Explaining Crash Predictions on Multivariate Time Series Data

Francesco Spinnato[1], Riccardo Guidotti[2], Mirco Nanni[3], Daniele Maccagnola[4], Giulia Paciello[4], and Antonio Bencini Farina[4]

[1] Scuola Normale Superiore, Pisa, Italy, francesco.spinnato@sns.it
[2] University of Pisa, Pisa, Italy, riccardo.guidotti@unipi.it
[3] ISTI-CNR, Pisa, Italy, mirco.nanni@isti.cnr.it
[4] Generali Italia, {name.surname}@generali.com

Abstract. The prediction of human mobility is enabled by the large and diversified availability of data. Insurance companies leverage such spatio-temporal information to develop effective deep learning-based approaches to provide top-quality services to their customers. In Assicurazioni Generali, an automatic decision-making model is used to check real-time multivariate time series and alert if a car crash happened. In such a way, a Generali operator can call the customer to provide first assistance. The high sensitivity of the model used, combined with the fact that the model is not interpretable, might cause the operator to call customers even though a car crash did not happen but only due to a harsh deviation or the fact that the road is bumpy. Our goal is to tackle the problem of interpretability for car crash prediction and propose an eXplainable Artificial Intelligence (XAI) workflow that allows gaining insights regarding the logic behind the deep learning predictive model adopted by Generali. We reach our goal by building an interpretable alternative to the current obscure model that also reduces the training data usage and the prediction time.

Keywords: Multivariate Time Series · Crash Prediction · Explainability · Interpretable Machine Learning · Car Insurance · Case Study

1 Introduction

The availability of real-time sequential data paired with accurate Artificial Intelligence (AI) decision-making systems is changing the business landscape for many mobility-related companies. Since the 1970s, Crash Data Recorders (CDR) have been increasingly used inside cars to monitor safety measures, establish human tolerance limits, and record impact speeds [36]. Lately, through the usage of powerful machine learning models, these devices are becoming a valuable source of data that can be used both for academic research purposes and for businesses, such as insurance companies, to monitor and improve customer service quality [35]. These recorders are usually installed on the airbag control module, collecting data over the period before and after the crash [36]. In our work, we collaborate

with Assicurazioni Generali, Generali in short, and we rely on their multivariate time series data and on their AI system based on a deep learning model to detect car crashes. Generali is one of the largest global insurance and asset management providers. At the heart of Generali strategy is its lifetime partner commitment to customers, achieved through innovative and personalized solutions, best-in-class customer experience and its digitized global distribution capabilities. To fulfil its goals in the car mobility field, Generali is developing an automatic AI-based decision-making system to provide first assistance to its customers. Through ad-hoc CDR, Generali records speed and acceleration as multivariate time series from each insured customer's car. Such data is used to train a deep learning model that enables the AI system to warn a Generali operator of possible car crashes. In turn, the operator will physically call the customer to check if something bad happened and to know which kind of assistance is required.

Two weaknesses are present in the current state. First, the high sensitivity of the AI system might cause unnecessary and harassing calls. Second, the AI system is based on a deep learning model that is inherently not interpretable. An interpretability layer is helpful for numerous reasons. From an ethical standpoint, as outlined in the Artificial Intelligence Act proposal by the European Commission [7], eXplainable Artificial Intelligence (XAI) can help in building user trust toward more transparent AI decisions. This trust can also lead to a competitive advantage, given that a consumer would prefer companies with a proven track record of explainable and trustworthy AI. Moreover, from a governance standpoint, an explainability layer is useful to optimize model performance and include a human in the loop. XAI is a branch of AI that focuses on allowing human users to comprehend and trust the decisions taken by complex black-box models used by AI systems. XAI is becoming more and more popular, and it is used to shed light on the accurate but otherwise opaque AI decision-making while supporting experts, their accountability, and responsibility in their decisions [9].

The objective of this work is to tackle the problem of interpretability for car crash prediction, proposing a pipeline that allows to gain insights regarding the logic behind a deep Convolutional Neural Network (CNN) classifier and build a more transparent predictive model. We use state-of-the-art XAI approaches to address this problem on a large, multivariate time series dataset. XAI for multivariate time series is still an underexplored topic. Since we aim to explain Neural Network (NN) black-box models, our interest is focused on post-hoc model-specific approaches, i.e., on XAI methods designed for NN that are only responsible for the explanation and leave the decision to the original model. Among them, the methods proposed in [3,2] are based on Grad-CAM [30] and can analyze the gradients of CNNs to output understand the most important features in the time series. However, to the best of our knowledge, none of the aforementioned approaches was tested on large multivariate time series with signals of different lengths. Furthermore, they can only be applied to CNNs, which would limit the proposed framework's expandability. For these reasons, our work makes use of GradientExplainer, a model-specific implementation of SHAP [22] which can deal with any NN model and with signals of different lengths, ensuring fast feature-

based explanations. In particular, we use GradientExplainer to distinguish the main areas of attention in the time series data. Then, using this information, we reduce the dimensionality of the dataset and train subsequence-based surrogate trees to imitate the prediction of the NN in a more interpretable way. Finally, we train the best performing surrogate tree as a possible interpretable predictor to be used as a replacement of the CNN. Preliminary results show that the proposed XAI workflow is promising *(i)* in terms of efficiency, as it reduces the data usage and cuts the prediction time, and *(ii)* in terms of effectiveness, providing an interpretability layer that helps operators better understand the prediction of the AI system.

2 Related Work

The literature on crash prediction is broad and studies car accidents from various perspectives. At a high level, we distinguish between *real-time* and *long-term* crash prediction.

Long-term crash prediction is a relatively little explored area, with only a few works in the literature. In [33] a machine learning method is applied to predict the users' driving behaviors based on movement statistics. Wang et al. extend standard approaches consisting of global aggregates of speed and mileage information by considering separately individual daytime and nighttime driving statistics. This increased detail of aggregation was shown to improve performance. In [10] the idea above is further developed by designing a data-driven model for predicting car drivers' risk of crash based on a model called individual mobility network. In [25], such a work is extended with geographical transfer learning.

While extremely useful in creating a general risk profile of a customer, long-term crash prediction approaches cannot be exploited to solve real-time crash prediction that, on the other hand, can be very important for car insurance companies to support the driver with first help and send immediate assistance. Indeed, a large part of the literature focuses on real-time crash prediction. In [29] a model is presented for real-time collision detection at road intersections by mining collision patterns. In [32], the idea is to provide feedback to the user while driving by identifying the events that will cause a crash in the next few seconds. These approaches relate to sensors and mobility data, while [4] tries to link crashes to both physiological parameters and behavioral characteristics. Another branch of the literature [18,1,24] focuses on identifying mobility areas, i.e., crossroads, intersections, parts of highways, that show characteristics usually associated with accidents, such as adverse weather conditions, increased traffic density, etc. To improve the performance of the proposed probabilistic model, the work in [15] uses individual speed and time headway of cars passing through predefined detector stations, besides peculiarities describing mobility areas. In [21] it is presented a survey analyzing the key problems associated with crash-frequency data and the strengths and weaknesses of some methodological approaches. In [16], is presented a simple car crash detection algorithm implemented on Android smartphones using accelerometer sensor and location sensor information to detect typical

patterns of car crash situations. Recently, in [27] is proposed a framework for automated accident detection based on multimodal sensors in cars.

We place our analysis more closely related to the second family of approaches, i.e., real-time crash prediction. However, while in all the aforementioned works, the classifiers are typically applied *before* the crash happening, in our scenario, the automatic decision system that suggests the presence of a crash is applied *after* that the crash happened. Also, a limitation of all the complex models used in the aforementioned approaches is that they are not interpretable, i.e., the supervised classifiers are black-box models [9]. Our goal is not to build such a model but to design an explanation workflow that allows a better understanding of why our pre-trained classification model signals that a crash happened.

3 Problem Description and Explanation Methodologies

Generali collects high-dimensional time series data through Crash Data Recorders (CDR). This data is transferred in real-time from the CDR to the insurance company servers, and it is processed through an automatic decision-making system. If the AI system signals the presence of a possible crash, the operator calls the customers to check if there is a need for first assistance. Generali aims at solving two criticalities in the current approach. First, *reduce false positives* by increasing the model precision in order to avoid unnecessary calls. Second, *increase the interpretability of the automatic decision-making system* to help the operator choose the right course of action.

We keep this paper self-contained, presenting all the necessary concepts to understand our proposal. Formally, we define a multivariate time series as follows:

Definition 1 (Multivariate Time Series)**.** A multivariate *time series* $X = \{x_{1,1}, \dots x_{j,k}, \dots, x_{m,d}\} \in \mathbb{R}^{m \times d}$ is an ordered set of m real-valued observations, each having $d > 1$ dimensions (or signals/channels).

A Time Series Classification (TSC) dataset is a set of time series with a vector of labels attached. Formally:

Definition 2 (TSC Dataset)**.** A *time series classification dataset* $\mathcal{D} = (\mathcal{X}, \mathbf{y})$ is a set of n time series, $\mathcal{X} = \{x_{1,1,1}, \dots x_{i,j,k}, \dots, x_{n,m,d}\} \in \mathbb{R}^{n \times m \times d}$, with a vector of assigned labels (or classes), $\mathbf{y} = \{y_1, y_2, \dots, y_n\} \in \{0, 1\}^n$.

In practice, observation (i, j, k) of a time series dataset is denoted by $x_{i,j,k}$. The index i denotes the i^{th} multivariate time series in the dataset, j denotes the j^{th} time-step, k denotes the k^{th} signal of the time series. Hence, we define Time Series Classification (TSC) as:

Definition 3 (TSC)**.** Given a TSC dataset $\mathcal{D}$, *Time Series Classification* is the task of training a function f from the space of possible inputs $\mathcal{X}$ to a probability distribution over the class variable values in $\mathbf{y}$.

In the rest of this section, we describe in detail the dataset provided by Generali, and the deep learning model adopted, referring to the definitions above.

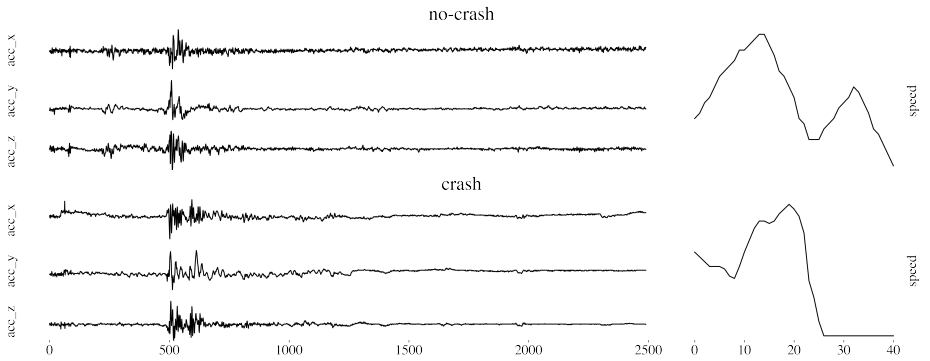

Fig. 1. Multivariate time series samples: *top* - no-crash instance, *bottom* - crash instance, *left* - acceleration times series, *right* - speed time series.

Dataset. The car crash dataset ($\mathcal{D}$) provided by Generali contains $n = 81,173$ instances. Each instance is a multivariate time series composed of four signals ($d = 4$), namely the *acceleration* of the car for the x, y, z axes, and the *speed* of the car. The acceleration signals of the car contain $m_1 = 2,490$ observations for each axis, and they are, in turn, a concatenation of two signals sampled at different frequencies. The speed signal is a recording containing $m_2 = 41$ time-steps. Each multivariate time series X_i is labelled either as $y_i = 1$ when it is a crash or as $y_i = 0$ when it is a no-crash. Crashes are rarer than no-crashes and represent only about the 6% of the dataset. From a classification perspective, the main critical issues in dealing with this dataset are the presence of signals with a big difference in length, i.e., speed and acceleration, and the heavy label unbalance. An example of two instances is depicted in Figure 1. The dataset is split by Generali into 50% training set, 25% validation set, and 25% test set.

Predictive Model. The automatic decision-making system adopted by Generali implements the predictive model with a Convolutional Neural Network (CNN) [17] with a deep structure[5] that returns a probability for a crash between 0 and 1. On the test set, the CNN has an accuracy of 0.961 and a precision of 0.701. As already mentioned, precision is paramount in this setting because every false positive, i.e., every no-crash classified as crash, causes unnecessary calls by human operators. Also, due to the CNN architecture, the predictive model is a black-box, i.e., given a prediction as crash or no-crash, for a human, it is not possible to understand the reasons that lead to that decision [9]. Thus, the challenges raised by Generali are *(i)* gaining insights regarding the logic behind the prediction of the provided CNN, without making any modification to the data or the model, to understand which parts of the data and which patterns of the multivariate time series are more important for the classification, *(ii)* building an efficient and effective, interpretable alternative for the provided CNN model, possibly reducing

[5] The specific structure and training details can not be disclosed due to company policies.

false positives and optimizing data usage. In particular, Generali requires high efficiency at test time, to minimize response time when dealing with new and unseen crash or no-crash instances.

Explanation Methodologies and Workflow. To meet these challenges, we propose a set of explanation methodologies organized as a workflow and divided into the following steps.

1. **Analyze the attention** of the provided CNN, inspecting its gradients, discovering the parts of the multivariate time series that are more relevant for the prediction (Section 4);
2. **Build interpretable subsequence-based surrogates** of the CNN that rely on subsequences to imitate its output, focusing on the parts highlighted by the previous step (Section 5);
3. **Train an interpretable subsequence-based model** directly on the real labels, as a possible replacement of the CNN, still focusing only on the parts highlighted in the first step while also leveraging the results of the second step (Section 6).

In the following, we present the background necessary to comprehend each step, as well as the experimental results and main findings.

4 Gradient-based Explainer

In order to inspect the CNN and to gain knowledge about its attention, we use a gradient-based interpretability approach called GradientExplainer, which is an extension of the Integrated Gradients method proposed in [31]. It is a feature attribution method that returns an explanation in the form of a saliency map, highlighting the contribution of each time-step for the classification [9]. Formally:

Definition 4 (Saliency Map)**.** Given a multivariate time series X a *saliency map* $\Phi = \{\phi_{j,k} \mid \forall j \in [1, m], k \in [1, d]\}$ contains a score $\phi_{j,k}$ for every real-valued observation $x_{j,k}$ of X.

In particular, this saliency map is returned in terms of approximated SHAP values [22], obtained by computing the expectations of gradients, sampling reference values from a background dataset. In practice, to compute the SHAP values, X is perturbed using a matrix $Z' \in \{0, 1\}^{m \times d}$ to decide which values to keep and which values to replace in X. Additive feature attribution methods assume the explanation to be linear; therefore, each input observation receives a positive or negative SHAP value, depending on its contribution to the model output. Formally, for multivariate time series:

Definition 5 (Additive Feature Attribution)**.** An additive feature attribution method g has an explanation model that is a linear function of binary variables, $g(Z') = \phi_0 + \sum_{j=1}^{m} \sum_{k=1}^{d} \phi_{j,k} z'_{j,k}$, where $z'_{j,k} \in \{0, 1\}$ and $\phi_{j,k} \in \mathbb{R}$.

In other words, given a time series X, the explanation model g tries to transparently approximate the prediction of a black-box classifier f in the local

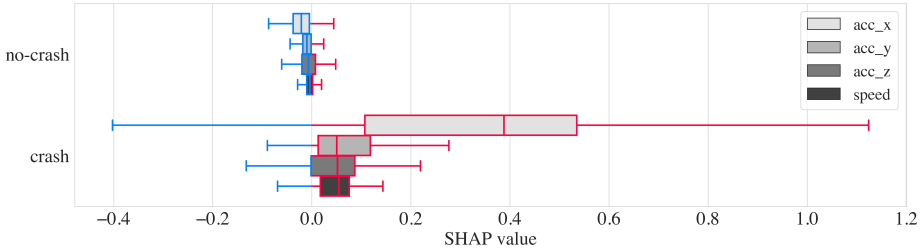

Fig. 2. Boxplots of SHAP values aggregated by signal. Red and blue colors highlight positive and negative SHAP values.

neighborhood of X, i.e., $g(Z') \approx f(X)$[6]. In our binary classification setting, for each time-step j and each signal k, a $\phi_{j,k}$ SHAP value close to 0 indicates that the point $x_{j,k}$ is almost irrelevant for the classification; a positive value indicates a contribution towards the class crash, while a negative value indicates a contribution towards the class no-crash. ϕ_0 is the base value, i.e. the default classification output for an "empty" time series. This kind of explanation is *local*, meaning that it can be used to shed light on the decision of the CNN for single time series. However, by retrieving such explanations for each instance of the dataset and aggregating the SHAP values at different granularity, we can gain an overview of the global behavior of the model. First, we aggregate SHAP values signal-wise to understand which are the most relevant dimensions for the classification. Then, they are aggregated observation-wise to recognize the time-steps of the time series that have a greater impact. Despite being simple operations, we highlight that, to the best of our knowledge, these two steps are a novel contribution in the panorama of multivariate time series explanation.

Signal Importance. The first aggregation is performed signal-wise for each multivariate time series by summing the SHAP values of each signal. This way, for every prediction of the CNN model, we can understand the impact of the different dimensions for every instance in the dataset. These sums are collected in a matrix $\Phi^{\text{signal}} \in \mathbb{R}^{n \times d}$ such that $\Phi^{\text{signal}}_{i,k}$ is defined as: $\Phi^{\text{signal}}_{i,k} = \sum_{j=1}^{m} \phi_{i,j,k}$ with $i \in [1, n],\ k \in [1, d]$. In our setting, Φ^{signal} is a matrix with n rows and $d = 4$ columns, corresponding to the sums of the SHAP values of all time-steps for each time series in $\mathcal{X}$ for each signal, i.e., the accelerations on the *x, y, z axes* and the *speed*. These sums are visualized in the boxplot in Figure 2 grouped w.r.t. the signal type and divided by class. The difference in width of the boxplot depends upon the base value ϕ_0. In our setting, ϕ_0 is closer to the class no-crash, probably due to the heavy label unbalance. From this analysis, it is quite clear that *the most relevant dimension is the acceleration on the x-axis*, for which the SHAP values deviate the most from 0. This suggests a higher degree of contribution towards the classification, both for a crash and no-crash time series.

[6] Please refer to [22] for more details.

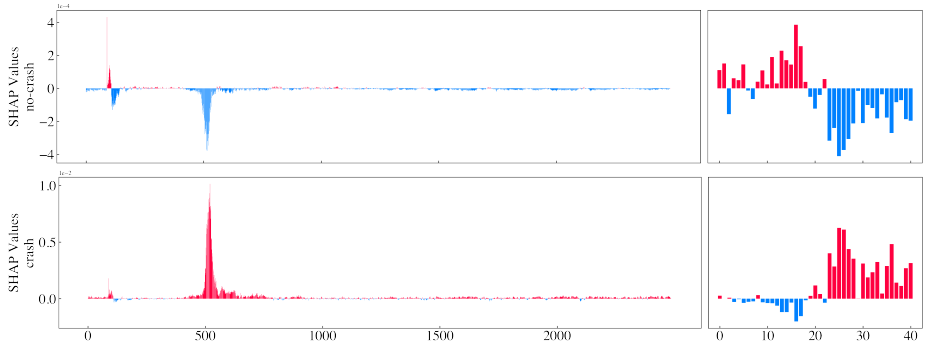

Fig. 3. Mean of SHAP values aggregated pointwise. *left*: accelerations; *right*: speed. Red and blue colors highlight positive and negative SHAP values.

The acceleration on the y and z-axis and the speed seem to be less impactful. However, their contributions are not irrelevant, even if smaller w.r.t. the *x-axis*.

Time-Step Importance. A more fine-grained insight can be obtained by averaging the SHAP values by time-step. Similarly to the previous point, we collected these averages in a matrix $\Phi^{\text{point}} \in \mathbb{R}^{m \times d}$ such that $\Phi^{\text{point}}_{j,k}$ is defined as: $\Phi^{\text{point}}_{j,k} = \frac{1}{n} \sum_{i=1}^{n} \phi_{i,j,k}$ with $j \in [1, m]$, $k \in [1, d]$. In this case, for each time-step of each signal, we can see its average contribution. In the barplot in Figure 3, we present such averages, stacking the accelerations on the left-hand side plots for better readability. We notice that, *the main area of attention for the acceleration signals is around the* 500^{th} *time-step, independently of the class.* Moreover, regarding no-crash instances, there is also a minor contribution around the 100^{th} time step, with a very high peak of positive SHAP values, corresponding to the point of the concatenation of signals having different frequencies. This concatenation point seems to nudge the CNN towards the class crash, even in no-crash instances. This concentration of high SHAP values highlights a possible defect of the black-box model provided by Generali, which regards the concatenation of signals sampled at different frequencies as an important feature for the classification. The second part of the time series seems more relevant for the speed signal, especially for the class crash. *These insights gained at the time-step granularity are extremely important because they suggest that only a very small part of the data is relevant for the classification.* Indeed, in the following sections, we use this information to optimize data usage, while we will continue to consider the four dimensions.

5 Subsequence-based Surrogates

Subsequences are one of the most common ways to build interpretable models in the time series domain. Formally:

Definition 6 (Subsequence)**.** Given a single signal $\mathbf{x} = \{x_1, \ldots, x_m\}$ of the multivariate time series X, a *subsequence* $\mathbf{s} = \{x_j, \ldots, x_{j+l-1}\}$ of length l is an ordered sequence of values such that $1 \leq j \leq m - l + 1$.

Subsequences can be real-valued, like shapelets [34], or can be symbolic, i.e. discretized, like SAX-based subsequences [19]. In general, subsequence-based classification approaches search for patterns that better discriminate the dataset labels, i.e., they try to find those subsequences that are most dissimilar between instances belonging to different classes. Subsequence extraction is computationally expensive, and therefore, using the insights previously gained, *we trim the x, y, z acceleration signals taking only the observations between the* 400^{th} *and* 800^{th} *time-step*. In this way, the issue earlier raised regarding the concatenation of the two signals sampled at different frequencies, is also avoided. On the other hand, we leave the speed signal as it is. We denote this filtered version of $\mathcal{X}$ with $\mathcal{X}'$.

Shapelet-based Subsequences. We perform shapelet extraction by using Learning-Shapelets (LTS) [8], an approach that learns shapelets via stochastic gradient descent without the need to explore all the possible candidates. Once the most discriminative shapelets are found, the Shapelet Transform [20] is applied in order to transform the time series dataset into a simplified representation.

Definition 7 (Shapelet Transform)**.** Given a time series dataset $\mathcal{X}$ and a set S containing h shapelets, the *Shapelet Transform*, σ, converts $\mathcal{X} \in \mathbb{R}^{n \times m \times d}$ into a real-valued matrix $T \in \mathbb{R}^{n \times h}$, obtained by taking the minimum Euclidean distance between each time series in $\mathcal{X}$, and each shapelet in S, via a sliding-window.

As a note, the sliding-window Euclidean distance is calculated w.r.t. the signal/channel the shapelet was extracted from.

SAX-based Subsequences. The Symbolic Aggregate approXimation (SAX) algorithm [19] transforms time series into strings. SAX uses the Piecewise Aggregate Approximation (PAA) [14] to divide the time series into equally sized intervals and averages the values of each interval. Then, the time series is discretized using a sequence of symbols. For our application, the subsequences extraction is performed by using MR-SEQL [26], an approach that combines SAX with different-sized sliding windows, thus converting time series into a symbolic form with multiple resolutions. The most discriminative subsequences are searched with SEQL [12], an algorithm that filters the symbolic patterns using a greedy feature selection strategy. Similar to the Shapelet Transform, the dataset is transformed into a new representation, having as features the extracted subsequences and as values 0 or 1 depending on the absence or presence of subsequences inside the time series. Formally:

Definition 8 (Symbolic Subsequence Transform)**.** Given a dataset $\mathcal{X}$ and a set S containing h symbolic subsequences, the *Symbolic Subsequence Transform*, σ, converts $\mathcal{X} \in \mathbb{R}^{n \times m \times d}$ into a binary-valued matrix $T \in \{0, 1\}^{n \times h}$, obtained by checking if each subsequence in S is contained or not in each time series in $\mathcal{X}$.

For interpretability purposes, the symbolic subsequences can be easily mapped back to the original segments of the time series.

Table 1. Surrogates performance *(higher is better, best values in bold).*

	SAX DT	SAX RF	SAX XGB	SAX LGB	SAX CAT	SHP DT	SHP RF	SHP XGB	SHP LGB	SHP CAT
accuracy	0.954	0.966	0.975	0.974	**0.976**	0.950	0.950	0.951	0.951	0.951
precision	0.621	**0.912**	0.862	0.855	0.862	0.639	0.641	0.624	0.623	0.623
recall	0.567	0.475	0.681	0.678	**0.697**	0.360	0.361	0.435	0.438	0.437
fscore	0.593	0.625	0.761	0.756	**0.771**	0.460	0.462	0.513	0.514	0.514

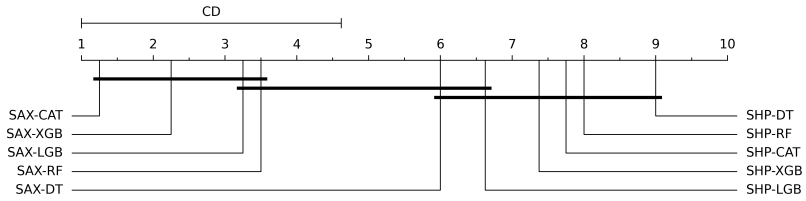

Fig. 4. Critical Difference diagram for the surrogate models. The models are sorted from the best (*left*) to the worst (*right*). Models connected with a line have performance that are not statistically different using a Nemenyi test [11] with a p-value of 0.05.

Tree-based Global Surrogates. Regardless of the way subsequences are extracted, the new simplified tabular representation $T' = \sigma(\mathcal{X}')$ can be paired with any classification model, having the advantage of a more interpretable input [20]. Practically, most of the implementations of subsequence extraction methods work only on univariate time series or multivariate time series having signals with the same length. Since we have different-sized dimensions, we independently extract the subsequences from each signal of the multivariate time series[7], and we concatenate the resulting transformed datasets column-wise.

As classification models, we adopt tree-based classifiers because they simultaneously offer good performance and provide partial interpretability by granting the possibility to access the feature importance. We train the following five models[8]: a standard Decision Tree (DT) as baseline, Random Forest (RF) [5], XGBoost (XGB) [6], LightGBM (LGB) [13] and CatBoost (CAT) [28]. We highlight that *we train these classifiers as global surrogates*, i.e., not on the original dataset labels $\mathbf{y}$, but on the prediction of the CNN, $\mathbf{y}_{\text{CNN}} = \text{CNN}(\mathcal{X})$, in order to imitate its output in a more interpretable way. In other words, the purpose of these surrogate models is to emulate the underlying global logic of the CNN as well as possible [9]. To guarantee a high level of generalization, the surrogate models are trained on the prediction of the CNN for the validation set.

[7] The subsequence extraction is performed using the default implementation parameters for `MR-SEQL` and the heuristic proposed in [8] for `LTS`.

[8] All the models are trained using the default library implementation parameters: `Scikit-learn` for DT, RF, `XGBoost` for XGB, `LightGBM` for LGB, `CatBoost` for CAT.

Table 2. Models performance.

	accuracy	*precision*	*recall*	*fscore*	*roc-auc*	*runtime*
CNN	0.961	0.701	0.660	0.680	0.924	784 ± 69.5
SAX-CAT	0.958	0.760	0.471	0.582	0.911	218 ± 23.9

We measure the performance on the test set in terms of accuracy, precision, recall and f-score, and we report them in Table 1. All the metrics are computed w.r.t. the prediction of the CNN, i.e. given a model M: $metric = eval(\mathbf{y}_{\text{CNN}}, \mathbf{y}_{\text{M}})$ with $\mathbf{y}_{\text{M}} = \text{M}(\sigma(\mathcal{X}'))$. Also, we summarize the results in the Critical Difference (CD) plot in Figure 4. In general, all SAX-based methods outperform their shapelet-based counterpart. They are quite successful in imitating the output of the CNN, using a fraction of the input data. Specifically, SAX-LGB, SAX-XGB, SAX-CAT and SAX-RF perform better than SAX-DT. SAX-RF achieves the highest precision; however, it falls behind in all other performance metrics. From these results, it is quite clear that the best performing model overall is SAX-CAT. Given that SAX-CAT imitates the provided model with high accuracy, the next logical step is to train it on the real dataset labels as a completely independent model, bypassing the need of querying the CNN.

6 Subsequence-based Classifier

The best performing model of Section 5 (SAX-CAT) is chosen as a possible interpretable replacement of the CNN. As for the surrogates, SAX-CAT is trained on the transformed version of the trimmed set, $T' = \sigma(\mathcal{X}')$, which only contains around 17% of the initial observations. However, differently from the previous section, SAX-CAT is trained on the original training set labels $\mathbf{y}$. The optimal number of iterations of CatBoost is set to 295 by monitoring the Logloss on the validation set. We keep the other parameters as default values.

In Table 2 we benchmark the performance of SAX-CAT and the CNN on the test set, comparing $\mathbf{y}_{\text{CNN}}$ and $\mathbf{y}_{\text{SAX-CAT}}$ with the original test labels $\mathbf{y}$. This result shows a comparable performance in terms of accuracy, with a substantial improvement in precision. On the other side, there is a degradation in performance both for the recall and f-score. However, as required by Generali, the main purpose of our work was to reduce the number of false positives, giving less weight to false negatives. In the next stage of our collaboration, we will try to maintain the precision we reached in this work by increasing the recall. On training time, the most expensive computation for SAX-CAT is the extraction of discriminative subsequences, which takes about 7 hours. However, this search has to be performed only once. On the other hand, the training of the CNN takes less than an hour. On test time, the average runtimes to classify and explain an unseen test instance are $784\,\text{ms} \pm 69.5\,\text{ms}$ for the CNN and $218\,\text{ms} \pm 23.9\,\text{ms}$ for SAX-CAT. The runtime of SAX-CAT includes the subsequence transform, the classification and explanation in terms of features importance, which is attached to the SAX-based features.

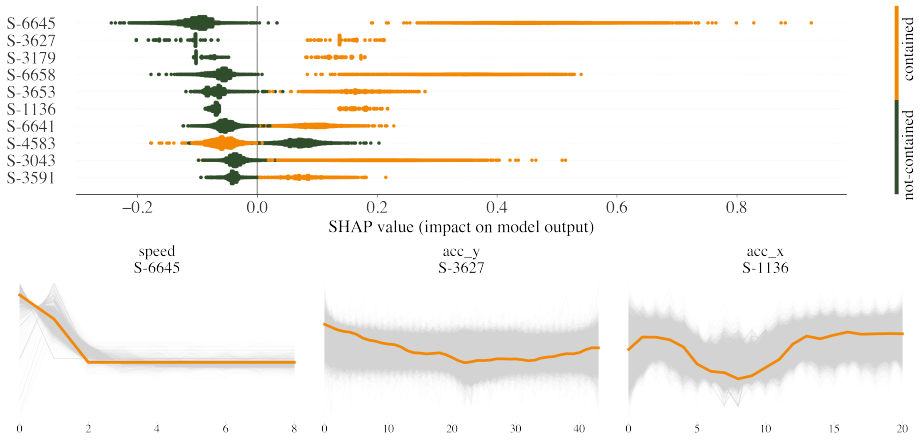

Fig. 5. SHAP values summary plot (*top*); sample of relevant subsequences (*bottom*).

The most significant advantage of using tree-based approaches is that their predictions can be efficiently and effectively interpreted using SHAP's TreeExplainer [23], both from a local and global perspective. Formally, the formula in Definition 5 can be rewritten as: $g(\mathbf{z}') = \phi_0 + \sum_{j=1}^{h} \phi_j z'_j$ where h is the number of extracted subsequences. The extracted contributions are guaranteed to be consistent and locally accurate. Moreover, these local explanations can be aggregated to have a general global overview of the logic behind the model. Practically, in our setting, we can understand which subsequences are more relevant for classifying crash and no-crash instances.

The global explanation plot is presented in Figure 5 (top). This summary global plot depicts the SHAP values for all the instances and subsequences in the dataset and sorts them by the overall impact on the model prediction. Specifically, on the y-axis are presented the top-10 subsequences, sorted from the most influential (top) to the least influential (bottom). For example, the 3591^{th} subsequence (*S-3591*) is the least important subsequence in the top-10. Each point on each row in the plot corresponds to one multivariate time series. Points colored in orange indicate that the corresponding subsequence is contained in the multivariate time series, while points colored in green represent a not-contained subsequence. For example, an orange point in the *S-3591* line means that the time series corresponding to that point contains the subsequence with index 3591. The SHAP values are plotted on the x-axis, showing the contribution of feature values toward the classification. For example, an orange point with a SHAP value greater than 0 in the *S-3591* line means that for that multivariate time series, the presence of the subsequence *S-3591* influenced the model in predicting the class crash. In general, this plot helps visualize if the presence/absence of subsequences contributes to the no-crash (negative SHAP values) or crash (positive SHAP values) class at a global level. In our case, the most influential subsequence is *S-6645*, belonging to the *speed signal*, which represents a decrement of car speed

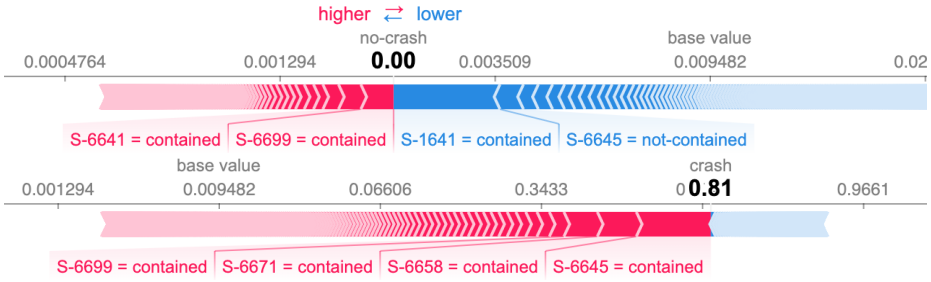

Fig. 6. SHAP values explanation for the two instances in Figure 1 obtained from SAX-CAT. *top*: no-crash instance; *bottom*: crash instance.

followed by a stop. As expected, the presence of *S-6645* contributes to the class crash, while its absence is an indicator of the class no-crash.

Three example subsequences are presented in Figure 5 (bottom). Given that they are symbolic subsequences, they can assume slightly different shapes in the dataset; therefore, a representative subsequence is presented in orange, while the other subsequences are presented in a faded gray. The representative is computed as a medoid, i.e., the subsequence that minimizes the average Euclidean distance w.r.t. the other subsequences. Given their higher frequency, subsequences belonging to the acceleration axes are inherently harder to interpret. However, they could still be of interest to domain experts. In fact, the importance of these subsequences for the model output suggests that the car's jerk (or jolt), i.e., the rate at which the car's acceleration changes w.r.t. time, is probably relevant to the classification. For example, the most important subsequence for the *y-axis* is *S-3627*, and presents a decrement in acceleration, contributing to the class crash. *S-1136* is instead most relevant for the *x-axis* and presents a decrement followed by an increment in acceleration, also contributing to the class crash.

Besides having a global explanation, we can also understand the classification at a local level, i.e., for each individual time series. In Figure 6 we report the explanations for the two instances depicted in Figure 1. Features contributing to the class crash are colored in red, while features contributing to the class no-crash are colored in blue. These explanation plots, combined with the knowledge of the shape of the subsequences, can help domain experts to understand the logic behind the prediction of the tree-based approach. For example, the top plot represents the subsequence contribution towards the classification of the first no-crash instance in Figure 1. In this case, the presence of the 6641^{th} and the 6699^{th} subsequence pushes the prediction toward the class crash, while the presence of the 1641^{th} subsequence and the absence of the 6645^{th} contribute towards the class no-crash. Given that the instance is a no-crash, contributions towards no-crash are greater in magnitude w.r.t. contributions towards the class crash. Moreover, many other minor contributions are too small to be shown in the plot but can still be accessed by Generali operators. The prediction for the crash instance is similarly explained in the bottom part of the same figure.

7 Conclusion

We have presented a XAI workflow to tackle the problem of explainability in the domain of crash prediction. First, through a gradient-based approach, we have gained insights into the attention of the CNN adopted by Generali, understanding which part of the data is more relevant to its predicted output. Using this knowledge, we have optimized the data usage, and we have built and compared subsequence-based surrogates which imitate the prediction of the CNN. Finally, we have trained the best surrogate model, i.e., SAX-CAT, on the original dataset labels as an interpretable replacement of the CNN. We have observed that, with respect to the CNN, the predictive performance of SAX-CAT is slightly subpar in terms of recall and f-score, comparable in terms of accuracy, and achieves higher precision, thus reducing false positives which was one of the goals of this work. In addition, the observed results are also extremely promising in terms of runtime and efficiency at test time: SAX-CAT is three times faster than the CNN in making the prediction and uses only 17% of the original data. Also, the predictions of SAX-CAT are also interpretable both from a local and global standpoint, providing an explainability layer that allows understanding the logic behind the model in terms of subsequences. As future research directions, we would like to increase the performance by further fine-tuning the subsequence-based models. Another perspective is to focus on data by carefully relaxing the constraint of efficiency in terms of data utilization and using a larger part of the dataset for the training, or by representing the time series in other feature spaces. Furthermore, we plan on improving the interpretability of the framework. Indeed, the current explanation is in the form of feature importance, which can sometimes be challenging to understand as the number of features grows. In this sense, we are investigating different explanation types, such as prototypes, counterfactuals and decision rules, to further simplify the explanation, possibly injecting the experts' domain knowledge. Finally, the flexibility of the proposed XAI workflow allows for its deployment in other fields that heavily rely on multivariate time series data.

Acknowledgment. This work has been partially supported by the European Community Horizon 2020 programme under the funding schemes: G.A. 871042 *SoBigData++*, G.A. 952026 *HumanE AI Net*, and G.A. 834756 *XAI*.

References

1. Abdel-Aty, M.A., et al.: Calibrating a real-time traffic crash-prediction model using archived weather and its traffic data. IEEE Trans. Intell. Transp. Syst. **7** (2006)
2. Assaf, R., et al.: Explainable deep neural networks for multivariate time series predictions. In: IJCAI. pp. 6488–6490 (2019)
3. Assaf, R., et al.: Mtex-cnn: Multivariate time series explanations for predictions with convolutional neural networks. In: ICDM. pp. 952–957. IEEE (2019)
4. Ba, Y., et al.: Crash prediction with behavioral and physiological features for advanced vehicle collision avoidance system. TR_C **74**, 22–33 (2017)
5. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
6. Chen, T., et al.: Xgboost: A scalable tree boosting system. pp. 785–794 (08 2016)

7. European Commission: Proposal for the artificial intelligence act. (2021)
8. Grabocka, J., et al.: Learning time-series shapelets. In: KDD. ACM (2014)
9. Guidotti, R., et al.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5) (2019)
10. Guidotti, R., et al.: Crash prediction and risk assessment with individual mobility networks. In: MDM. IEEE (2020)
11. Hollander, M., et al.: Nonparametric statistical methods, vol. 751. JW&S (2013)
12. Ifrim, G., et al.: Bounded coordinate-descent for biological sequence classification in high dimensional predictor space. In: KDD. pp. 708–716 (2011)
13. Ke, G., et al.: Lightgbm: A highly efficient gradient boosting decision tree. In: Guyon, I., et al. (eds.) NIPS. vol. 30. Curran Associates, Inc. (2017)
14. Keogh, E., et al.: Scaling up dynamic time warping for datamining applications. In: KDD. p. 285–289. KDD '00, ACM, New York, NY, USA (2000)
15. Kweon, Y.J., et al.: Development of crash prediction models with individual vehicular data. TR_C **19**(6) (2011)
16. Lahn, J., et al.: Car crash detection on smartphones. In: iWOAR. ACM (2015)
17. LeCun, Y., et al.: Gradient-based learning applied to doc. recognition. IEEE (1998)
18. Lee, C., et al.: Real-time crash prediction model for application to crash prevention in freeway traffic. TRR **1840**(1), 67–77 (2003)
19. Lin, J., et al.: Experiencing SAX: A novel symbolic representation of time series. Data Min. Knowl. Discov. **15**, 107–144 (08 2007)
20. Lines, J., et al.: A Shapelet Transform for Time Series Classification. In: KDD. p. 289–297. KDD '12, ACM, New York, NY, USA (2012)
21. Lord, D., et al.: The statistical analysis of crash-frequency data: a review and assessment of methodological alternatives. TR_A **44**(5) (2010)
22. Lundberg, S.M., et al.: A unified approach to interpreting model predictions. In: NIPS. pp. 4768–4777 (2017)
23. Lundberg, S.M., et al.: From local explanations to global understanding with explainable ai for trees. Nat. Mach. Intell. **2**(1), 56–67 (2020)
24. Mannering, F.L., et al.: Analytic methods in accident research: Methodological frontier and future directions. Analytic methods in accident research **1**, 1–22 (2014)
25. Nanni, M., et al.: City indicators for geographical transfer learning: an application to crash prediction. GeoInformatica pp. 1–32 (2022)
26. Nguyen, T.L., et al.: Interpretable time series classification using linear models and multi-resolution symbolic representations. DAMI **33**(4), 1183–1222 (2019)
27. Pour, H.H., et al.: A machine learning framework for automated accident detection based on multimodal sensors in cars. Sensors **22**(10), 3634 (2022)
28. Prokhorenkova, L., et al.: Catboost: Unbiased boosting with categorical features. In: NeurIPS. p. 6639–6649. NIPS'18, Curran Associates Inc. (2018)
29. Salim, F.D., et al.: Collision pattern modeling and real-time collision detection at road intersections. In: ITSC. pp. 161–166. IEEE (2007)
30. Selvaraju, R.R., et al.: Grad-CAM: Visual explanations from deep networks via gradient-based localization. In: ICCV. pp. 618–626 (2017)
31. Sundararajan, M., et al.: Axiomatic attribution for deep networks. In: ICML. Proceedings of Machine Learning Research, vol. 70, pp. 3319–3328. PMLR (2017)
32. Wang, J., et al.: Real-time driving danger level prediction (2010)
33. Wang, Y., et al.: Ml methods for driving risk. In: EM-GIS. ACM (2017)
34. Ye, L., et al.: Time series shapelets: a new primitive for data mining (06 2009)
35. Zantalis, F., et al.: A review of machine learning and IoT in smart transportation. Future Internet **11**(4), 94 (2019)
36. Ziebinski, A., et al.: Review of advanced driver assistance systems (ADAS) (2017)