine learning model explainable by opening their black box. Their diffusion has contributed to the growth of explainable Artificial Intelligence, as described, for example, in [10–12].

While successful, Shapley values suffer, however, from a main disadvantage. They are not normalised: their value depends on the unit of measurement of the response variable. Differently from measures such as the Area Under the Curve AUC (see, e.g. [13]), or the R^2, Shapley values cannot be interpreted as a percentage value, limiting their accountability. Thus, they cannot be used to compare the explainability of different response variables.

To solve this issue, [6] introduced Shapley–Lorenz values (SLV), which represent a global measure of explainability, normalised by definition. Essentially, Shapley–Lorenz values replace the difference between the two predictions in Eq. (1) with the difference between their predictive performance, as expressed by the Lorenz Zonoid, a function of the well known Gini coefficient, which takes values in the [0,1] range (see, e.g. [14]).

More formally, the Shapley–Lorenz value of a feature variable k (with $k = 1, \dots, K$) is the following:

$$LZ^k(\hat{Y}) = \sum_{X' \subseteq C(X) \setminus X_k} \frac{|X'|!(K - |X'| - 1)!}{K!} \cdot [LZ(\hat{Y}_{X' \cup X_k}) - LZ(\hat{Y}_{X'})], \tag{2}$$

where $C(X) \setminus X_k$ and $|X'|$ were already defined in Eq. (1), whereas $LZ(\hat{Y}_{X' \cup X_k})$ and $LZ(\hat{Y}_{X'})$ are the Lorenz Zonoids of the response variable Y explained by the models which, respectively, include $X' \cup X_k$ predictors or only X' predictors.

We remark that, similarly to Shapley values, Shapley–Lorenz values are model-agnostic: they can be applied to any machine learning model, differently from model specific explanation such as, for example, feature importance plots (see, e.g. [15]). However, differently from Shapley values, Shapley–Lorenz values are normalised, taking values in the close interval $[0, 1]$. This feature plays a key role on the interpretation side: the Shapley–Lorenz value of a variable provides the percentage of variability of the response variable that it explains, i.e. its added value.

Shapley–Lorenz values are generally non-negative, according to the inclusion principle, which holds exactly for nested linear models, as demonstrated in [14]. When the underlying machine learning model is non-linear, the inclusion principle is only approximate and, consequently, Shapley–Lorenz values may become negative.

2.7. S.A.F.E. Artificial Intelligence

Machine Learning (ML) methods are boosting the applications of Artificial Intelligence. However, differently from ordinary computer software and applications, AI not only converts inputs into outputs, but can also change the surrounding environment, with the risk of creating harms for individuals and organisations.

This is the reason why policy makers, regulators and standard bodies around the world are issuing regulations and recommendations that AI developers, deployers and users should follow to manage the risks arising from the adoption of AI methods.

AI risk management requires to develop a consistent set of AI risk metrics that can be employed to monitor the compliance of AI applications. Such a set of metrics is not common practice, yet.

Recently, [16] have summarised the requirements in the existing regulations and recommendations into four main measurable "S.A.F.E." key principles: "S" for sustainability; "A" for accuracy; "F" for fairness; "E" for explainability.

Sustainability refers to the robustness of an AI system to extreme events, such as cyber attacks or environmental issues. The measurement of robustness is well known in the statistics and machine learning literature and it is usually conducted in terms of the difference between the accuracy of two different predictions, obtained respectively under normal and perturbed input data. Accuracy can be measured by the Area Under the ROC Curve (AUC) or by the Root Mean Square Error (RMSE) (see, e.g. [17]).

The measurement of Accuracy is well known in statistics and machine learning: the RMSE is routinely employed for a continuous response; the AUC is typically employed for a binary response (see, e.g. [13,18]). More recently, [19] have introduced a new accuracy measure that can be applied to both types of responses, generalising the AUC to the continuous case.

Fairness is one of the most important requirements for AI applications. Unfairness indicates that the output leads to an unequal treatment of different population groups, by gender, age, and nationality, for example. Several recent research papers deal with the measurement of fairness. Most of them are based on parity measures, which calculate the difference in accuracy of the AI output obtained separately on different population groups. Accuracy can be measured employing AUC or RMSE (see, e.g. [20]).

Explainability is a requirement that has emerged with the development of highly accurate machine learning models which have, however, so many parameters that it is very difficult to reconduct the output to the inputs which determine it. The measurement of explainability requires attaching to each input variable an importance weight that expresses its influence on the final output such as expressed, for instance, by Shapley values or by Shapley Lorenz values. A model is explainable if there exists at least one input variable which significantly impacts the output.

A recent paper [21] suggested to measure the S.A.F.E. principles using state of the art metrics, such as AUC, RMSE and Shapley values which are not, however, consistent with each other. A more consistent measurement model was put forward by [16], who replaced traditional metrics with new ones based on Lorenz Zonoids, the multivariate extension of the Gini coefficient.

In this paper, we will assess whether machine learning models for financial time series are S.A.F.E.: Sustainable, Accurate, Fair and Explainable, employing the score metrics introduced in [16], which will now be recalled.

Explainability score. For a machine learning model with K predictors, the score for explainability can be calculated on the whole sample as:

$$Ex - Score = \frac{\sum_{k=1}^{K} SL_k}{LZ(Y)}, \tag{3}$$

where $LZ(Y)$ corresponds to the Lorenz Zonoid-value of the response variable Y, and SL_k denotes the Shapley–Lorenz values associated with the kth predictor.

Accuracy score. Consider a machine learning model with a set of $k \leq K$ predictors, properly chosen by means of a feature selection procedure, and a test sample from the dataset. The model's accuracy score can be measured as follows:

$$Ac - Score = \frac{LZ(\hat{Y}_{X_1,\ldots,X_k})}{LZ(Y_{test})}, \tag{4}$$

where $LZ(\hat{Y}_{X_1,\ldots,X_k})$ is the Lorenz Zonoid of the predicted response variable, obtained using k predictors on the test set, and $LZ(Y_{test})$ is the Y response variable Lorenz Zonoid value computed on the same test set.

Fairness score. Consider a machine learning model with k predictors, resulting from the feature selection procedure and, in addition, a protected variable which is categorised in M population groups. We can then measure the fairness score of the model as:

$$Fair - Score = 1 - LZ(V_M^{SL}), \tag{5}$$

where $LZ(V_M^{SL})$ denotes the Lorenz Zonoid computed on the vector V_M^{SL} whose elements correspond to the sum of the selected predictors' Shapley–Lorenz values in each population.

Sustainability score. Consider a machine learning model with k predictors, resulting from the feature selection procedure, and, in addition, divide the sample in G distinct groups, corresponding to increasing values of the predicted response. We can then measure the sustainability score of a machine learning model as:

$$Sust - Score = 1 - LZ(V_G^{SL}), \tag{6}$$

where $LZ(V_G^{SL})$ indicates the Lorenz Zonoid calculated on the vector V_G^{SL}, whose elements correspond to the sum of the selected predictors' Shapley–Lorenz values in each group.

In the next Section we will apply the previous scores to the proposed machine learning time series models.

3. Application

We have applied the above mentioned models to the data described in [16], who employed Shapley–Lorenz values to explain the results of a neural network model applied to time series of daily Bitcoin prices. Their model is a simple feed forward network that does not take time dependency into account, assuming that the daily time series observations are independent in time.

Below are the descriptive statistics for the daily time series under consideration, covering the period from May 18, 2016 to April 30, 2018 (see Table 1).

Table 1
Descriptive statistics: Minimum (Min), Mean, Median (Med), Maximum (Max) and Standard Deviation (SD) of all available daily prices.

Variable	Min	Mean	Med	Max	SD
$BITCOIN$	438.38	3919.05	1713.00	19 650.01	4318.98
USD/EUR	0.80	0.88	0.89	0.96	0.04
$GOLD$	1128.42	1275.57	1276.83	1366.38	52.34
$SP500$	2000.54	2399.17	2390.90	2872.87	212.31
$USD/YUAN$	6.27	6.68	6.67	6.96	0.19
OIL	39.51	49.36	49.30	57.20	3.37

Table 2
Shapley Lorenz values for the NNAR, LSTM and GRU neural network models.

NNAR model	
Variable	Shapley–Lorenz value (SLV)
USD/EUR	0.1212
$GOLD$	0.0238
$SP500$	0.0163
OIL	0.0086
$USD/YUAN$	0.0706
LSTM model	
Variable	Shapley–Lorenz value (SLV)
USD/EUR	0.0574
$GOLD$	0.0305
$USD/YUAN$	0.0229
$SP500$	0.0156
OIL	0.0143
GRU model	
Variable	Shapley–Lorenz value (SLV)
$USD/YUAN$	0.0463
OIL	0.0396
$SP500$	0.0322
$GOLD$	0.0230
USD/EUR	0.0298

Before describing our empirical findings, we would like to emphasize that all models have been implemented in Python and are available upon request, ensuring full reproducibility of our work. Additionally, to facilitate comparability, we have used the same training and validation data for all models.

Finally, we remark that, for each model, all predicted prices have been normalised in the interval $[0, 1]$, calculating $\frac{p-min}{max--min}$, with p the predicted price; min and max, respectively, its minimum and maximum over the validation set.

3.1. Autoregressive neural networks

Several alternative structures of autoregressive neural networks have been compared, in terms of the Root Mean Square Error (RMSE) of the corresponding predictions.

The comparison has been made employing the first year of data, subdivided into a training set with all data before September 30, 2017, and a validation set with all data from October 1, 2017 to December 31, 2017. This corresponds to a proportion of about 70% training data and 30% validation data.

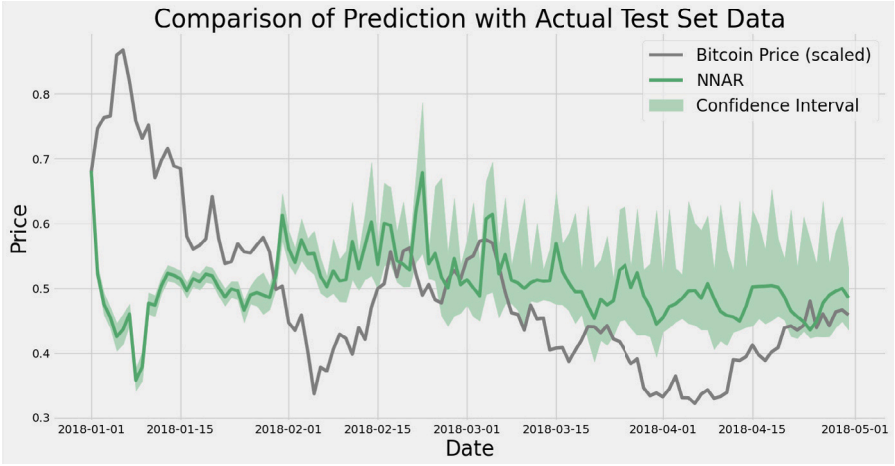
For the given train/validation data split, the best neural network configuration is found for a network that corresponds to the presence of two lagged variables and four neurons in the hidden layer, with a RMSE equal to 0.2801. If we increase the number of hidden layer neurons, the RMSE in the training set decreases, but the RMSE in the validation set increases, consistently with the well known over fitting behaviour of complex neural networks.

We have applied the chosen model to obtain predictions from January 1, 2018, onwards. They are represented in Fig. 4(a) for the NNAR model.

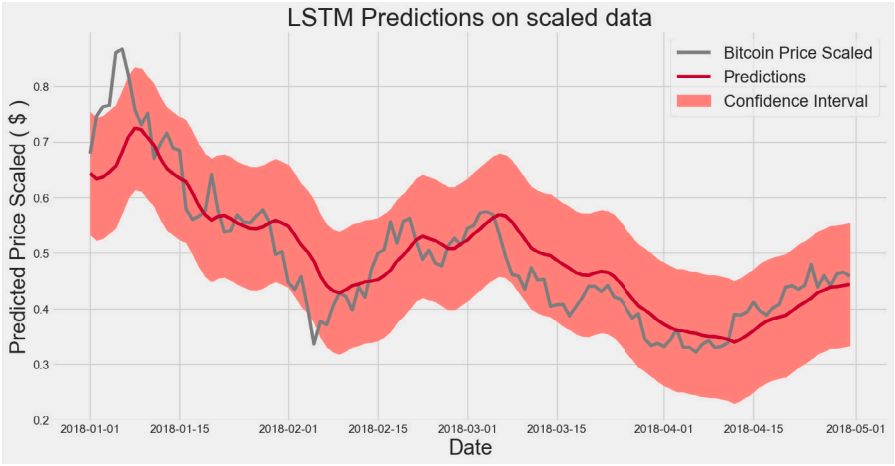
Overall, if from Fig. 4(a) we calculate the square root of the euclidean distance between the (scaled) Bitcoin prices to be predicted and the corresponding predictions, we obtain a RMSE equal to 0.13.

This result should be compared to the RMSE of the predictions obtained with the time independent model of [16], which is equal to 0.14 and, therefore, the NNAR model slightly improves predictive accuracy.

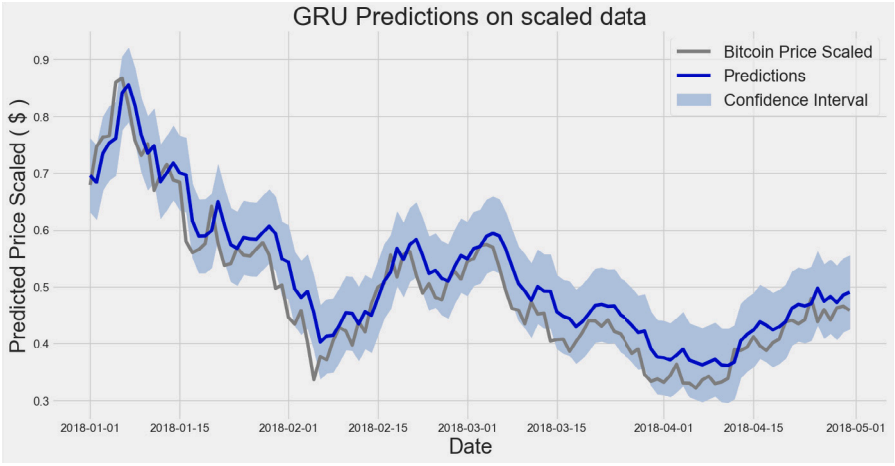
We now move to explainability, to assess which explanatory variables mostly drive the predictions derived from the neural network models. In the top part of Table 2 we report the Shapley–Lorenz values of the predictions in Fig. 4(a), in absolute values.



(a) NNAR model predictions and confidence intervals



(b) LSTM model predictions and confidence intervals



(c) GRU model predictions and confidence intervals

Fig. 4. Model predictions and confidence intervals for the NNAR, LSTM and GRU neural network models. All predictions are calculated in a rolling window mode: all data up to day t is included to obtain the prediction of the Bitcoin price at time $t + 1$.

From the top part of Table 2 note that, for the NNAR model, only the USD/EUR exchange rate explains more than the 10% of the variability of the response (Bitcoin prices). These findings differ from those illustrated in [16], which reported a Shapley–Lorenz value of about 35% for Gold, followed by SP500 at about 10%, and the other three variables below 10%.

The difference in Shapley–Lorenz values is due to the different employed models. In [16], the response daily prices were assumed to be independent of each other; whereas in the NNAR model considered here, the same prices are dependent on each other, according to an autoregressive structure. This implies that the variability of the Bitcoin prices is explained not only by the prices of the classical assets, as in [16], but also by the lagged values of the response itself, likely to matter, being the Bitcoin a rather speculative asset. The insertion of the lagged variables may improve the model's predictive performance, but it can also reduce the explanation due to the classical assets, correlated with them. This motivates the difference between the results displayed in Table 2 and the findings showed in [16].

In other words, when correctly taking into account the endogenous autoregressive dependence between Bitcoin prices, as in this paper, the exogenous influence of classical assets decreases.

3.2. Long Short Term Memory networks

We have compared several alternative structures of LSTM networks, in terms of the RMSE of the predictions made for the last quarter of 2017, on the basis of the first three quarters of the same year.

The best LSTM network corresponds to a learning rate of 0.01, 10 epochs and 100 units in the short memory.

The predictions obtained using the best LSTM model, for the validation set, are represented in Fig. 4(b).

From Fig. 4(b), note that the predictions appear to be more accurate and precise than those provided by the NNAR model. Overall, the RMSE of the predictions from the LSTM model in Fig. 4(b) is equal to 0.056, more than two times lower than the RMSE obtained with the NNAR model, for the same data and train/validation partition. The result indicates that the predictive accuracy of the LSTM is much higher than that of the NNAR model.

Moving to explainability, the middle part of Table 2 reports the obtained Shapley–Lorenz values for the LSTM model.

Comparing the results given by the NNAR model with those of the LSTM model, it seems that the LSTM model is less explainable than the NNAR: summing the five Shapley–Lorenz values we obtain a total of about 0.24 for the NNAR and of about 0.11 for the LSTM. The worse performance of the LSTM model can be due to the fact that the LSTM model gives more weight to the endogenous autoregressive component, and less to the exogenous classical financial prices. In summary, when we move from the NNAR to the LSTM, a decrease in explainability of about 45% arises. However, it still appears as an acceptable result if considered in combination with an increase in accuracy of about 232%.

3.3. Gated Recurrent Unit networks

We have then applied alternative Gate Recurrent Unit networks, with the same data and train/validation splits as for the other models. The best model is obtained with a Learning Rate of 0.001, 10 epochs and 100 units.

The chosen model leads to the predictions depicted in Fig. 4(c), with the corresponding confidence intervals.

The overall RMSE of the predictions in Fig. 4(c) is equal to 0.04: lower than the RMSE derived with the LSTM model.

Moving to the assessment of explainability, we report in the bottom part of Table 2 the Shapley–Lorenz values corresponding to the predictions of the GRU model.

Comparing the Shapley–Lorenz values from the GRU model with those from the LSTM model, note that the two explanations are similar. The sum of the five Shapley–Lorenz values for the GRU model is equal to about 18%, lower than that of the NNAR model (but slightly higher than that of the LSTM model). As discussed for the LSTM model, this result is likely to be due to the prevalence of the endogenous component, represented by the lagged response variables. This is also consistent with the majority of existing literature, which indicates that Bitcoin is a speculative asset. In summary, when we move from the NNAR to the GRU, we get a decrease in explainability of about 25% which is an acceptable result if considered in combination with an increase in accuracy of about 325%.

3.4. S.A.F.E. AI comparison

We now extend model comparison to consider also the Sustainability and the Accuracy of the different models, in line with the S.A.F.E. model proposed in [16]. To this aim, Table 3 presents a comparison of the three models in terms of accuracy, explainability and robustness. We also report, for the sake of comparison, the RMSE of the different models.

To improve the robustness of our results, we include in the comparison two further neural network models that are time independent, as the model in [16]: a multilayer perceptron network (MLP) and a Radial Basis Function network (RBF).

Table 3 shows that the GRU and the LSTM model have an accuracy, as measured by the Ac-Score, much higher than that of the NNAR model, and of the two time independent models, the MLP and the RBF. The GRU is more accurate than the LSTM. The same Table shows that, in terms of explainability, the MLP, the RBF and the NNAR models are more explainable than the LSTM and the GRU models, however at the expense of a much lower predictive performance, in line with what already discussed. The GRU is slightly more explainable than the LSTM. These findings are in line with what already mentioned when comparing the RMSE and the Shapley–Lorenz values of the models. In addition, Table 3 shows that, in terms of sustainability, the Sust-Score of the LSTM and

Table 3
Comparison of the MLP, RBF, NNAR, LSTM and GRU models with in terms of the S.A.F.E. metrics and of the RMSE.

Model	Sust-Score	Ac-Score	Ex-score	RMSE
MLP	0.9661	0.4518	0.5114	0.1046
RBF	0.9538	0.4519	0.5443	0.0982
NNAR	0.7157	0.3718	0.2405	0.1358
LSTM	0.9607	0.8186	0.1122	0.0561
GRU	0.9244	0.8865	0.1778	0.0439

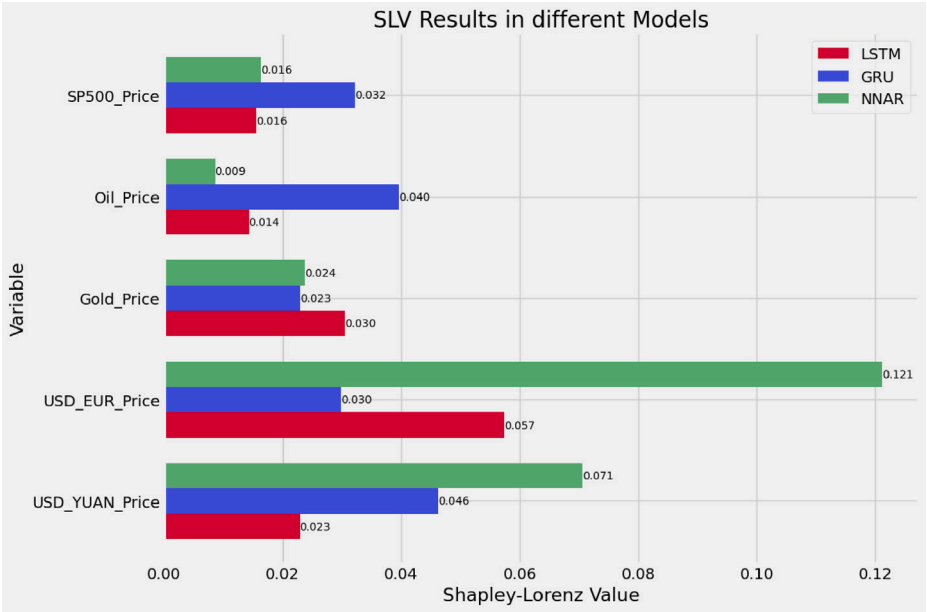


Fig. 5. Shapley–Lorenz values in different models.

of the GRU are much higher than that of the NNAR, indicating a higher robustness, intended as a stronger resilience to extreme data. The two time independent models, the MLP and the RBF, also achieve high robustness, likely due to their lower complexity.

We can now draw a final ranking of the five considered models. In terms of accuracy, the GRU, followed by the LSTM, are the best models; in terms of robustness, all models, different from NNAR, perform quite well; in terms of explainability, the simpler MLP and RBF networks are the best models. We can thus conclude that model choice depends on what is deemed more relevant: if it is predictive accuracy, the GRU, followed by the LSTM, is the best choice; if it is explainability, it is better to employ a simpler MLP or RBF; if it is robustness, all models part from the NNAR are good.

3.5. Financial interpretation

We now consider the implications of the models in terms of the considered financial application. Fig. 5 displays the output reported in Table 2 in a graphical format easier to visualise.

Fig. 5 clearly shows the higher importance of the explanations, for the NNAR model, with respect to the recurrent models.

To better understand the explanations, it is worth examining, for each explanatory variable, the relationship between its Shapley values and the original values.

A high correlation (in absolute value) would indicate that most of the variability of a variable is relevant for the prediction of the response; whereas a low correlation would indicate that the variable's variation is not related to that of the response.

Table 4 contains the obtained correlations, for all considered time dependent networks: NNAR, LSTM and GRU, as well as for the two time independent networks: MLP and RBF.

Table 4 indicates that the NNAR network does not present strong correlations, whereas the recurrent networks (LSTM and GRU) lead to a strong correlation for the exchange rates USD/YUAN, USD/EUR and for the SP500 index, similarly as the two independent networks (MLP and RBF).

These findings support once more the superiority of the recurrent networks, such as LSTM and GRU. They have a better predictive accuracy. In addition, even though their Shapley–Lorenz values for the explanatory variables are low, their variability is strongly correlated with that of the exogenous variables, and particularly for SP500, USD/YUAN and USD/EUR exchanges rates, similarly to what occurs for the least accurate and simpler time independent models.

Table 4
Correlations between Shapley values and explanatory variables.

Model	USD/YUAN	USD/EUR	GOLD	OIL	SP500
MLP	0.4400	−0.5700	−0.2800	−0.2900	0.9800
RBF	−0.8800	0.6800	−0.3600	−0.5000	0.6900
NNAR	−0.2600	0.0012	0.3300	−0.2000	−0.2400
LSTM	0.8600	0.4700	−0.3500	−0.4300	0.6800
GRU	0.8900	0.5400	−0.2700	−0.2300	0.6600

Table 5
Comparison of the variable importance rankings obtained from the NNAR, LSTM and GRU models, using both Shapley and Shapley–Lorenz values.

Shapley values		
NNAR	LSTM	GRU
USD/EUR	USD/EUR	$USD/YUAN$
$GOLD$	$USD/YUAN$	OIL
$SP500$	$SP500$	$SP500$
$USD/YUAN$	$GOLD$	USD/EUR
OIL	OIL	$GOLD$
Shapley–Lorenz values		
NNAR	LSTM	GRU
USD/EUR	USD_EUR	$USD/YUAN$
$GOLD$	$GOLD$	OIL
$SP500$	USD_YUAN	$SP500$
OIL	$SP500$	USD/EUR
$USD/YUAN$	OIL	$GOLD$

This indicates that recurrent network models better "transform" the variability of the inputs into predicted values of the response.

We conclude our analysis examining the robustness of our results in terms of the adopted explainability metrics. Specifically, we compare the variable rankings obtained using Shapley–Lorenz values against those obtained using the standard Shapley values.

Table 5 summarises all the rankings obtained with the three proposed neural network models, first using Shapley values and then Shapley–Lorenz values.

From Table 5, it can be observed that the NNAR and GRU models do not exhibit significant differences in the rankings obtained using classical Shapley values, compared to those obtained using Shapley–Lorenz values. Only LSTM presents a slight difference, as the price of Gold (GOLD) appears to assume a different level of importance. Overall, we can state that the rankings from Shapley–Lorenz values are highly consistent with those obtained from the standard Shapley values. The former are, however, preferable, as normalised and, therefore, easier to interpret.

3.6. Interaction between lagged variables

The above discussion clearly indicates that complex recurrent network models, such as LSTM and GRU, may considerably improve predictive accuracy, but at the expense of a lower explainability, particularly in terms of the exogenous variables. As the explainability of the predictors is the results of the interaction of many lagged effects, it becomes crucial to explain the results of these models in terms of the interactions among the lagged values of the variables, by means of Shapley–Lorenz values. In this section we show how to carry out this task.

As mentioned in the previous section, in a recurrent network model, such as LSTM or GRU, the lags describe how the past values of the response and of the predictors impact the forecasts. The ability of the model to remember and integrate information from previous time lags relies to its internal mechanisms, such as cell states and gating units. These mechanisms enable LSTMs and GRUs to selectively retain or forget information from various lags, which is crucial for modelling temporal sequences with long-term dependencies.

From an explainability standpoint, deriving an analytical formula that expresses the Shapley–Lorenz values in terms of the lagged variables is a complex task. However, the relationship can be estimated empirically. To this end, we applied an LSTM model for various values of the maximum number of lags, specifically for lag = $\{1, 5, 10, 15, 30, 60\}$, always using the same train/test partition. From the output of each model, we calculated the Root Mean Square Error (RMSE) on the test set.

The top panel of Fig. 6 displays the Shapley–Lorenz values for the five exogenous predictors as a function of the lags of the LSTM neural network, while the bottom panel illustrates the RMSE as a function of the same lags.

Fig. 6 shows that Lag 1 variables provide the best RMSE, and that, in this case, the largest Shapley–Lorenz value corresponds to the SP500, followed by the USD/EUR and USD/YUAN exchange rates, with Gold and Oil price at the end of the list.

When examining lag = 5 and lag = 60, which show comparable RMSE, the Shapley–Lorenz values for the USD/EUR and USD/YUAN exchange rates are similar. In contrast, the Shapley–Lorenz values for Gold and Oil prices increase when moving from lag = 5 to lag = 60, reducing the influence of the SP500 index. These findings may result from the fact that commodity prices

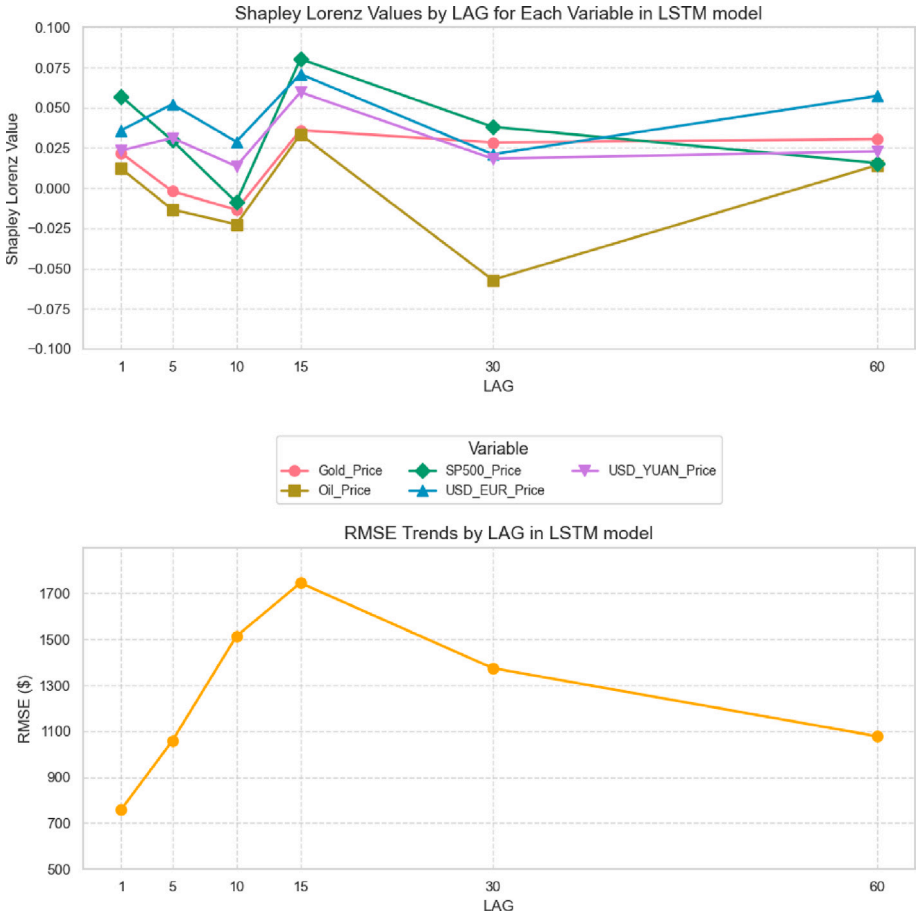


Fig. 6. Lag explainability of the LSTM neural network. Top panel: Shapley–Lorenz values for the five predictors as a function of the lags. Bottom panel: Root Mean Square Error (RMSE) as a function of the lags.

exhibit longer auto-correlations compared to the SP500 index, possibly due to market frictions associated with them. Thus, larger lags help capture the effects of these factors more effectively, consistent with the findings of [22].

For comparison, the top panel of Fig. 7 displays the Shapley–Lorenz values for the same five predictors as a function of the lags of the GRU neural network, while the bottom panel illustrates the RMSE of the GRU as a function of the lagged values.

From Fig. 7 note that the RMSE of the GRU is smaller than that of the LSTM. This justifies why in the previous sections we have chosen a lag of 60 to compare Shapley–Lorenz values across different neural networks.

Analysing in more detail the results from the GRU model, it arises that a lag equal to 15 provides the best performance in terms of the RMSE. At this lag, the USD/EUR exchange rate is the most important factor, followed by the SP500 index and the USD/YUAN exchange rate. Oil and Gold are at the bottom of the factor importance list. This is consistent with the behaviour of the LSTM for a lag set equal to 5. On the other hand, when the lag is increased at 60, the most important factors shift to the USD/YUAN exchange rate and Oil, followed by the SP500 index, the USD/EUR exchange rate, and Gold.

We can summarise the analysis concluding that the Shapley values of the two models share a common pattern: for small lags, the exchange rates and the equity index dominate in importance, while for large lags, the importance of commodities increases, and the influence of SP500 decreases. This behaviour aligns with the longer auto-correlations existing in commodity prices, which are well captured by both the recurrent networks. Additionally, the importance of the predictors tends to become stable as the number of lags increases.

4. Conclusions

In the paper we have proposed explainable AI models for time series of data, which are much employed in finance, using Bitcoin price prediction as a use case.

We have extended the Shapley–Lorenz method to time series machine learning models, which, unlike the more well-known Shapley value method, yields easily interpretable normalised values.

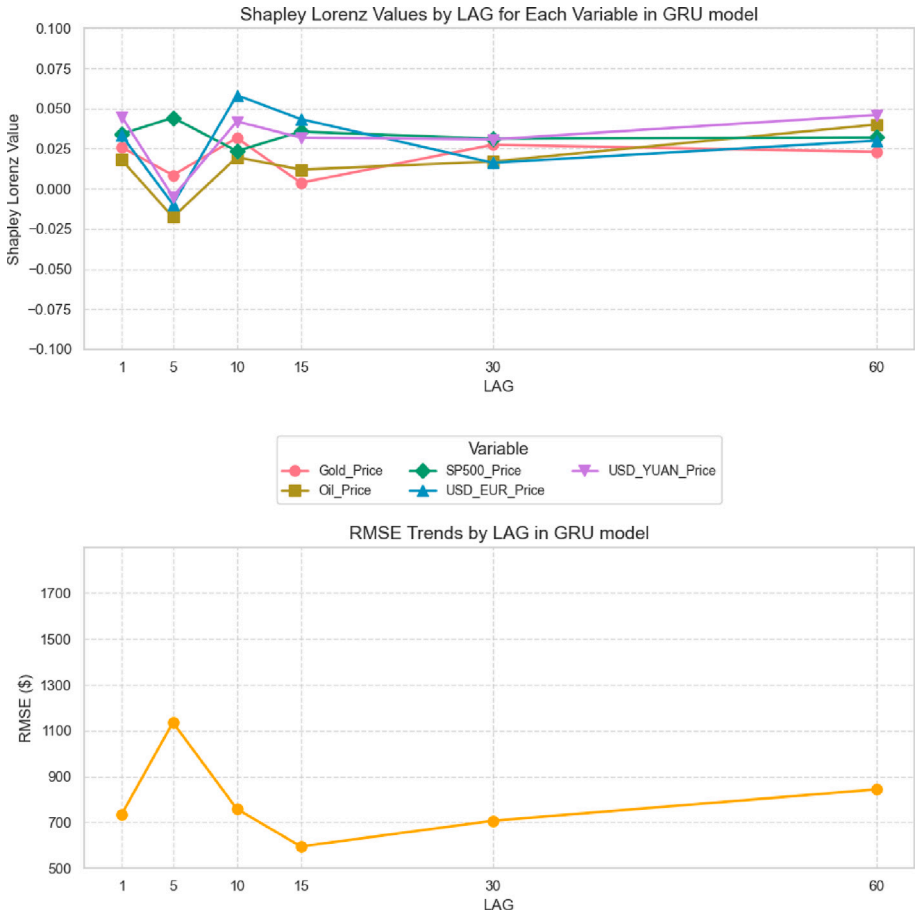


Fig. 7. GRU neural network. Top panel: Shapley–Lorenz values for the five predictors as a function of lags. Bottom panel: Root Mean Square Error (RMSE) as a function of lags.

The extension has allowed us to compare models not only in terms of predictive accuracy, but also in terms of explainability and in terms of robustness, according to the SAFE-AI assessment framework.

The empirical findings have shown that, in terms of accuracy, the GRU and the LSTM are the best models; in terms of robustness, all models, apart from the NNAR, perform well; in terms of explainability, simpler networks, such as MLP and RBF, are the best models. Model choice depends on which criteria is deemed more relevant.

From a financial viewpoint, the application of the Shapley–Lorenz method to our data highlights that the contribution of the classical financial variables to the explanation of the Bitcoin prices is limited, especially for the most accurate models. Such limited degree of explanation is due to the much lower variability of classical assets, with respect to Bitcoin prices. However, in the paper we have shown that, especially using recurrent neural networks, the variability of the explanations is much correlated to the corresponding input variables. This means that the variability of the inputs is exploited, as much as possible, to obtain better predictions.

From an applied viewpoint, three main findings emerge from our empirical analysis. First, to predict Bitcoin prices, recurrent neural networks are more accurate, but less explainable than classic neural networks. Second, Bitcoin prices are mostly affected by their lagged endogenous values, rather than by exogenous financial prices. Third, although limited, the contribution of financial prices to Bitcoin price prediction is well captured by recurrent neural networks.

Furthermore, the analysis of the interactions among lagged values in recurrent networks indicate that the exchange rates and the equity index dominate in explainability, while for larger lags, the importance of commodities increases, and the influence of SP500 decreases.

Future research should include the extension of what proposed and, in particular, of the normalised S.A.F.E. metrics, including Shapley–Lorenz values, to Generative AI models, which extend (forward) recurrent networks with (forward–backward) attention mechanisms.

CRediT authorship contribution statement

Paolo Giudici: Supervision, Methodology. **Alessandro Piergallini:** Writing – original draft, Formal analysis. **Maria Cristina Recchioni:** Writing – review & editing, Supervision, Methodology. **Emanuela Raffinetti:** Validation, Methodology.

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.

Acknowledgements

This study was funded by: the European Union - NextGenerationEU, in the framework of the GRINS- Growing Resilient, INclusive and Sustainable (GRINS PE00000018). The views and opinions expressed are solely those of the authors and do not necessarily reflect those of the European Union, nor can the European Union be held responsible for them. We also thank the Editor and the three anonymous referees for their stimulating comments and remarks.

Data availability

Data will be made available on request.

References

[1] L. Cao, AI in Finance: Challenges, Techniques and Opportunities, Tehnical Report, University of Sidney, 2021.
[2] O. Triebe, N. Laptev, R. Rajagopal, P. Fishwick, AR-NET: A simple auto-regressive neural network for time-series, 2019, See URL https://arxiv.org/abs/1911.12436.
[3] A. Sherstinski, Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network, Physica D 4904 (2020) 132306.
[4] B. Nyunga, P. Rodrigues, K. Sako, Neural networks for financial time series forecasting, Entropy (2022).
[5] L. Shapley, A value for n-person games, Contrib. Theory Games II (1953) 307–317.
[6] P. Giudici, E. Raffinetti, Shapley–Lorenz eXplainable Artificial Intelligence, Expert Syst. Appl. 167 (2021) 114104.
[7] P. Tan, M. Steinbach, A. Karpatne, V. Kumar, Introduction to Data Mining, Pearson, 2018.
[8] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Adv. Neural Inf. Process. Syst. 30 (2017).
[9] C. Molnar, Interpretable machine learning: A guide for making black box models explainable, Artificial Intelligence (2019) 1–38.
[10] P. Bracke, A. Datta, C. Jung, S. Shayak, Machine learning explainability in finance: an application to default risk analysis, 2019, Staff Working Paper No. 816, Bank of England.
[11] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, ACM Comput. Surv. (CSUR) 51 (5) (2018) 1–42.
[12] R. Dwivedi, D. Dave, H. Naik, S. Singhal, R. Omer, P. Patel, B. Qjan, Z. Wen, T. Shah, G. Morgan, R. Ranjan, Explainable AI (XAI): Core ideas, techniques and solutions, ACM Comput. Surv. 106 (2023).
[13] D. Hand, R. Till, A simple generalisation of the area Under the ROC curve for multiple class classification problem, Mach. Learn. 45 (2001) 171–186.
[14] P. Giudici, E. Raffinetti, Lorenz model selection, J. Classification 37 (3) (2020) 754–768.
[15] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning, with Applications in R, Springer, 2021.
[16] P. Giudici, E. Raffinetti, SAFE artificial intelligence in finance, Financ. Res. Lett. 13 (2023) 104088.
[17] Nair, et al., Maximum likelihood uncertainty estimation: Robustness to outliers, 2022, See URL https://arxiv.org/abs/2202.03870.
[18] T. Gneiting, Making and evaluating point MForecasts, J. Amer. Statist. Assoc. 106 (494) (2011) 746–762.
[19] P. Giudici, E. Raffinetti, RGA: a unified approach of predictive accuracy, Adv. Data Anal. Appl. (2024) in press.
[20] L. Quy, et al., A survey on datasets for fairness-aware machine learning, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 12 (3) (2022) e1452.
[21] P. Giudici, M. Centurelli, S. Turchetta, Artificial intelligence risk measurement, Expert Syst. Appl. 235 (2024) 121220.
[22] A. Oglend, T. Kleppe, On the behavior of commodity prices when speculative storage is bounded., J. Econom. Dynam. Control 175 (2017) 52–69.