scientific reports

Check for updates

OPEN

A comparison between machine and deep learning models on high stationarity data

Domenico Santoro[1,5], Tiziana Ciano[2,3,5] & Massimiliano Ferrara[3,4,5]✉

Advances in sensor, computing, and communication technologies are enabling big data analytics by providing time series data. However, conventional models struggle to identify sequence features and forecast accuracy. This paper investigates time series features and shows that some machine learning algorithms can outperform deep learning models. In particular, the problem analyzed concerned predicting the number of vehicles passing through an Italian tollbooth in 2021. The dataset, composed of 8766 rows and 6 columns relating to additional tollbooths, proved to have high stationarity and was treated through machine learning methods such as support vector machine, random forest, and eXtreme gradient boosting (XGBoost), as well as deep learning through recurrent neural networks with long short-term memory (RNN-LSTM) cells. From the comparison of these models, the prediction through the XGBoost algorithm outperforms competing algorithms, particularly in terms of MAE and MSE. The result highlights how a shallower algorithm than a neural network is, in this case, able to obtain a better adaptation to the time series instead of a much deeper model that tends to develop a smoother prediction.

Recent advances in sensor, computing, and communication technologies are primary sources that are rich in providing time series data. Some technical evidence in this direction also arises in decision sciences and economics, particularly in mathematical finance. These advances transform how complex real-world systems are monitored and controlled[1,2]. Time series forecasting is one of the most critical aspects of big data analytics. However, conventional time series forecasting models cannot effectively identify appropriate sequence features, often leading to a lack of forecast accuracy. Time series are generated chronologically and have high dimensionality and temporal dependence. High dimensionality allows for more information about the behavior of the series, but generally, for analysis, it is crucial to consider each time point as one dimension. Instead, temporal dependencies mean that even two numerically identical points can belong to different classes or predict different behaviors. Time series can be divided into single-variable time series and multi-variable time series, secondary to the notice of the number of sampling variables at a given point in time. These combined characteristics make accurate time series prediction very difficult. Time series are statistical recordings of stochastic processes over time, focusing on discrete, equally spaced observations. They have temporal dependence, where the distribution of an observation depends on previous values and are typically analyzed over all non-negative integers. "Stationarity" is a crucial concept in time series, indicating that a series' behavior remains constant over time, despite variations. Stationary series have a well-understood theory and are fundamental to studying time series, although many non-stationary ones are related. Stationarity is an invariant property that means statistical characteristics of a time series remain consistent over time. While it may not be plausible over long periods, it is often assumed in statistical analysis of time series over shorter intervals. There are two definitions of stationarity: weak stationarity, which only considers the covariance of a process, and strict stationarity, which assumes distributions remain invariant over time. Numerous approaches to the prediction of temporal series have been proposed in the literature, including the autoregressive approach[3,4], the autoregressive approach of integrated mobile media[5,6], the support vector machine approach[7,8], and neural network-based approaches[9–11]. Various hybrid approaches have been proposed[12–15]. Deep learning is a new approach that combines non-linear neural networks to obtain a

[1]Department of Economics, Management and Territory, University of Foggia, 71121 Foggia, FG, Italy. [2]Department of Economics and Political Sciences, University of Aosta Valley, 11100 Aosta, AO, Italy. [3]Department of Law, Economics and Human Sciences & Decisions_Lab, University "Mediterranea" of Reggio Calabria, 89125 Reggio Calabria, RC, Italy. [4]Department of Management and Technology, ICRIOS - The Invernizzi Centre for Research in Innovation, Organization, Strategy and Entrepreneurship, Bocconi University, 20136 Milan, MI, Italy. [5]These authors contributed equally: Domenico Santoro, Tiziana Ciano and Massimiliano Ferrara. ✉email: massimiliano.ferrara@unirc.it

multi-dimensional representation of original input[16,17]. It can learn the functionalities of input data, improving accuracy in non-linear and non-static datasets. The use of neural networks in predicting time series has become increasingly frequent thanks to the ever-increasing computational capacity and advanced techniques. Specifically, neural networks based on long short-term memory (LSTM) architecture have become state-of-the-art in the prediction literature thanks to the memory effect. For example, Varnousfaderan and Shibab[18] use different types of LSTM-based networks to predict bird movement for flight planning to minimize collisions, highlighting how the ability to learn long-order dependence in sequence prediction problems allows for very accurate predictions. Sen et al.[19] use neural networks in financial markets to predict asset prices and build an efficient portfolio, demonstrating how LSTM cells are optimal even in the presence of financial data. Zdravković et al.[20] compare different types of LSTM neural networks to predict the fluid temperature in the district heating systems (DHS) supply line, demonstrating how, after an accurate transformation of the dataset, these neural networks can obtain very high prediction accuracy values. Baesmat et al.[21] develop a hybrid approach for prediction in power system operations by combining neural networks with Artificial Bee Colony (ABC) algorithms, thanks to which they can improve network learning procedures and obtain superior results compared to classical models. At the same time, Baesmat and Shiri[22] demonstrate that a curve-fitting approach can outperform the previous neural network-based method. Or Wen and Li[23], that improve the predictive capabilities of the LSTM through the Attention mechanism in a particular model called *LSTM-attention-LSTM* based on encoder-decoder architecture, demonstrating how the latter is more accurate than many vanilla models in the prediction task.

In this paper, we will deeply investigate some time series features, considering some endogenous mathematical aspects that arose from the observations related to a class of big data from a certain library. We will show that implementing some machine learning (ML) algorithms will be more effective concerning a more robust model, as LSTM is usually determined with this issue. For example, Abbasimher et al.[24] propose using an XGBoost regressor on two renewable energy consumption datasets through a two-stage forecasting framework and comparing this algorithm with the main deep learning models. From this analysis, the authors highlight how the XGBoost regressor outperforms its competitors. Alipour and Charandabi[25] use the XGBoost classifier in combination with NLP models to improve price movement prediction, demonstrating how this combination is optimal. Ghasemi and Naser[26] use some ML algorithms such as XGBoost and random forest to predict compressive strength properties for 3D printed concrete mixes, highlighting how these two algorithms obtain excellent results and allow the identification of the most significant features. Qiu and Wang[27] use the K-Means algorithm to perform customer segmentation of customers in the credit card industry, demonstrating how non-complex clustering algorithms can produce excellent results. Additional ML methods, such as Compressed Sensing, are used to study wireless communications in Industrial Internet-of-Things (IIoT) devices[28,29], or Bayesian Learning-based algorithms for channel estimation[30].

In several cases, however, the XGBoost algorithm has been directly compared with LSTM-based neural networks for the prediction task. For example, Frifra et al.[31] propose a comparison between LSTM and XGBoost to predict storm characteristics and occurrence in Western France, highlighting how, in their case, XGBoost is more accurate than LSTM networks. Hu et al.[32] compare the XGBoost algorithm with RNN-LSTM for predicting wind waves, which require more usability than numerical methods inspired by land physics. From the comparison, it is clear that XGBoost generally performs better than RNN-LSTM. Tehranian[33] compares different ML algorithms, such as random forest, XGBoost, probit, and neural networks, in predicting economic recessions by exploiting macroeconomic indicators and market sentiment, highlighting how ML algorithms are the most accurate. Fan et al.[34] analyzing cooling load predictions by comparing ML algorithms with neural networks, highlighting how non-linear models obtain lower performance than XGBoost, although requiring more time for computation. Or Wei et al.[35], which compare different models in the prediction of a heating load of a residential district, from which it is clear that the ML models, such as XGBoost and SVR, are the fastest (in training time) and obtain excellent results on a par with those obtained by the LSTM network. Furthermore, the propensity for ML algorithms that require fewer hyperparameters is also significant.

In many cases, it is evident that the non-linearity of neural network-based models underperforms the model's potential since they often deal with data characterized by stationarity. Some main limitations concern the impossibility of increasing the accuracy beyond a certain threshold. Others, instead, have to do with intrinsic characteristics of the time series. In the latter case, much data is derived from recordings of physical/natural phenomena or related to repeated human activities that appear stationary. So far, many authors have preferred to resort to dataset manipulations to eliminate stationarity, for example, by applying restrictions (where possible) or working with decompositions of the latter to obtain higher accuracy values of DL models. However, it is clear that models characterized by a lower complexity are more accurate in the prediction phase than competitors in this type of time series. The main contributions of this paper are:

- The analysis of the vehicle flows dataset from some Italian tollbooths that highlight highly stationary characteristics;
- The comparison between RNN-LSTM, XGBoost, SVM, and random forest in the prediction task based on the previous dataset through the best hyperparameters' combination;
- The explainability analysis of the best performing algorithm, XGBoost, through the SHAP framework to highlight which are the most significant features.

Road-map of the paper

This article is structured as follows: Sect. " Machine and deep learning algorithms" introduces the machine and deep learning algorithms/models to compare for prediction; Section "Data description" presents the data used from tollbooths and the main characteristics; "Comparison between models" reports the comparisons between

the main hyperparameter combinations of the different algorithms in the prediction task, based on the dataset used and analyzes the explainability of the XGBoost model in terms of feature importance; finally, Sect. "Conclusions" concludes the paper with an overview of the work done, some final remarks and its limitations.

Machine and deep learning algorithms

Nowadays, the algorithms and techniques for time series prediction are increasingly "deep" and performing. However, a task of the same type can be performed with different methods, which leads to results that, in most cases, achieve better accuracy with more information. For example, a DL model widely used for the prediction task is neural networks. An artificial neural network (ANN) is a computational model inspired by the human brain, which comprises artificial neurons[36] that perform computations within them. The key feature of ANNs is the ability to learn, i.e. adapting the network parameters to specific data. A first specific type of ANN is the feedforward neural network (FNN), where connections move in a one-way sequence from one node to the next, like in the Perceptron[37] case. On the other hand, ANNs that can be equipped with feedback connections in which training requires different time instants are called recurrent neural networks (RNNs). The unfolding in time process for training makes these types of networks ideal for data sequences. To train an RNN, considering feedback connections, a particular version of the Backpropagation[38] algorithm is used: the *backpropagation through time* (BPTT), in which the gradients are computed at each time step.

Neural networks suffer from a problem related to the gradient of the loss function to be computed, which leads to the *explosion* or *vanishing* of the gradient and can lead to the interruption of training. To prevent this problem, a particular architecture was introduced: the *Long-short term memory* (LSTM)[39]. This unit uses specific control gates to "decide" which information should be forwarded to the next level. Specifically, the LSTM cell is made up of an *input gate*, an *output gate*, and a *forget gate*. Considering an input X_t and the previous hidden state S_{t-1}, a new state S_t can be described as:

$$\begin{aligned}
f_t &= \sigma(X_t U^f + S_{t-1} W^f + b_f),\\
i_t &= \sigma(X_t U^i + S_{t-1} W^i + b_i),\\
\tilde{C}_t &= \tanh(X_t U^c + S_{t-1} W^c + b_c),\\
C_t &= C_{t-1} \odot f_t + i_t \odot \tilde{C}_t,\\
o_t &= \sigma(X_t U^o + S_{t-1} W^o + b_o),\\
S_t &= o_t \odot \tanh(C_t),
\end{aligned}$$

where σ is the sigmoid activation function, i the identify gate, f the forget gate, o the output gate, C the cell state, U the input weight matrix, W the recurrent weight matrix, b the bias, and $\odot$ represents the Hadamard product. RNN-LSTM represents the newest and most widespread architectures for time series forecasting.

On the other hand, ML algorithms used for prediction are generally more explainable than those of DL's competitors. A first type of proposed model is *support vector machines* (SVMs)[40], initially used for classification, has been extended for the regression task. Specifically, SVM finds the optimum separating hyperplane (OSH) between two classes, and its main objective is to maximize the margin between classes of training samples[41]. Its extension, *support vector regression* (SVR), also called ϵ-SVR, minimize the loss function[42,43]

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\zeta_i + \zeta_i^*) \tag{1}$$

under the following constraints:

$$\begin{cases} y_i - w^T\phi + b \leq \epsilon + \zeta_i \\ w^T\phi + b - y_i \leq \epsilon + \zeta_i^* \\ \zeta_i, \zeta_i^*, \epsilon \geq 0 \end{cases}$$

where w is the weight vector, C is a regularization term, ζ_i, ζ_i^* are slack variables related to prediction error, b is the bias term, ϕ is a map function over the feature space, y_i is the coefficient vector, and ϵ is the error parameter user-defined. In this way, the ϵ-SVR finds the linear function that deviates at most ϵ from the coefficient vector[43]. To improve the separability of the input data, it is possible to apply a kernel function that adds non-linearity and transports them into a higher dimensional space. An example is represented by the *radial basis function* (RBF) between two points, $K(x_1, x_2)$, defined as:

$$K(x_1, x_2) = \exp\left(\frac{||x_1 - x_2||^2}{2\sigma^2}\right),$$

where σ is an hyperparameter.

A different approach from the previous one is to use decision Trees to carry out classification or regression tasks, as in the random forest[44] (RF) case, an ensemble method that uses many trees. These are generated randomly through a training phase on a random sample with a replacement of the training set (*bagging*) and a restriction of the features. Through this mechanism, random forest is also used to identify the most important features by minimizing the *out-of-bag error* (OOB), i.e. the error on values not considered in the sampling process. Furthermore, no form of pruning is applied to the trees[45].

A further evolution in the use of trees is *eXtreme Gradient Boosting* (XGBoost)[46], an iterative algorithm implemented in a boosting library. The main algorithm implemented for learning is the sequential creation of regression trees, the classification and regression tree (CART)[47]. The potential of using decision trees lies in dividing alternatives' space into different subsets based on a measure, a process which, repeated recursively, allows classification rules to be obtained. XGBoost training generates sequential trees to minimize prediction errors[48,49]. Specifically, the objective function to minimize can be divided into two components[50], the error function $L(\cdot)$ and a regularization term $\Omega(\cdot)$:

$$Obj = \sum_i L(y_i, \hat{y}_i) + \Omega(f), \tag{2}$$

where $L(y_i, \hat{y}_i)$ is the loss function for i-th tree with prediction $\hat{y}_i$. Instead, the $\Omega(f)$ is composed:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_j \omega_j^2, \tag{3}$$

where T is the number of leaves whose weight is represented by ω, γ is a learning rate used for pruning, and λ in a regularization parameter. Identifying the optimal branch of the tree in this algorithm occurs through a greedy method, with which the candidate with the highest probability is searched for and continues on that path. Unlike LSTM, XGBoost enjoys much higher explainability due to the classifier's "simplicity" of the decisions obtainable at each level and its high generalizability and computational speed. A popular framework to further improve the explainability of this algorithm (and, in general, machine learning algorithms) is *SHapley Additive exPlanations (SHAP)*[51]. Specifically, SHAP allows explaining each feature's contribution to the model used. This framework is based on a Game Theory approach that measures each player's contribution in a cooperative game, the Shapley value. For a feature x_j, the Shapley value is given by[52]:

$$SHAP(x_j) = \sum_{X \subseteq Y \setminus \{j\}} \frac{k!(p-k-1)!}{p!}(f(Y) - f(X)) \tag{4}$$

where p is the number of features in total feature set Y, $X \subseteq Y \setminus \{j\}$ is the set of features combinations without the j-th, and $f(X)$, $f(Y)$ are the model prediction in different feature sets. A variant for tree-based algorithms is TreeSHAP[53], which is computationally less expensive than the basic framework.

Data description

The prediction tests with LSTM and XGBoost were carried out on a dataset relating to the number of vehicles passing through 5 Italian tollbooths on different days. Sequential numbering indicates the "interest" for each of them, linked, for example, to geographical factors. In this sense, *Tollbooth 1* is of greater interest than *Tollbooth 5* and is the subject of the prediction task. Specifically, the dataset used represents a restriction of the originally collected data, which included a series of additional variables linked to climatic conditions and extended over a longer period. The original dataset, weighing over 250 MB, was reduced to the current version (around 100 MB), containing the hourly data of the vehicles passing through the tollbooths from 1/1/2021 to 12/31/2021 (in US format). For more information related to the Data, see the Acknowledgment at the end of the present work. The dataset comprises 8766 rows and 6 features related to the registration time and the 5 most relevant tollbooths, as shown in Table 1. Figure 1 contains a plot of the different tollbooths, differentiated by color, while Table 2 presents some statistics of this dataset.

Graphically, it is evident how the different time series are characterized by stationarity, in which many hours are characterized by the passage of no vehicles, especially at night, followed by hours of heavy traffic. We have performed, with the *statsmodels* Python module, the *Augmented Dickey-Fuller* (ADF)[54] and *Kwiatkowski-Phillips-Schmidt-Shin* (KPSS)[55] tests to prove it, as shown also in Table 2. Particularly, the ADF tests the null hypothesis H_0, which is that the series presents a unit root against an alternative hypothesis H_1, which is the absence of unit roots. On the other hand, KPSS tests a null hypothesis H_0 of trend-stationarity of the series against an alternative hypothesis H_1 of the presence of a unit root. At a 95% level, the ADF test on the different features demonstrates the stationarity of the latter since the null hypothesis H_0 can be rejected. Similarly, from the KPSS test, it is clear that at a level of 95%, the null hypothesis H_0 can be rejected, highlighting non-stationarity. This contrast between the two tests indicates how the considered time series are *difference-stationary processes*. To highlight stationarity

Date - hour	Tollbooth 1	Tollbooth 2	Tollbooth 3	Tollbooth 4	Tollbooth 5
2021-01-01 - 00:00	10	4	0	1	0
2021-01-01 - 01:00	1	8	0	0	0
2021-01-01 - 02:00	0	5	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
2021-12-31 - 22:00	20	135	7	9	2
2021-12-31 - 23:00	16	116	8	5	3

Table 1. Sample of the dataset used (values from beginning and ending dates).

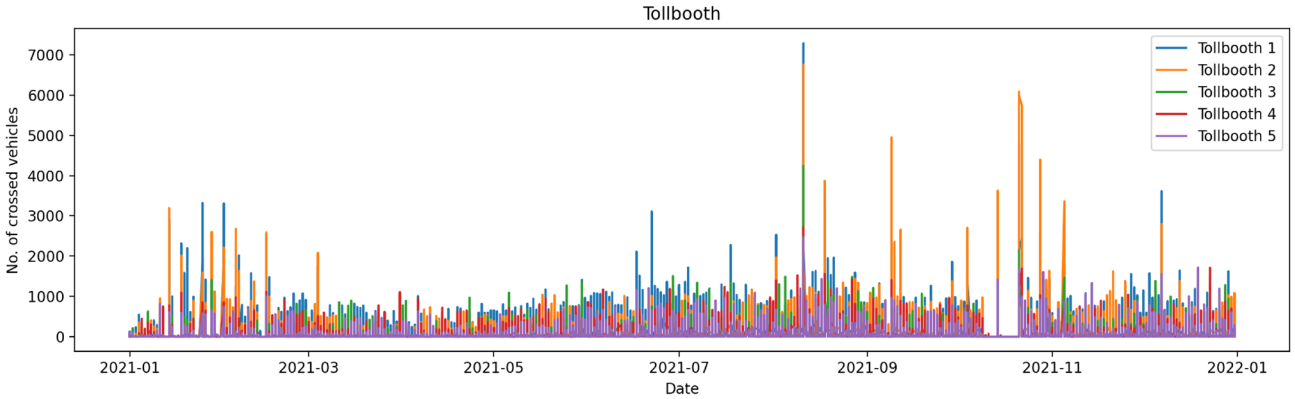


Fig. 1. Dataset features plot indexed by hours.

Feature	Mean	St. dev	Min	25%	50%	75%	Max	ADF (p-val)	KPSS (p-val)
Tollbooth 1	193.20	325.79	0.0	7.0	82.0	255.0	7285.0	−13.65 (0.0)	2.64 (0.01)
Tollbooth 2	197.23	352.67	0.0	7.0	96.0	280.0	6750.0	−13.29 (0.0)	4.59 (0.01)
Tollbooth 3	53.46	140.72	0.0	0.0	12.0	47.0	4241.0	−14.22 (0.0)	3.19 (0.01)
Tollbooth 4	68.64	145.20	0.0	1.0	19.0	75.0	2715.0	−13.86 (0.0)	3.31 (0.01)
Tollbooth 5	45.36	125.13	0.0	0.0	8.0	32.0	2469.0	−13.97 (0.0)	3.84 (0.01)

Table 2. Dataset main statistics, ADF and KPSS tests.

through the KPSS test, a new series can be built by differencing between different time-step observations, as shown in Fig. 2, bringing the results of the two tests into a common agreement.

Comparison between models

We want to test the predictive capabilities of SVM, Random Forest, XGBoost, and RNN-LSTM on the *Tollbooth 1* feature. However, unlike Machine Learning models, RNN-LSTM needs to reshape the dataset size by considering a sliding window to "look back", which is why several attempts have been made with a maximum window of 24 hours in the past, from which the dimensionality tensor has a maximum size equal to (7865, 24, 4). All analyses were performed through *Python*, and the scaler used in all cases is the *StandardScaler*. We have set the LSTM network structure with a maximum of 5 input layers with several neurons from 1 to 30, and 1 output layer with 1 neuron only given the one output feature. Given the data type, adding excessive complexity to the network was inappropriate. Table 3 shows the remaining hyperparameters, which control the learning process. On the other hand, from the ML algorithm side, also Table 3 shows the hyperparameters for XGBoost, ϵ-SVR, and Random Forest. Particularly, to best adapt them to the dataset type, a GridSearchCV was applied to select the best combination of hyperparameters. For XGBoost and Random Forest, several tests were carried out by modifying the max depth of the trees and number of estimators, while for ϵ-SVR, the substantial change concerns the type of kernel used. The dataset was divided into a training set (80%) and a test set (20%). The size of the test set is different because the RNN-LSTM considers a 3D tensor to be the size of the training set, reducing the

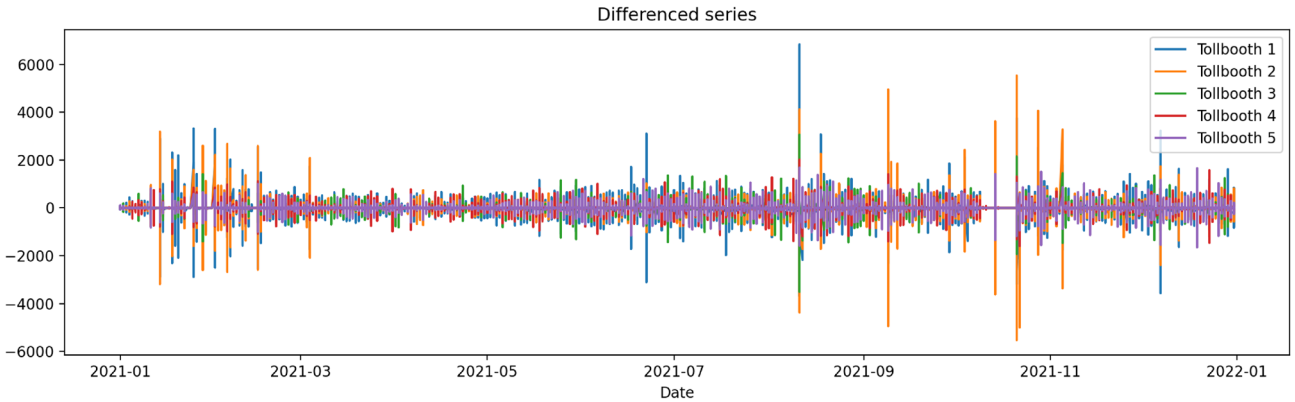


Fig. 2. Differenced series.

RNN-LSTM	Values	XGBoost	Values	SVR	Values	Random forest	Values
Layers	From 2 to 5	Max depth	(1, 6, 30)	Kernel	(Linear, rbf)	N_estimators	(10, 100)
N. of neurons	From 1 to 30	Subsample	1	Gamma	Scale	Max_depth	None
Activ. func.	Sigmoid	Tree method	Exact	Tol.	Default	Bootstrap	True
Learning rate	0.0005	Sampling	Uniform	C	From 0.01 to 5	Oob_score	True
Optimizer	Adam	Grow policy	Lossguide	Epsilon	From 0.1 to 1	Max_leaf_nodes	None
Batch size	32	Min split loss	0.005	Coef0	0	Warm_start	False
Epochs	300	Learning rate	0.3	Max_iter	− 1	Min_samples_split	2
Time step	3	Lambda	1	Shrinking	True	Min_samples_leaf	1

Table 3. Hyperparameters of different models.

number of observations allocated to the test set. In contrast, the other machine learning algorithms consider a 2D array. Table 4 compares the different RNN-LSTM and the machine learning algorithms in terms of mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), and R^2 of the prediction. These metrics are calculated as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|,$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

$$R^2 = 1 - \frac{\sum_{i^1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})},$$

where y_i represents the observed data, $\hat{y}_i$ is the predicted ones, and $\bar{y}$ is the average value of each features. Greater attention is paid to the MAE and MSE, where the lower values indicate a better model performance in the prediction phase. Specifically, for the LSTM, there is a description of layers and neurons per layer that minimizes the MAE at the best sliding window value obtained (in most cases, equal to 24). The notation used to describe LSTM networks is $LSTM_{layers:\{neurons\ per\ layer\}}$, for XGBoost is $XGBoost_{max\_depth}$, for SVM is $SVM_{kernel;C;\epsilon}$, and for random forest is $RF_{n\_estimators}$. For example, an LSTM network with 3 layers and 1 neuron in the first layer, 10 in the second, and 1 in the last layer will be indicated as $LSTM_{3:\{1,10,1\}}$. Table 4 presents, in bold, the best values

Model		MAE	MSE	RMSE	R^2
$LSTM_{2:\{2,1\}}$		**0.3846**	**0.3368**	**0.5803**	**0.1812**
$LSTM_{2:\{4,1\}}$		0.4012	0.3416	0.5844	0.1256
$LSTM_{3:\{1,10,1\}}$		0.5141	0.4011	0.6333	0.0171
$LSTM_{3:\{5,10,1\}}$		0.4369	0.3661	0.6050	0.0971
$LSTM_{4:\{1,5,10,1\}}$		0.4611	0.3901	0.6245	0.0101
$LSTM_{4:\{5,15,30,1\}}$		0.4763	0.3961	0.6294	0.0085
$LSTM_{5:\{10,20,5,2,1\}}$		0.4311	0.3727	0.6104	0.1041
⟶	$XGBoost_1$	0.3391	0.5750	0.7583	0.4520
	$XGBoost_6$	**0.2679**	**0.3091**	**0.3559**	**0.7058**
	$XGBoost_{30}$	0.2801	0.3416	0.5845	0.6159
$SVR_{lin;1;0.5}$		0.5038	0.9957	0.9978	0.5077
$SVR_{lin;0.01;0.1}$		**0.3457**	**0.8211**	**0.9061**	**0.6162**
$SVR_{rbf;1;0.1}$		0.4529	1.4783	1.2158	0.2881
$SVR_{rbf;5;0.5}$		0.4069	1.1624	1.0781	0.4236
RF_{10}		**0.2885**	**0.3412**	**0.5841**	**0.6453**
RF_{100}		0.2973	0.3624	0.6020	0.5945
RF_{500}		0.3064	0.3411	0.6209	0.5853

Table 4. Comparison between deep and machine learning algorithms (MAE and MSE lower the better). Significant values are in bold.

relating to the metrics considered. Specifically, the combination of hyperparameters has been identified for each model to obtain more performing metrics through Cross-Validation. However, further values are reported to show how the accuracy is drastically reduced with minimal variations in the hyperparameters. A first piece of evidence from the MAE and MSE values from the LSTM network is that a relatively simple model (consisting of 2 layers and 3 neurons in total) obtains the best results compared to evolutions with multiple states and neurons. Information of this type pushes us to test the prediction with less complex models, from which we see how XGBoost obtains the best performance among all the models considered, almost on par with random forest. XGBoost's advantage over the latter is boosting, but the use of Decision Trees allows, in both cases, the building of very high-performance models. Further evidence of the prevalence of "simple" models compared to more complex ones can be observed from the ϵ-SVR. In this model, using a *linear* kernel produces a model with the lowest MAE compared to an RBF-type kernel. The latter allows the addition of non-linearity to the model, which, as highlighted for RNN-LSTM networks, does not bring any advantage to this data type. The high stationarity of the data makes it difficult to add non-linearity to extract more information, from which a more explainable and branched algorithm like XGBoost manages to outperform a complex model like LSTM. Figure 3 shows an example of prediction on the 200-hour test set of the best models ($LSTM_{2:\{2,1\}}$, XGBoost_6, $\text{SVR}_{lin;0.01;0.1}$, and RF_{10}). Specifically, even graphically, we can see that the prediction with LSTM tends to be less stationary and smoother while maintaining the prediction around a trend. At the same time, XGBoost optimally adapts the detrended predicted series to the original one, which is the best choice for a prediction with this type of time series. A similar behavior is adopted by Random Forest (which still uses Decision Trees) but achieves lower performance than XGBoost.

Depending on the results, it may be interesting to transfer the characteristics of this specific model to other domains. Transfer learning (TL) allows the transfer of information from a source domain to a target domain, such as information on instances, parameters, and feature characteristics[56]. In this case, the TL can be used on the influx of vehicles at motorway tollbooths for which there is a lot of missing data due to malfunctions. Although having the same data distribution is difficult, it is still possible to benefit from very accurate predictive models.

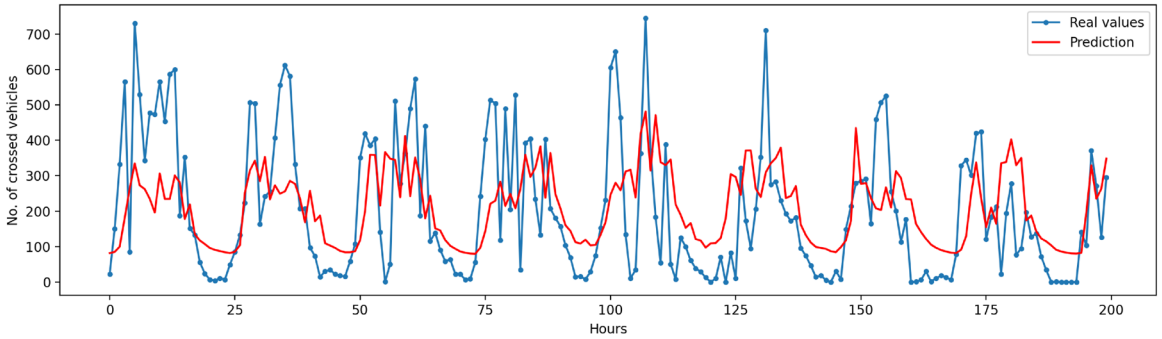
Going into explainability in detail, the SHAP framework allows us to study the importance of the different features in the prediction phase. In this case, the idea is to use it on the XGBoost algorithm, which has outperformed its competitors. Considering *Tollbooth 1* as the target feature, as shown in Fig. 4b, the most important feature that affects the prediction is *Tollbooth 3* linked to the highest Shapley value, followed by *Tollbooth 4*. The distribution can explain this relationship over time of vehicles that passed through *Tollbooths* 2 to 5. Assuming that *Tollbooth 1* is the one of greatest interest to travelers and absorbs the greatest number of vehicles that pass through at different times of the day, different types of users use the remaining toll booths. In this case, *Tollbooth 3* has a distribution of vehicles very similar to *Tollbooth 1* at different times of the day, albeit with a much smaller number of vehicles, which is why it is the feature that most influence the model. Instead, *Tollbooth 2*, despite having a very high average number of vehicles passed through (on a par with *Tollbooth 1*), has a different temporal distribution, which makes it a feature characterized by minimal importance and almost on a par with *Tollbooth 5*. The summary plot, however, present in Fig. 4a, allows us to illustrate different Shapley values as the instances vary, considering the increase in feature values depending on the color intensity of each point. Specifically, the high values achieved by the different features that impact the model correspond to increasingly higher Shapley values and, consequently, higher predicted values (in terms of vehicles passed through). Although not the most important, the *Tollbooth 2* feature reaches higher values (regarding vehicles passed through), pushing towards an increase in the Shapley value. This analysis through the SHAP framework shows that the dataset used, although characterized by few features, has optimal characteristics since no feature has a zero magnitude. Therefore, all the features impact the final model, although some in a limited way compared to others.

Conclusions

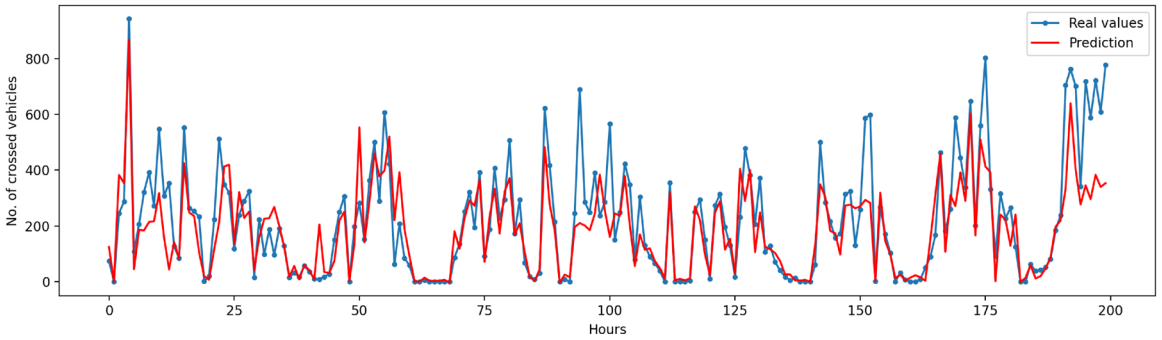
Time series prediction represents a fundamental task in many sectors. However, the presence of stationary data is still challenging, especially if the prediction is carried out using deep learning techniques. This work considers data from motorway tollbooths characterized by high stationarity. Here, a series of comparisons were made between machine and deep learning algorithms. Specifically, RNN-LSTM, XGBoost, ϵ-SVR, and Random Forest.

The results highlight how XGBoost outperformed the algorithms for prediction on data with these characteristics, obtaining the best results in terms of MAE, MSE, RMSE, and R^2 is clear how the Deep Learning models tend to neutralize the excessive number of peaks in the time series considered, producing a smoother prediction but not corresponding to reality. Using machine learning algorithms such as XGBoost is preferable to more complex models.

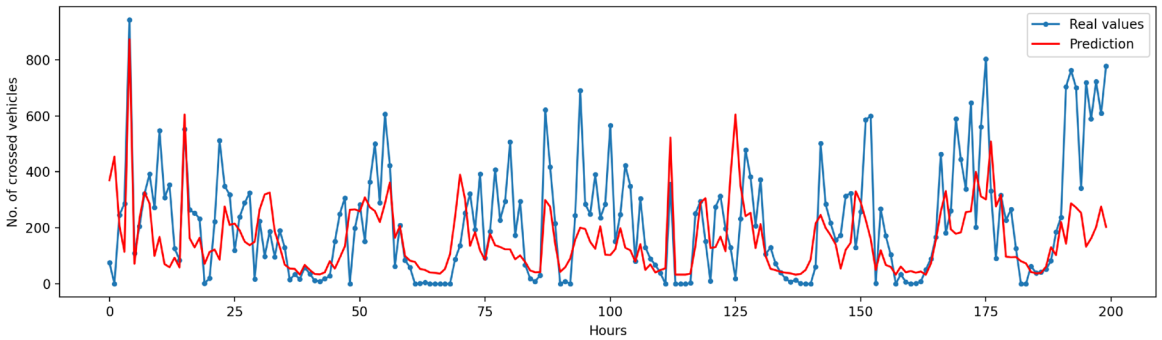
The advantage of this result is the possibility of using a computationally less expensive algorithm on this highly stationary data since XGBoost does not require the use of a large number of parameters like an LSTM neural network. Furthermore, using a CART-based algorithm like XGBoost allows us to benefit from a certain degree of explainability of what contributed to the model's performance. However, using a machine learning algorithm can be seen as a limitation since, in a historical moment in which deep learning models achieved extraordinary performance in many areas, this demonstrates the ineffectiveness of neural networks on data with extreme characteristics such as high stationarity. A further limitation concerns the explainability of the phenomenon since it is possible to identify which are the most essential features. Still, due to the strong peaks in the data, it remains challenging to understand which are the most significant patterns that can be used for prediction.



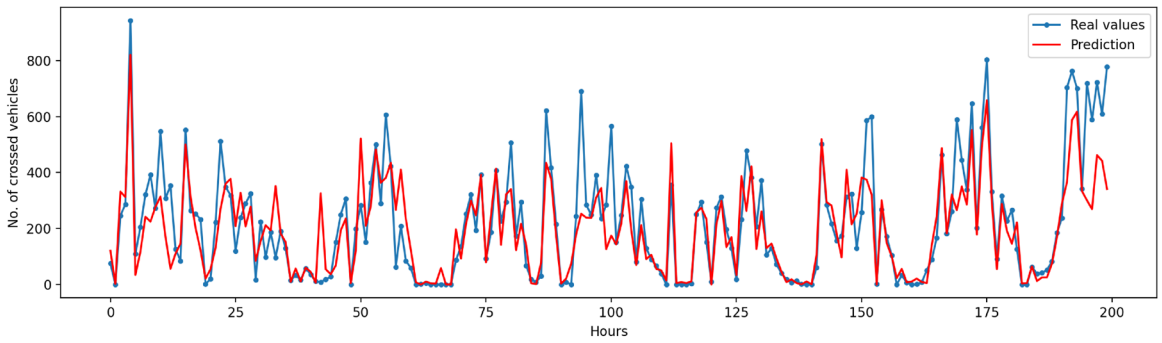
(a) $LSTM_{2;\{2,1\}}$



(b) $XGBoost_6$

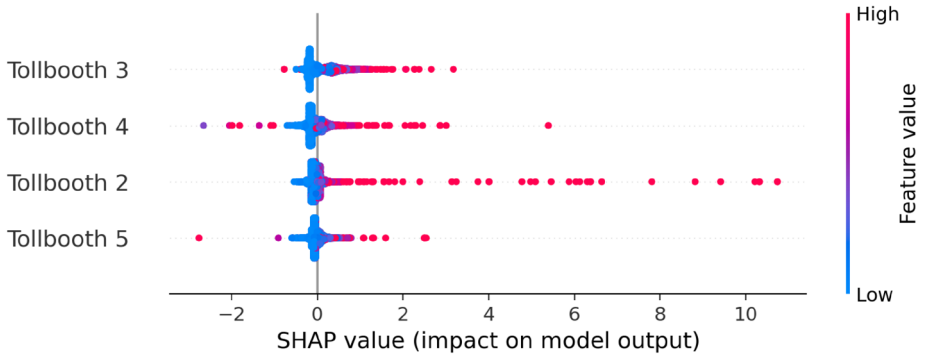


(c) $SVR_{lin;0.01;0.1}$

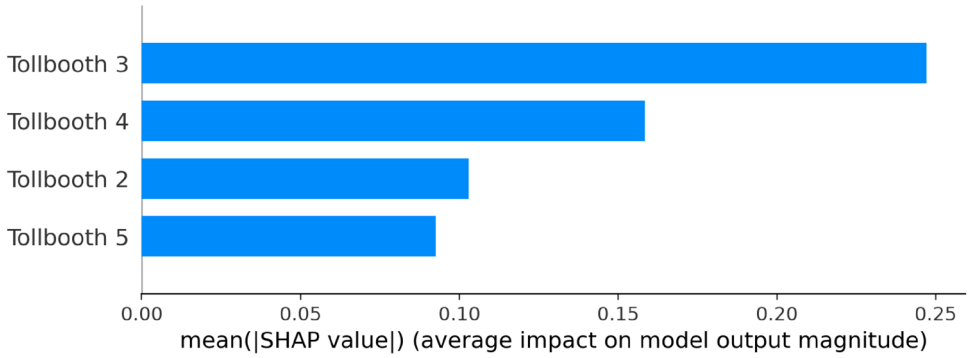


(d) RF_{10}

Fig. 3. Comparison between different models.



(a) Directionality impact of each feature



(b) Model influence for each feature

Fig. 4. Feature importance summary using SHAP.

Data availability

The datasets generated and/or analyzed during the current study are not publicly available since they belong in full to the MONTUR Project still under development (see Acknowledgments), but are available from the corresponding author on reasonable request.

Received: 15 April 2024; Accepted: 14 August 2024
Published online: 21 August 2024

References

1. Cheng, C. *et al.* Time series forecasting for nonlinear and non-stationary processes: A review and comparative study. *IIE Trans.* **47**, 1053–1071 (2015).
2. Schober, P. *et al.* Stochastic computing design and implementation of a sound source localization system. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **13**, 295–311. https://doi.org/10.1109/JETCAS.2023.3243604 (2023).
3. Akaike, H. Fitting autoregreesive models for prediction. In *Selected Papers of Hirotugu Akaike* (ed. Akaike, H.) 131–135 (Springer, 1969).
4. Hurvich, C. M. & Tsai, C.-L. Regression and time series model selection in small samples. *Biometrika* **76**, 297–307 (1989).
5. Box, G. E. & Pierce, D. A. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *J. Am. Stat. Assoc.* **65**, 1509–1526 (1970).
6. Williams, B. M., Durvasula, P. K. & Brown, D. E. Urban freeway traffic flow prediction: Application of seasonal autoregressive integrated moving average and exponential smoothing models. *Transp. Res. Rec.* **1644**, 132–141 (1998).
7. Cao, L.-J. & Tay, F. E. H. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Trans. Neural Netw.* **14**, 1506–1518 (2003).
8. Müller, K.-R. *et al.* Predicting time series with support vector machines. In *International conference on artificial neural networks*, 999–1004 (Springer, 1997).
9. Zhang, G. P. & Berardi, V. L. Time series forecasting with neural network ensembles: An application for exchange rate prediction. *J. Oper. Res. Soc.* **52**, 652–664 (2001).
10. Noel, M. M. & Pandian, B. J. Control of a nonlinear liquid level system using a new artificial neural network based reinforcement learning approach. *Appl. Soft Comput.* **23**, 444–451 (2014).
11. Chen, Y., Yang, B. & Dong, J. Time-series prediction using a local linear wavelet neural network. *Neurocomputing* **69**, 449–465 (2006).
12. Zhang, G. P. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing* **50**, 159–175 (2003).
13. Jain, A. & Kumar, A. M. Hybrid neural network models for hydrologic time series forecasting. *Appl. Soft Comput.* **7**, 585–592 (2007).
14. Aladag, C. H., Egrioglu, E. & Kadilar, C. Forecasting nonlinear time series with a hybrid methodology. *Appl. Math. Lett.* **22**, 1467–1470 (2009).

15. Maguire, L. P., Roche, B., McGinnity, T. M. & McDaid, L. Predicting a chaotic time series using a fuzzy neural network. *Inf. Sci.* **112**, 125–136 (1998).
16. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
17. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **61**, 85–117 (2015).
18. Varnousfaderani, E. S. & Shihab, S. A. M. Bird movement prediction using long short-term memory networks to prevent bird strikes with low altitude aircraft. *AIAA Aviat. 2023 Forum*https://doi.org/10.2514/6.2023-4531.c1 *(2023).*
19. Sen, J., Dutta, A. & Mehtab, S. Stock portfolio optimization using a deep learning lstm model. *2021 IEEE Mysore Sub Section International Conference (MysuruCon)* 263–271, https://doi.org/10.1109/MysuruCon52639.2021.9641662 (2021).
20. Zdravković, M., Ćirić, I. & Ignjatović, M. Explainable heat demand forecasting for the novel control strategies of district heating systems. *Annu. Rev. Control.* **53**, 405–413. https://doi.org/10.1016/j.arcontrol.2022.03.009 (2022).
21. Baesmat, K. H., Masoudipour, I. & Samet, H. Improving the performance of short-term load forecast using a hybrid artificial neural network and artificial bee colony algorithm amélioration des performances de la prévision de la charge à court terme à l'aide d'un réseau neuronal artificiel hybride et d'un algorithme de colonies d'abeilles artificielles. *IEEE Can. J. Electr. Comput. Eng.* **44**, 275–282. https://doi.org/10.1109/ICJECE.2021.3056125 (2021).
22. Baesmat, H. K. & Shiri, A. A new combined method for future energy forecasting in electrical networks. *Int. Trans. Electr. Energy Syst.* **29**, e2749. https://doi.org/10.1002/etep.2749 (2019) (**E2749 ITEES-17-0407.R4**).
23. Wen, X. & Li, W. Time series prediction based on lstm-attention-lstm model. *IEEE Access* **11**, 48322–48331. https://doi.org/10.1109/ACCESS.2023.3276628 (2023).
24. Abbasimehr, H., Paki, R. & Bahrini, A. A novel xgboost-based featurization approach to forecast renewable energy consumption with deep learning models. *Sustain. Comput. Inform. Syst.* **38**, 100863. https://doi.org/10.1016/j.suscom.2023.100863 (2023).
25. Alipour, P. & Esmaeilpour Charandabi, S. The impact of tweet sentiments on the return of cryptocurrencies: Rule-based vs. machine learning approaches. *Eur. J. Bus. Manag. Res.* **9**, 1–5. https://doi.org/10.24018/ejbmr.2024.9.1.2180 (2024).
26. Ghasemi, A. & Naser, M. Tailoring 3d printed concrete through explainable artificial intelligence. *Structures* **56**, 104850. https://doi.org/10.1016/j.istruc.2023.07.040 (2023).
27. Qiu, Y. & Wang, J. A machine learning approach to credit card customer segmentation for economic stability. In *Proc. of the 4th International Conference on Economic Management and Big Data Applications, ICEMBDA 2023, October 27–29, 2023, Tianjin, China*[SPACE]https://doi.org/10.4108/eai.27-10-2023.2342007 (2024).
28. Wang, H. *et al.* Machine learning-enabled mimo-fbmc communication channel parameter estimation in iiot: A distributed cs approach. *Digit. Commun. Netw.* **9**, 306–312. https://doi.org/10.1016/j.dcan.2022.10.012 (2023).
29. Wang, H., Xu, L., Yan, Z. & Gulliver, T. A. Low-complexity mimo-fbmc sparse channel parameter estimation for industrial big data communications. *IEEE Trans. Ind. Inf.* **17**, 3422–3430. https://doi.org/10.1109/TII.2020.2995598 (2021).
30. Wang, H. *et al.* Sparse Bayesian learning based channel estimation in fbmc/oqam industrial iot networks. *Comput. Commun.* **176**, 40–45. https://doi.org/10.1016/j.comcom.2021.05.020 (2021).
31. Frifra, A., Maanan, M., Maanan, M. & Rhinane, H. Harnessing lstm and xgboost algorithms for storm prediction. *Sci. Rep.*https://doi.org/10.1038/s41598-024-62182-0 *(2024).*
32. Hu, H., van der Westhuysen, A. J., Chu, P. & Fujisaki-Manome, A. Predicting lake erie wave heights and periods using xgboost and lstm. *Ocean Model.* **164**, 101832. https://doi.org/10.1016/j.ocemod.2021.101832 (2021).
33. Tehranian, K. Can machine learning catch economic recessions using economic and market sentiments? http://arxiv.org/abs/2308.16200v1 (2023).
34. Fan, C., Xiao, F. & Zhao, Y. A short-term building cooling load prediction method using deep learning algorithms. *Appl. Energy* **195**, 222–233. https://doi.org/10.1016/j.apenergy.2017.03.064 (2017).
35. Wei, Z. *et al.* Prediction of residential district heating load based on machine learning: A case study. *Energy* **231**, 120950. https://doi.org/10.1016/j.energy.2021.120950 (2021).
36. McCullock, W. S. & Pitts, W. H. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **5**, 115–133 (1943).
37. Rosenblatt, F. The percepron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **65**, 386 (1958).
38. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representation by back-propagation errors. *Nature*https://doi.org/10.1038/323533a0 *(1986).*
39. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 (1997).
40. Vapnik, V. N. *The Nature of Statistical Learning Theory* (Springer, 1995).
41. Ciano, T. & Ferrara, M. Karush-kuhn-tucker conditions and lagrangian approach for improving machine learning techniques: A survey and new developments. *Atti della Accademia Peloritana dei Pericolanti - Classe di Scienze Fisiche, Matematiche e Naturali* **102**, 1. https://doi.org/10.1478/AAPP.1021A1 (2024).
42. Sabzekar, M. & Hasheminejad, S. M. H. Robust regression using support vector regressions. *Chaos Solitons Fractals* **144**, 110738. https://doi.org/10.1016/j.chaos.2021.110738 (2021).
43. Klopfenstein, Q. & Vaiter, S. Linear support vector regression with linear constraints. *Mach. Learn.* **110**, 1939–1974. https://doi.org/10.1007/s10994-021-06018-2 (2021).
44. Breiman, L. Random forests. *Mach. Learn.* **45**, 5–32. https://doi.org/10.1023/A:1010933404324 (2001).
45. Biau, G. Analysis of a random forests model. *J. Mach. Learn. Res.* **13**, 1063–1095 (2012).
46. Chen, T. & Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 785–794, https://doi.org/10.1145/2939672.2939785 (Association for Computing Machinery, 2016).
47. Breiman, L., Friedman, J., Olshen, R. & Stone, C. J. *Classification and Regression Trees* (Chapman and Hall/CRC, 1984).
48. Li, S. & Zhang, X. Research on orthopedic auxiliary classification and prediction model based on xgboost algorithm. *Neural Comput. Appl.* **32**, 1971–1979. https://doi.org/10.1007/s00521-019-04378-4 (2020).
49. Mohril, R. S., Solanki, B. S., Kulkarni, M. S. & Lad, B. K. Xgboost based residual life prediction in the presence of human error in maintenance. *Neural Comput. Appl.* **35**, 3025–3039. https://doi.org/10.1007/s00521-022-07216-2 (2022).
50. Mustapha, I. B., Abdulkareem, Z., Abdulkareem, M. & Ganiyu, A. Predictive modeling of physical and mechanical properties of pervious concrete using xgboost. *Neural Comput. Appl.*https://doi.org/10.1007/s00521-024-09553-w *(2024).*
51. Lundberg, S. M. & Lee, S.-I. A unified approach to interpreting model predictions. In *Proc. of the 31st International Conference on Neural Information Processing Systems* 4768–4777 (2017).
52. Li, Z. Extracting spatial effects from machine learning model using local interpretation method: An example of shap and xgboost. *Comput. Environ. Urban Syst.* **96**, 101845. https://doi.org/10.1016/j.compenvurbsys.2022.101845 (2022).
53. Lundberg, S. M., Erion, G. G. & Lee, S.-I. Consistent individualized feature attribution for tree ensembles. http://arxiv.org/abs/1802.03888 (2018).
54. Dickey, D. A. & Fuller, W. A. Distribution of the estimators for autoregressive time series with a unit root. *J. Am. Stat. Assoc.* **74**, 427–431. https://doi.org/10.2307/2286348 (1979).
55. Kwiatkowski, D., Phillips, P. C., Schmidt, P. & Shin, Y. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?. *J. Econom.* **54**, 159–178. https://doi.org/10.1016/0304-4076(92)90104-Y (1992).

56. Liu, W., Liu, W. D. & Gu, J. Predictive model for water absorption in sublayers using a joint distribution adaption based xgboost transfer learning method. *J. Petrol. Sci. Eng.* **188**, 106937. https://doi.org/10.1016/j.petrol.2020.106937 (2020).

Acknowledgements
The authors acknowledge the University of Aosta Valley, in particular the Department of Economics and Political Sciences by the CT-TEM UNIVDA - Centro Transfrontaliero sul Turismo e l'Economia di Montagna and their Director. Prof. Marco Alderighi for their support through the MONTUR Project. A part of Data testing was developed by using "Real Time Series" extrapolated by the mentioned project and will be adopted as the basis for future work. The present work defines the crucial and pivotal structural elements of the Decision Support Systems will be developed into the MONTUR Project. The Authors thank equally the Decisions LAB - Department of Law, Economics and Human Sciences - University Mediterranea of Reggio Calabria for its support to the present research. Funded by European Union- Next Generation EU, Component M4C2, Investment 1.1., Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) - Notice 1409, 14/09/2022- BANDO PRIN 2022 PNRR. Project title: "Climate risk and uncertainty: environmental sustainability and asset pricing". Project code "P20225MJW8", CUP: E53D23016470001.

Author contributions
M.F. and T.C. conceptualization, M.F. and T.C. data acquisition, M.F. and T.C. conceived the experiment(s), D.S. conducted the experiment(s), D.S., T.C., and M.F. analyzed the results and selected the models, M.F. project administration. All authors reviewed the manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to M.F.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.