TÉCNICO LISBOA

Explainable Artificial Intelligence for Multivariate Time Series Regression Models

Pedro André Gonçalves Zenário

Thesis to obtain the Master of Science Degree in

Data Science and Engineering

Supervisors: Dr. Ruxandra Georgeta Barbulescu
Prof. Luís Miguel Teixeira d'Avila Pinto da Silveira

Examination Committee

Chairperson: Prof. Bruno Emanuel da Graça Martins
Supervisor: Dr. Ruxandra Georgeta Barbulescu
Member of the Committee: Prof. Alexandra Sofia Martins de Carvalho

November 2023

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First of all, I would like to thank my supervisors Professor Luís Miguel Silveira and Doctor Ruxandra Barbulescu for all their guidance throughout the development of this Thesis. Their availability to meet every week was incredibly helpful, their understanding of my absense from this work to devote myself to another course was also admirable and supportive, and for every advice and review of my work provided I am very grateful.

I would like to thank my family and Leonor, who has always been a constant support throughout my academic journey and specially for their support during the most difficult moments in the realization of this work.

Finally, I need to thank all of my fellow friends and colleagues that worked together with me, while working on their own Master's Thesis, which was very comforting.

Abstract

The rapid advancement of Artificial Intelligence (AI) has revolutionized various industries by offering innovative solutions to intricate problems. However, this progress often results in complex, opaque models, making it challenging to comprehend their decision-making processes. This complexity is particularly concerning in vital sectors like medicine and autonomous driving, where trust and transparency are paramount. To address this challenge, eXplainable AI (XAI) has emerged as a pivotal research area. Understanding the underlying patterns and behaviors in data, especially within complex and dynamic systems, is essential for informed decision-making.

This Thesis delves into XAI and Machine Learning (ML) interpretability, providing an extensive literature review that primarily focuses on elucidating the decision-making process of Deep Learning (DL) models in multivariate time-series (MTS) regression. The study centers on the analysis of the responses of a roundworm's nervous system (Caenorhabditis Elegans) under diverse stimuli. A Gated Recurrent Unit (GRU) model was modified for this purpose, with adjustments in the last layer to align with the chosen explanation methods. The core of this research lies in a post-hoc explanation technique called SHAP, which not only identifies the features influencing the DL model's decisions but also elucidates how features at different time points affect these decisions. The application of SHAP was validated through two experiments, involving limited input data during inference. These experiments corroborated the robustness and accuracy of SHAP explanations, demonstrating its potential to enhance the interpretability of DL models in MTS regression tasks. This study emphasizes the vital role of explainability in AI systems, enhancing trust and confidence in the decisions made by these complex algorithms.

Keywords

Explainable Artificial Intelligence, Machine Learning Interpretability, Regression, SHAP, Time Series

Resumo

O rápido avanço da Inteligência Artificial (AI) revolucionou vários sectores oferecendo soluções inovadoras para problemas complexos, mas à custa de modelos complexos e opacos, tornando difícil a compreensão dos processos de tomada de decisão. Esta complexidade é preocupante em sectores vitais como a medicina, onde a confiança e a transparência são fundamentais. AI Explicável (XAI) surgiu então como uma área de investigação fundamental. Compreender os padrões e comportamentos subjacentes em dados, especialmente em sistemas complexos e dinâmicos, é essencial para uma tomada de decisão informada. Esta tese investiga XAI e a interpretabilidade em Aprendizagem Automática, fornecendo uma revisão da literatura focando-se em elucidar o processo de tomada de decisão dos modelos de Aprendizagem Profunda na regressão de séries temporais multivariadas (MTS). O estudo centra-se na análise das respostas do sistema nervoso de uma lombriga sob diversos estímulos, usando um modelo Gated Recurrent Unit (GRU). O cerne desta investigação reside numa técnica de explicação pós-treino denominada SHAP, que não só identifica as características que influenciam as decisões do modelo, mas também elucida como as características em diferentes momentos afetam essas decisões. A aplicação de SHAP foi validada através de duas experiências, envolvendo dados de entrada limitados durante a inferência. Estas experiências corroboraram a robustez e a precisão das explicações de SHAP, demonstrando potencial para melhorar a interpretabilidade dos modelos em tarefas de regressão MTS. Este estudo realça o papel vital da explicabilidade nos sistemas AI, aumentando a confiança nas decisões tomadas por estes algoritmos complexos.

Palavras Chave

Inteligência Artificial Explicável, Interpretabilidade em Aprendizagem Automática, Regressão, Séries Temporais, SHAP

Contents

Acronyms

AI	Artificial Intelligence
DL	Deep Learning
FCM	Forward Crawling Motion
GRU	Gated Recurrent Unit
ML	Machine Learning
LIME	Local Interpretable Model-agnostic Explanations
RUL	Remaining Useful Life
SHAP	SHapley Additive exPlanations
TS	Time Series
MTS	Multivariate Time Series
XAI	eXplainable Artificial Intelligence

List of Figures

1

Introduction

Contents

1.1 Context and Motivation

Artificial Intelligence (AI) is fast becoming embedded in industries, economies and lives, making decisions, recommendations and predictions [7]. AI algorithms can be trained to perform a wide array of tasks, but these systems often are opaque and operate in a black box, leaving users in the dark about their decision-making processes. These systems, powered by AI and frequently employing Deep Learning (DL) techniques, exhibit remarkable predictive capabilities across diverse subjects. Yet, the value of these predictions is cast into doubt when the rationale behind them remains elusive [8]. The process has to be transparent, trustworthy and compliant - far removed from the opaque black box concept that has characterized some AI advances in recent times [7]. Consequently, it is imperative for AI algorithms to possess explanatory features, enabling users to comprehend the basis for specific decisions. This need has given rise to the concept of eXplainable Artificial Intelligence (XAI) [8].

XAI refers to the ability of algorithms to elucidate their reasoning and outline the strengths and weaknesses inherent in their decision-making procedures, that allow humans to trust and have confidence in Machine Learning (ML) algorithms. AI is now used to find predictive patterns within expanding enterprise data repositories. Companies leverage these insights to enhance existing decisions and gradually refine business outcomes. By deploying predictive models in a manner that allows for monitoring and management of decision outcomes, accuracy can be ensured. Decision-makers extract insights from these models, reinforcing their confidence in the noticeable improvements they provide [8]. By applying AI in this way, businesses can mitigate the black box element, helping to build trust and confidence among consumers [7].

While this advancement represents a natural progression in AI, explainability is key. It is paramount to guarantee that business outcomes entrusted to AI are comprehensible and subject to audit. As an example, in applications such as healthcare, for instance in cancer surgery, if AI decides to cut out a vital organ, and the surgeons are unable to understand the decision, they cannot risk the life of the patient. So if AI makes an incorrect decision, XAI provides a chance to identify the source and reason for the error and correct it. Another example is autonomous driving, if a self-driving car makes a bad decision it can be deadly. If an autonomous car faces an inevitable accident scenario, the decision it makes impacts greatly on its future use, whether it saves the driver or the pedestrians. Providing the rationale for each decision an autonomous car takes, helps to improve people's security on the road [9].

In numerous industries, explainability stands as a regulatory prerequisite for companies harnessing such models. Furthermore, with the General Data Protection Regulation (GDPR) [10] in effect, companies must furnish consumers with explanations for AI-driven decisions. Enterprise leaders harbor a curiosity to explore alternative possibilities and assess the business implications of their technological investments, but today there are uncertainties in leveraging AI models to accomplish this goal. XAI marks a pivotal stride in the evolution of sophisticated applications, promising a future where AI models

seamlessly integrate into existing business processes, enhancing outcomes. A time is on the horizon when these AI models can replace the traditional advanced analytics processes, adapting as businesses evolve, provided they remain under human governance and guidance. XAI will facilitate this transformative transition [8].

Time Series (TS) data is omnipresent in various natural phenomena, encompassing variables that change over time. Industries, particularly in fields like medicine, rely heavily on temporal data for crucial insights. Consequently, the application of ML techniques to such data is of particular importance, offering diverse practical applications. ML methods, when applied to TS, excel in tasks like classification, regression, forecasting, and clustering. DL methods, notably recurrent techniques with memory states and temporal convolutional layers in Convolutional Neural Networks, enhance accuracy and remove the need for extensive data preprocessing. However, the complexity of these advanced methods poses a challenge in terms of interpretability, a crucial concern for applications such as medical diagnosis and autonomous driving. The explainability of models applied on TS has not gathered much attention compared to the computer vision or the natural language processing fields. Achieving XAI for TS remains a pressing issue, distinct from the fields stated above, where intuitive understanding of data is more straightforward. Unlike interpreting images or text, humans are less accustomed to comprehending temporal data represented as signals evolving over time, making it challenging to assess explanations qualitatively. Addressing this challenge requires specialized knowledge and innovative techniques to uncover the underlying insights within temporal data [11].

In the realm of Multivariate Time Series (MTS) regression and forecasting, especially in fields like retail, medicine, and economics, heightened interpretability can yield practical benefits. Existing research often focuses on computing temporal variable importance to glean insights into datasets and models [12]. These importance scores serve purposes such as feature selection [13] and model compression [14]. For instance, the temporal variable importances can help identifying which temporal features are more important to the prediction model [15]. Techniques like SHapley Additive exPlanations (SHAP) offer nuanced local explanations, providing positive or negative insights into how features influence predicted values, unlike traditional methods that merely indicate feature importance without considering their contributions. For instance, Mokhtari et al. [16] utilized SHAP to interpret financial TS models, enabling experts to grasp the model's decisions more comprehensively [17].

This research delves into the realm of ML interpretability and explainability within TS regression, driven by the unique demands of diverse industries. Despite the growing attention towards XAI, there remains a notable gap in exploring interpretability techniques tailored for TS regression and MTS regression. Existing literature primarily emphasizes TS classification and forecasting tasks. In terms of the existing research on interpretability of TS, the focus primarily revolves around intrinsic explainability, and the absence of standardized evaluation metrics for local interpretability methods might be a

contributing factor behind the limited studies on deciphering TS regression models [4].

1.2 Main Objectives and Contributions

The main objective of this Thesis is to analyse post hoc XAI approaches for TS data, and achieve explainability in a MTS regression problem, understanding the decision-making process of the underlying DL model as well as draw actionable insights.

The context of the problem centers around trying to understand the foundational aspects of neural dynamics and overall brain organization, utilizing as an example a roundworm named *Caenorhabditis Elegans* for which the connectome is completely mapped, illustrating all of the neuronal connections in the nervous system. The used dataset is composed essentially by responses of the roundworm's nervous system under diverse stimuli, where both inputs and outputs are time series data with multiple features.

In the research process, besides providing an extensive literature review related to XAI and interpretability, a comprehensive analysis of two local model-agnostic explanation approaches, SHAP and Local Interpretable Model-agnostic Explanations (LIME), is performed, since in terms of explaining any black box model, these two methods are, by far, the most comprehensive and dominant across the literature methods for visualising feature interactions and feature importance [2]. Furthermore, SHAP is the chosen technique to be applied to the Gated Recurrent Unit (GRU) model, which provided local and global explanations, that were further validated through two comparative experiments, involving providing less input data during inference.

1.3 Organization of the Document

This thesis is organized as follows: **Chapter 1** provides the introduction, context and motivation for this work, main objectives and contributions, and describes the organization of the document. **Chapter 2** presents an extensive literature review in the context of XAI and ML interpretability, and the related work on XAI for TS data. **Chapter 3** goes deep into the theoretical definitions of local model-agnostic explanation methods, namely LIME and SHAP, and methods relative to them like DeepLIFT and Shapley Values, also stating their advantages and disadvantages. **Chapter 4** presents in a more technical way the explanation methods referred in Chapter 3, introducing their functions and visualization techniques. **Chapter 5** represents the analysis of the experimental setup, describing the data and model used and their implementation, as well as the implementation of DeepSHAP and of two comparative experiments. **Chapter 6** describes the obtained results from the SHAP analysis and the results from the two experiments, with room for a discussion at the end. **Chapter 7** states the conclusions and future work.

2

State of the Art and Related Work

Contents

It is of utmost importance to ensure that ML models make decisions for the right reasons. Comprehending the rationale behind a model's specific prediction can help users to debug, establish trust and further improve the existing model. This procedure is explored within the field of XAI, whose aim is to explain or present the model's decisions in a manner comprehensible to humans. One possible definition of interpretability is given by Tim Miller, that stated that "*Interpretability is the degree to which a human can understand the cause of a decision*" [18]. Interpretability refers to the ability to understand and explain the decisions, processes, and outcomes of complex ML models in a human-readable and understandable way, and it is important for several reasons, which will be introduced in the following sections, as well as the main aspects of XAI and Interpretability, their differences, the different existing methods, and their properties and evaluation.

2.1 Importance of Explainable AI and Interpretability

DL models are often referred to as black box models, which are known for being impossible to understand just by looking at their parameters since they are systems that conceal their internal mechanisms. Input features are fed into this model, which subsequently undertakes complex computations that ultimately yield decisions. Nevertheless, discerning precisely how the model arrives at its conclusions about the most important features, or even at what it looks at, is a very tough task. This challenge is amplified when dealing with Deep Neural Networks with more than 1 billion parameters, such as those employed in advanced language modeling algorithms, and interpreting the representations that the model has learned becomes an extremely complex challenge [4]. Consequently, delving into the foundation upon which these predictions are built emerges as a vital pursuit in the realm of AI developments. Ensuring the reliability of these models in real-world scenarios, validated through meaningful metrics, becomes imperative.

"*If a machine learning model performs well, why do we not just trust the model and ignore why it made a certain decision?*" [3]. The importance of algorithmic accountability has been emphasized on numerous occasions, one of them being a widely recognized and frequently referenced example, denoted in [1]. A DL system employed in image recognition tries to differentiate between images of wolves and husky dogs. However, due to consistent snowy backgrounds in wolf images, the distinguishing feature predominantly relies on the background. Consequently, when a husky is also photographed against a snowy background, it becomes susceptible to being misclassified as a wolf. In Figure 2.1 it is possible to visualize the informative segments within the image that guided the classifier's decisions, which sheds light on a limitation inherent in the learning base. Without the explanation methods, discerning that decisions are rooted in incorrect image regions would have been impossible to know. Thus, the conclusion from this is that the model should not be blindly trusted. In the absence of insight into the

basis upon which it formulated its decisions, distinguishing relevant features from noise or background elements becomes impossible. Hence, the key here lies in comprehending the inner workings of a model by generating explanatory insights [4].

Figure 2.1: Raw data of a Husky classified as a Wolf (left) and the explanation of a bad model's prediction in the "Husky vs. Wolf" classification task (right). Source: [1]

The example illustrated in Figure 2.1 underscores the critical importance of interpretability and explainability in ensuring algorithmic fairness, detecting potential biases or issues within the training data, and verifying the expected performance of algorithms.

In predictive modeling, there are models that probably do not require explanations because there are no serious consequences if mistakes happen. However, this exemption does not hold true for all of the existing problems. In many applications, especially those involving critical decisions like healthcare and legal matters, it is essential for humans to **trust** the decisions made by AI systems. Interpretability serves as the linchpin, since it allows humans to verify that the model's decisions are valid, fair, and aligned with ethical guidelines. **Fairness** refers to ensuring that predictions are unbiased and do not discriminate against marginalized groups. By understanding how the model arrives at its conclusions, it becomes easier to identify and address instances where the model might be unfairly favoring certain groups or making discriminatory decisions. Often, biased predictions originate from the ML model absorbing biases from the training data, making interpretability a useful debugging tool for detecting and rectifying biases in these ML models [3].

Certain objectives related to interpretability, not yet thoroughly discussed, can be succinctly delineated by referencing the works cited in [3, 19, 20]:

- **Privacy**: Ensuring protection of sensitive data against exposure.
- **Causality**: Ensuring detection of only causal relationships, aligning technical explainability with human understanding. This alignment is pivotal in facilitating meaningful interactions between humans and ML systems.

- **Reliability/Robustness**: Guaranteeing that minor input alterations do not lead to significant prediction changes. Its objective lies in fortifying ML systems against the influence of noisy inputs and shifts in their operating domains. This research direction closely correlates with challenges posed by domain adaptation and transfer learning.

For the purpose of becoming aware of the practicality of XAI thoroughly, it is essential to comprehend the trade-off between the efficiency of ML models and their ability to generate predictions that are interpretable and explainable. This trade-off, often termed as the *Model Interpretability vs. Model Performance trade-off*, is marked by the challenge of interpreting complex models accurately, in which the increasing of the complexity of the model increases the difficulty of correctly interpreting the model [3]. On one side, there are the opaque black box models, and on the opposing end there are the so-called white box or glass box models such as linear and decision tree-based models, which have a propensity for producing explainable results. Although these latter models are more amenable to explanation and interpretation, they do not attain the state-of-the-art performance levels exhibited by the former. This difference in performance and interpretability arises from their frugal design. In real-world scenarios, the ultimate objective is to obtain models that not only excel in performance but also offer easy interpretability. This is crucial for comprehending the decision-making processes and, if necessary, validating or rectifying them [2].

2.2 Explainability vs. Interpretability

Within the realm of ML, the terms $explainability$ and $interpretability$ are frequently employed interchangeably in the literature. Despite their apparent similarities and shared purpose, it remains crucial to delineate their disparities for a deeper comprehension of the concepts. This task is not trivial, as it lacks a concrete definition for interpretability or explainability, and they remain unquantified by an universally accepted metric [2]. Nonetheless, numerous attempts have arisen [21–23] in order to clarify these two notions. Among the various definitions of interpretability, Doshi-Velez and Kim's characterization in their research [22] as "*the ability to explain or to present in understandable terms to a human*" is widely acknowledged. It is worth noting that these explanations lack mathematical formality and rigorousness [24]. A distinction between interpretability and explainability will follow.

Interpretability, within the context of AI, denotes the ability to understand the decision-making process of an AI model [18]. An interpretable model operates transparently, offering insights into the connections between inputs and outputs. In essence, an interpretable algorithm can be explained clearly and understandably by a human being, therefore interpretability is crucial to ensure that users can understand and trust AI models. It is noteworthy that the model's interpretability is distinct from its ability to accurately represent reality, since a model, regardless of its performance, could still be easy to understand.

DL models possess the capability to represent complex, non-linear relationships and execute tasks that would be impossible for logic-based decision trees to perform. However, the very complexity of those models, which enables them to learn the non-linear relationships, is also the same reason why they may not be interpretable [4].

On the other hand, explainability pertains to the ability to explain the decision-making process of an AI model in terms understandable to the end user [25]. An explainable model provides a clear and intuitive rationale for the decisions it makes, facilitating user comprehension of why the model produced a particular result. Essentially, explainability focuses on the *why* behind an algorithm's specific decisions and the justifications underpinning those choices, and so it tries to provide post hoc explanations for pre-existing black box models, while interpretable ML is centered on formulating models that possess inherent interpretability [19].

In summary, while both interpretability and explainability are important for understanding AI models, they diverge in some key concepts:

- **Depth of insight**: Interpretability focuses on understanding the inner workings of the models, while explainability concentrates on explaining the decisions it arrives at. Consequently, interpretability requires a greater level of detail than explainability.
- **Model complexity**: More complex AI models, such as Deep Neural Networks, can prove challenging to interpret due to their complex structure and the interactions between different parts of the model. In such scenarios, explainability might offer a more feasible solution, as it prioritizes explaining decisions rather than understanding the model itself.
- **Communication**: Interpretability concerns the understanding of the model by AI experts and researchers, whereas explainability leans toward communicating model decisions to end users. As a result, explainability demands a simpler and more user-friendly manner of presenting information.

2.3 Taxonomy of Explainable AI and Interpretability

Differentiation among explanation methods in the realm of ML stems from their specific focuses within the ML domain. Several approaches shed light on data explanation, delving into the intricate aspects of dataset features. Conversely, alternative methods revolve around the model that utilizes the dataset, aiming to provide insights into its decision-making process [4].

In accordance to what was previously stated in Section 2.1, two primary model categories emerge: those referred to as white box, and then those referred to as black box. The concept of interpretability pertains to the former type, also recognized as glass box models, and is characterized as intrinsic. This form of interpretability covers all the models which have an inherently interpretable internal structure,

designed in such a way that their decisions can be explained due to their construction method, like a decision tree. For example, to build a model to predict whether someone will play tennis based on weather conditions. The conditions considered can be ”Outlook” (sunny, overcast, rainy), ”Temperature” (hot, mild, cold), and ”Humidity” (high, normal), and the target variable is ”whether the individual will play tennis” (yes, no). In this situation, the tree structure of that decision tree represents the decision-making process of the model, and it is interpretable since it is possible to follow the tree to understand how the decision was made. Additionally, they are generally simple and easy to interpret.

In contrast, black box models require a post hoc explanation through an external framework, independent of the model, post its creation. Post hoc interpretability techniques try to explain a model's prediction without explaining the exact internal mechanism of the model itself. Consequently, to make a black box model explainable, some techniques for extracting explanations from the model's internal logic or outputs must be adapted, including model-agnostic approaches or surrogate models [4].

Methods to achieve interpretability in ML can be categorized according to distinct criteria, leading to the availability of multiple taxonomies for addressing the field of XAI. One of these taxonomies, represented as a mind-map in Figure 2.2, delineates ML interpretability techniques, offering a valuable separation of these methods based on distinct classifying factors. A concise description of these factors follows on the next sections.

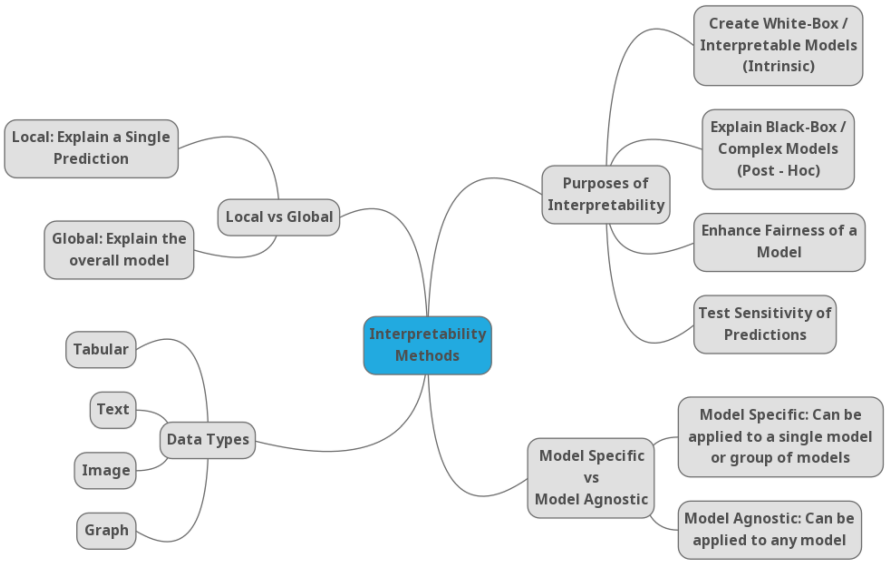

Figure 2.2: Taxonomy mind-map of ML Interpretability Methods. Source: [2].

2.3.1 Purposes of Interpretability

Intrinsic interpretability pertains to ML models that are considered understandable due to their simple structures, such as concise decision trees or sparse linear models, while post hoc interpretability, or

extrinsic interpretability, involves employing interpretation methods after model training, like permutation feature importance [3]. Meaning that this differentiation revolves around whether interpretability is achieved by simplifying the ML model's complexity during its construction (intrinsic), or by utilizing techniques that analyze the model subsequent to its training (post hoc) [4].

Methods that promote fairness are designed to eliminate bias from both training data and model predictions, and they aim to train models capable of generating impartial predictions originally from the beginning [26].

Techniques that work towards the analysis of the reliability of model predictions involve methods aimed at evaluating and questioning ML models, ensuring the credibility of their predictions. These methods employ sensitivity analysis to examine the robustness of learned functions and observe how output predictions change in response to subtle, deliberate alterations in the input parameters [4].

2.3.2 Model-Specific vs. Model-Agnostic

Interpretation tools in the realm of ML can vary significantly, falling into two broad categories: those tailored to specific model classes and those agnostic to model type. Model-specific interpretation tools, as the name implies, are restricted to particular models and are based on the details of the specific structures of the models. These approaches exploit the internals of the ML model like the neural network, and use reverse engineering approach to provide explanations of how the specific DL or ML algorithm is giving the relevant decision [27]. The primary advantage of employing model-specific techniques lies in their capacity to supply a deeper comprehension of the decision-making mechanisms of ML models, providing an intricate understanding of their internal operations. This, in turn, allows for the development of more customized and problem-tailored explainable models. Nevertheless, it is worth noting that certain model-specific techniques might augment the model's complexity. This trade-off between interpretability and performance, already stated previously in Section 2.1, becomes evident, as models with more complex structures, which typically yield more accurate predictions, become challenging to interpret due to the complexity of their decision-making processes [4]. One example of a model-specific approach to DL models is based on deconvolution. It is used to reverse the process of convolution, which traverses the path of the Convolutional Neural Network (that goes from image data to the final class) in reverse order, going from final class to original image, pointing out specific regions in the image that contribute to the decision [27].

On the other hand, model-agnostic tools have the versatility to be employed on any ML model, functioning after the model's training phase. These universally applicable techniques typically involve analyzing pairs of feature inputs and their corresponding outputs. It is important to note that by their very nature, these methods lack access to internal aspects of the model, such as weights or structural details [3]. They derive explanations by perturbing and altering the input data and observing how these

alterations affect the model's performance compared to the original data. Consequently, they unveil valuable insights into the specific localized regions within the input data that hold higher sensitivity. These techniques can be applied to any ML algorithm, offering flexibility, without influencing its overall performance [4].

2.3.3 Local vs. Global Interpretability

This criterion concerns whether the interpretation method explains an individual prediction (local) or the overall behavior of the entire model (global) [3].

The term $Local$ pertains to those approaches that explain specific predictions, and by concentrating on an individual case, these methods can potentially simplify the otherwise complex model, possibly revealing simpler, linear, or monotonic dependencies on certain features, rather than complex interdependecies [3].

Conversely, $Global$ pertains to methods and tools that offer insights into the entire model's interpretation. These methods unveil general trends regarding the impact of variables on the model's averaged output [4].

2.3.4 Output of Interpretation Methods

Interpretation methods within the realm of ML yield a spectrum of outcomes that are organized into different categories [3]:

- **Feature Summary Statistics**: Many methods produce summary statistics for individual features. These statistics can range from single values like feature importance to more complex results such as pairwise feature interaction strengths involving combinations of features.
- **Feature Summary Visualization**: Visualizations are used to represent various feature summary statistics. Some summaries only gain meaningful insight when presented visually, as in the case of partial dependence plots that illustrate the relationship between a feature and the average predicted outcome. Visual representations are preferred over tabular formats.
- **Model Internals**: This category involves interpreting intrinsically interpretable models. It encompasses understanding components like the weights in linear models or the learned tree structure in decision trees. Sometimes, the line blurs between model internals and feature summary statistics, as seen in linear models where weights serve both as internal model parameters and feature summaries.
- **Data Point Explanations**: Methods falling into this category offer data points, either existing or newly generated, to enhance model interpretability. Counterfactual explanations, for instance,

identify similar data points by altering features to induce relevant changes in predicted outcomes. Similarly, identifying prototypes of predicted classes aids in comprehension. However, generating new data points necessitates their own interpretability, more suitable for images and text than complex tabular data.

- **Intrinsically Interpretable Models**: One strategy for understanding black box models involves approximating them with interpretable models. These interpretable models can be understood by analyzing internal parameters or feature summary statistics.

By recognizing and classifying these outcomes, the landscape of interpretation methods becomes clearer, offering a comprehensive framework for understanding the various paths towards model interpretability.

2.4 Evaluation of Explainable AI and Interpretability

In the domain of ML, unanimous consensus on the interpretability concept is lacking, and the methods for measuring it remain unclear [3]. However, prior research have been conducted, leading to the development of evaluation techniques, as is the case of the work by Doshi-Velez and Kim (2017) [22]. A tripartite framework is put forth to measure interpretability, which can be split into the need or not of humans, the first one including **application-grounded** and **human-grounded evaluations**, and the latter named **functionally-grounded evaluation**.

In the **application-grounded evaluation** (real task), the explanation is immersed within the product and it is subjected to test by end-users. A concrete example can be a fracture detection software featuring a ML module designed to pinpoint and highlight fractures in X-ray images. At this application-grounded evaluation, radiologists would directly test the fracture detection software to assess its efficacy. This demands a well-structured experimental arrangement and a grasp of quality assessment methodologies. A reliable baseline here revolves around the efficacy of a human explaining the same decision [3].

At a **human-grounded evaluation** (simple task), it is entailed a simplified version of the application-grounded evaluation. The distinction lies in executing these experiments not with domain experts, but with individuals possessing basic knowledge. This approach yields cost-efficient experiments and facilitates recruitment of a larger pool of testers. For example, one could present different explanations to a user who then selects the best one.

Finally, the **functionally-grounded evaluation** (proxy task) removes the need for human involvement. This kind of evaluation is appealing because even general human-subject experiments require time and costs both to perform and to get necessary approvals (e.g., IRBs [28]), which is probably beyond the resources of a ML researcher. This methodology is most effective when the model category in

use has already undergone evaluation via human-grounded evaluation. For instance, if it is established that end-users comprehend decision trees, a proxy for measuring explanation quality could be the depth of the tree. Shorter trees would garner higher scores for explainability. A reasonable condition to impose would be that the predictive performance of the tree remains satisfactory and does not degrade significantly compared to a larger tree [3]. Taking the example of the decision tree model stated in Section 2.3 that can predict whether a person will play tennis that day and supposing that two models with the same end goal existed, if one is significantly larger than the other and both models present similar predictive performance, this functionally-grounded evaluation would output that the shorter tree has superior explanation quality than the larger due to its depth.

2.5 Properties of Explainable AI and Interpretability

When the goal revolves around clarifying the predictions of ML models, it is prudent to rely on an explanatory approach, which constitutes a computational procedure engineered to create explanations. These explanations commonly establish a link between the feature values of an instance and the model's prediction in a manner understandable to humans [3].

Robnik-Sikonja and Bohanec (2018) [29] educate on the traits of both explanatory techniques and individual explications, which can be employed to assess the quality of an explanatory approach or the explanation itself.

2.5.1 Properties of Explanation Methods

Expressive Power refers to the structure by which the approach generates explanations. The approach could produce if-then rules (propositional logic), decision trees, linear models, or other formats.

Translucency assesses the degree to which the explanatory approach depends on exploring the ML model. It can be decompositional (decomposes internal representation of the model, e.g., in neural networks meaning of individual neurons), pedagogical (treating the model as a black box), or eclectic (combining both compositional and pedagogical types). For instance, techniques rooted in intrinsically interpretable models, such as linear regression (model-specific), are highly translucent, allowing the approach to exploit more data to formulate explanations. Techniques that exclusively manipulate inputs and observe predictions possess no transparency, enhancing the portability of the explanatory approach due to their treatment of the ML model as a black box. Depending on the context, varying levels of transparency might be preferable [3].

Portability embraces the range of ML models compatible with the explanation technique. Surrogate models might present the utmost portability for an explanatory approach. Conversely, techniques tailored exclusively for particular models like Recurrent Neural Networks exhibit low portability.

Algorithmic Complexity pertains to the computational complexity of the explanation-generating approach. This characteristic is vital when computational time presents a limitation for explanation generation.

2.5.2 Properties of Individual Explanations

Quality of explanations is another aspect of utmost importance, which groups several properties of explanation methods:

- **Accuracy**: This property helps to understand the extent of an explanation's ability to predict unseen data. Is it notably crucial to obtain a high level of precision when utilizing the explanation for predictions in place of the ML model. Lower precision might be acceptable if the ML model's precision is also low, and if the aim is to explain the actions of the black box model. In this instance, only fidelity holds significance [3].

- **Fidelity**: Describes how well the explanation reflect the behavior of the prediction model. Among the foremost properties of an explanation, high fidelity is paramount and is one of the most important and desired properties, as a low-fidelity explanation proves futile in explaining a ML model. Accuracy and fidelity are closely intertwined. When the black box model exhibits high accuracy and the explanation has high fidelity, the explanation concurrently attains high accuracy [3]. Certain explanations only offer local fidelity, meaning that the explanation closely approximates the model's prediction solely for a subset of the data (e.g., local surrogate models) or even for an individual data instance (e.g., Shapley Values). Local fidelity does not imply general fidelity, meaning that features that are important in a local context may not be important in the global context of the model.

- **Consistency**: Reflects the magnitude of disparity between explanations derived from different models trained on the same task and that yield akin predictions. Models that share resemblances might generate similar predictions for a specific data point. However, the rationale behind these predictions could stem from distinct features due to the divergence in their explanatory techniques. Yet, if the rationales align closely, their consistency is evident, and consistency is desirable when models rely on akin relationships [3].

- **Stability**: Answers the question related to the extent of similarity of explanations for similar instances. Whereas consistency compares explanations across models, stability juxtaposes explanations for akin instances within a fixed model. High stability denotes that minor deviations in feature values do not significantly alter the explanation, unless these subtle variations similarly exert pronounced change in the prediction. A lack of stability might emerge from high variance within

the explanation method. In other words, the explanation method is strongly affected by slight adjustments in the instance's feature values to be explained. In addition, a lack of stability can also be induced by non-deterministic components within the explanation method, such as a data sampling step, like the local surrogate method uses. Optimal stability is always desirable [3].

- **Comprehensibility**: Represents how well humans understand the explanations. This property, although exceedingly important to get right, is arduous to define and measure. Methods to measure comprehensibility include quantifying the extent of the explanation (e.g., number of decision rules) or evaluating how effectively individuals can predict the behavior of the ML model based on the explanations. Moreover, the comprehensibility of the features employed in the explanation should merit consideration since a complex transformation of features might be less comprehensible than the original features [3].
- **Certainty**: Refers to understanding if the explanation mirrors the certainty of the ML model. Several ML models merely give predictions without indicating the model's assurance in the prediction's accuracy. An explanation that includes the model's certainty proves exceedingly valuable.
- **Degree of Importance**: Speaks to the extent to which the explanation reflects the significance of features or segments within the explanation.
- **Novelty**: Helps to find out if the explanation reflects whether a data instance to be explained originates from a "new" region far removed from the training data distribution. In such instances, the model might prove imprecise, rendering the explanation futile. The notion of novelty parallels the concept of certainty: greater novelty heightens the likelihood of low model certainty due to insufficient data [3].
- **Representativeness**: Concerns the quantity of instances that the explanation covers, it can be the entire model (e.g., interpretation of weights in a linear regression model) or exclusively represent an individual prediction (e.g., Shapley Values).

2.6 Related work on Explainable AI for Time Series Data

The work done by Natalia Jakubiak entitled "Analysis of Explainable Artificial Intelligence on Time Series Data" [4] focuses on XAI with a specific emphasis on its application in analyzing TS data. It is closely related to the work done in this thesis regarding XAI, since the author utilizes two notable model-agnostic attribution methods, LIME and SHAP, that are post hoc feature attribution methods that, for a single instance, provide an explanation by highlighting how important each feature at each time step in the input was to the final prediction. Additionally, the research investigated the use of attention mechanisms in forecasting models.

Attention mechanisms act as additional network layers, highlighting the model's focus on specific elements during different time steps. These mechanisms, although leading to more complex models and longer training times, provided valuable insights by filtering redundant information and emphasizing relevant data. Also, by extracting temporal relationships between inputs and predictions, they offered insights into feature importance for real-time Remaining Useful Life (RUL) predictions. Using the NASA turbofan engine dataset C-MAPSS, the thesis formulates the TS forecasting task as a prognosis for the RUL of a turbofan engine. Bi-LSTM architectures, both with and without attention layers, are employed for RUL prediction. Compared to SHAP, attention mechanisms offered comparable results, outperforming SHAP for the C-MAPSS dataset. The real-time interpretability of attention mechanisms, combined with their seamless integration into training processes, makes them highly attractive for model interpretability. The attention mechanism, integrated directly into the model structure, offered immediate, accurate, and clear explanations, albeit with a slight compromise in prediction accuracy. The post hoc methods, while computationally expensive, maintained model accuracy while offering interpretable explanations. The study suggested a potential combination of these methods for future research to enhance the accuracy and interpretability of ML models.

The thesis then delves into generating explanations for these predictions, which are visualized for representative engine conditions (Cases A and B, which represent the engine in a good condition, and the engine with signs of its service life coming to and end, respectively) to ensure user comprehensibility.

A novel method to evaluate explanation fidelity, considering TS regression tasks, is introduced. This approach involves altering data through perturbation methods, considering the inter-dependency of time points, and comparing model outputs for original and perturbed instances. By assessing the quality metric for all changed inputs using various XAI methods, the thesis quantitatively measures the local fidelity of explanations, ensuring they accurately represent the model's behavior in specific contexts.

The study's findings contribute significantly to the integration of explainability into ML models, particularly in real-time applications such as predicting the RUL of machines, and develops a robust evaluation method for explanation fidelity. This work contributes to bridging the gap between XAI and TS data applications, crucial for enhancing the transparency and trustworthiness of AI models in real-world scenarios.

In the paper entitled "Evaluation of Interpretability Methods for Multivariate Time Series Forecasting" [17] by Ozan Ozyegen, Igor Ilic, and Mucahit Cevik, the authors conducted a comprehensive study on interpretability methods for TS forecasting models. The study involved the comparison of three models — LSTM, TDNN, and GBR — across four diverse datasets: Electricity, Rossmann, Walmart, and Ohio. Performance evaluations were conducted using metrics like Normalized Root Mean Squared Error (NRMSE) and Normalized Deviation (ND), and GBR consistently outperformed LSTM and TDNN models, demonstrating superior predictive accuracy across most datasets.

In the realm of interpretability, the researchers examined three methods: Omission (Global), Omission (Local), and SHAP, alongside a random baseline. SHAP emerged as a standout performer, especially for tree-based models like GBR. The global omission method, simpler computationally, also proved effective. The study introduced two evaluation metrics, AOPCR and APT, to assess local fidelity. SHAP consistently demonstrated high performance, although the choice of the method depended on the specific experimental goals. Additionally, the study revealed that local explanations could potentially function as feature selection methods, aiding in the identification of significant features within datasets.

A crucial aspect of the research was the impact of feature selection on model performance, SHAP's ability to identify significant features was compared with other methods such as Mutual Information, F-Statistic, and Tree Importance. SHAP consistently proved effective as a feature selection method, accurately identifying relevant features. Moreover, using a subset of features significantly reduced model training times while maintaining reasonable accuracy.

The study also delved into sensitivity analysis, exploring how the effectiveness of interpretability methods, particularly SHAP, was influenced by the predictive model's performance. The researchers found that local explanations generated for LSTM models were more sensitive to model hyperparameters compared to GBR models, and, interestingly, GBR models exhibited a robust ranking of local explanations across different datasets and evaluation metrics.

In summary, the study highlighted the critical importance of choosing appropriate interpretability methods based on specific task requirements. SHAP emerged as a powerful tool for understanding complex TS forecasting models, particularly those based on tree algorithms like GBR. These findings provide valuable insights for researchers and practitioners, significantly advancing the field of interpretability in TS forecasting.

The paper entitled "TimeSHAP: Explaining Recurrent Models through Sequence Perturbations" [30] (Bento et al., 2021) addresses the need for model transparency in the context of recurrent models, particularly in fraud detection. These models, like Recurrent Neural Networks, are highly effective but often considered opaque, which raises concerns among stakeholders. To address this, the authors introduce TimeSHAP, a method for explaining recurrent model predictions in sequential tasks. TimeSHAP is a model-agnostic, post-hoc explainer designed to work with recurrent models. It builds upon the principles of KernelSHAP, a popular attribution method, and extends its application to sequential data. TimeSHAP provides three types of explanations: 1) Feature-level explanations, where it can attribute importance to individual features, helping to understand the role of specific input features in the model's predictions; 2) Event-level explanations, since in sequential data it is not only important to know which features matter but also which events in the sequence contribute the most to the prediction, and so TimeSHAP can isolate events that significantly impact the model's decision; 3) Cell-level explanations, which dive even

deeper by attributing importance to individual cells in the input matrix, allowing a granular understanding of the model's behavior.

One significant challenge when explaining recurrent models is the exponential increase in event coalitions as the sequence length grows. To address this issue, TimeSHAP introduces a temporal coalition pruning algorithm, which groups older, less relevant events into a single coalition, significantly reducing computational costs while preserving essential information. This strategy enables efficient explanation generation for long sequences.

The authors validate TimeSHAP on a practical fraud detection model used in a real-world financial institution. The model comprises a GRU layer and a feedforward classifier, achieving high recall rates at low false positive rates. TimeSHAP provides insights into the model's decision-making process: 1) Identification of Fraud Patterns: TimeSHAP confirms the model's ability to detect a common fraud pattern consisting of enrollment, login, and transaction events in account takeover cases. This alignment with domain knowledge enhances trust in the model's capabilities; 2) Sequential Understanding: The explanations highlight that, on average, the most recent event contributes to only $41\,\%$ of the model's decision, indicating the model's ability to consider historical events. This contrasts with some other post-hoc methods that might assign $100\,\%$ importance to the latest event; 3) Potential Discriminatory Reasoning: TimeSHAP reveals notably high attribution to the client's age, hinting at potential discriminatory reasoning. This observation is later confirmed in a bias audit, emphasizing the importance of ethical considerations in model development.

In summary, TimeSHAP presents a valuable tool for making recurrent models more transparent and interpretable. It offers a balance between model-agnostic interpretability, post-hoc analysis, and the ability to handle sequences efficiently. Through a real-world case study, the paper demonstrates the practical utility of TimeSHAP in explaining complex recurrent models used in tasks like fraud detection.

In the paper entitled "SeqSHAP: Subsequence Level Shapley Value Explanations for Sequential Predictions" [31], the authors introduce SeqSHAP, a novel Shapley value-based explainer designed to enhance the interpretability of ML models in sequential scenarios. SeqSHAP is motivated by the need for more intuitive and context-aware explanations in domains where data is inherently sequential, such as e-commerce transactions, healthcare records, cybersecurity, and natural language processing. SeqSHAP addresses several key challenges faced by traditional explanation methods.

Unlike conventional methods that offer explanations at the feature or element level, SeqSHAP takes a unique approach by explaining model predictions in terms of subsequences. This subsequence-level interpretation is rooted in human cognition and aligns with how individuals naturally identify patterns in sequences of events. By grouping related events into sessions or subsequences, SeqSHAP provides more meaningful and intuitive explanations for end-users, which is particularly crucial in applications

where users need to take action based on model recommendations, like fraud detection in e-commerce transactions.

To ensure that the generated subsequences encapsulate meaningful contextual information, SeqSHAP introduces a distribution-based segmentation method. This approach is far more sophisticated than conventional segmentation techniques like fixed windows. Instead, it maximizes the distribution discrepancy among adjacent subsequences, ensuring that they contain distinct contextual details. This approach reflects the complexities and nuances of sequential data, enhancing the quality of the explanations.

The authors conduct rigorous experiments on real-world transaction datasets to demonstrate the effectiveness of SeqSHAP. The results establish the method's reliability and its potential for deployment in practical applications. Additionally, user studies confirm that SeqSHAP's subsequence-level explanations resonate with human intuition, underscoring its usability and practicality.

SeqSHAP overcomes the limitations of traditional element-wise explanations, which can overwhelm users with excessive detail. By focusing on meaningful subsequences, SeqSHAP streamlines the information, making it more digestible and actionable for end-users. Moreover, SeqSHAP optimizes computational efficiency by reducing the size of the feature space while maintaining precision, which is crucial for scalability and real-time applications.

While the paper's primary focus is on fraud detection in e-commerce transactions, SeqSHAP's subsequence-level explanation approach has the potential to benefit a wide range of sequential applications, including healthcare, cybersecurity, and natural language processing. Beyond improving interpretability, SeqSHAP's insights into the importance of specific subsequences can guide model refinement and enhance the overall performance of sequential models.

In conclusion, SeqSHAP represents a significant leap forward in XAI, especially in the context of sequential data analysis. Its emphasis on subsequence-level explanations and the innovative distribution-based segmentation method promises to unlock a deeper understanding of sequential data, ultimately leading to more reliable and actionable ML predictions.

The research by Jalali, Schindler, Haslhofer and Rauber, named "Machine Learning Interpretability Techniques for Outage Prediction: A Comparative Study" [32], delves into the critical task of explaining decisions made by black-box models, specifically Deep Neural Networks, when applied to sensor data, focusing on the PHM08-CMAPSS dataset. This dataset, a benchmark in Predictive Maintenance (PdM), contains run-to-failure data from 218 similar aircraft engines. The dataset is extensive, featuring operational settings, vibration data collected from sensors, and binary attributes, making it a robust basis for this study.

In the pursuit of interpretability, the research employs a range of techniques. The models under

consideration include both shallow and complex architectures. Shallow models, represented by Support Vector Regression (SVR) and interpretable models like Bayesian linear regression and decision trees, are contrasted with complex black-box models such as Long-Short Term Memories (LSTMs) and Gated Recurrent Networks (GRUs). These models undergo rigorous training on the preprocessed sensor data. To evaluate the performance of these models, the researchers employ Root Mean Squared Error (RMSE). While SVR exhibits promising results, challenges emerge in the models' ability to provide accurate predictions, especially at the early stages of an engine's life. This difficulty arises due to the intricate variations present in the sensor data during this period.

The interpretability techniques under scrutiny include LIME, SHAP, Partial Dependency Plot (PDP), and Accumulated Local Effect (ALE). These methods are systematically applied to unravel the decisions made by the trained models. LIME emerges as the frontrunner in this analysis, it excels in providing comprehensible explanations, focusing on specific sensors and their behavior. However, it also highlights the models' challenges in handling noise in the signals. On the other hand, SHAP offers valuable insights into influential features and their impact on predictions. Despite its effectiveness, it grapples with computational complexity, and certain model architectures, particularly those involving Recurrent Neural Networks (RNNs), pose challenges for this technique. PDP and ALE, while offering global explanations, come with limitations. These approaches, designed for tabular data, struggle when applied to RNN architectures like LSTMs and GRUs.

In summary, the study underscores the paramount importance of interpretability in the realm of black-box models, especially concerning critical applications like Predictive Maintenance. LIME stands out as the most effective approach in this context, shedding light on the decisions made by models operating on intricate sensor data. However, it is crucial to acknowledge the challenges posed by noise and the nuances of specific sensor behaviors, emphasizing the need for ongoing research to enhance the interpretability of complex machine learning models, particularly in the domain of TS data.

In the research paper entitled "TS-MULE: Local Interpretable Model-Agnostic Explanations for Time Series Forecast Models" [33] by Schlegel, Vo, Keim and Seebacher, they presented TS-MULE, an approach in the realm of interpretable ML, focusing on TS data.

TS-MULE is an extension of the LIME approach, meticulously designed to address the intricate challenges of understanding univariate and MTS data. It incorporates five novel segmentation algorithms and introduces three replacement strategies, providing nuanced insights into complex temporal patterns. The segmentation strategies include static windows and exponential windows, leverage the matrix profile and SAX transformation. The matrix profile-based techniques — slope segmentation and bin segmentation — employ sophisticated algorithms to identify significant changes and local trends within the TS data. SAX segmentation, on the other hand, utilizes horizontal binning to discern variations in the value

range, facilitating the creation of meaningful segments. Regarding performance evaluation, the segmentation algorithms are rigorously evaluated using perturbation analysis for fidelity, where they assess their effectiveness across multiple datasets, including Beijing Air Quality 2.5, Beijing Multi-Site Air Quality, and Metro Interstate Traffic data. Through comprehensive evaluations, they demonstrate that TS-MULE outperforms random perturbations, thereby substantiating the relevance of the identified input values for accurate model predictions.

In conclusion, TS-MULE represents a significant leap in the field of interpretable ML for TS data. Their meticulous segmentation techniques, coupled with rigorous evaluations, underscore the methodology's efficacy in providing meaningful and accurate explanations for TS forecasting models.

Even though researches and studies on XAI for TS data are growing more than ever, a major setback for this Thesis is the lack of work done focused on MTS data for regression. This work is centered in trying to explain the decision-making process of a GRU model for a regression problem, where the inputs are 4 different TS features, and the outputs are 4 different TS features. However, the existing related work in XAI for TS data prioritizes forecasting and classification problems.

3

Local Model-Agnostic Explanation Methods

Contents

Model-agnostic interpretation approaches, as opposed to methods tailored to particular models, provide elevated flexibility. Such methodologies are universally applicable to ML models regardless of the algorithms they employ. Researchers can exercise the freedom to choose the algorithms they deem optimal for addressing specific inquiries. They can then subject their models to consistent interpretation approaches, yielding improved model contrasts. These methodologies deploy post hoc interpretation techniques to deconstruct trained ML models [2]. The general concept revolves around explaining model behaviors by observing alterations in model predictions resulting from manipulations of input data. This stands in contrast to breaking down the models for insights into model architectures [34].

Apart from the model-agnostic techniques, there exist two alternative pathways: interpretable models and model-specific methods. It is generally observed that employing interpretable models requires the abdication of predictive performance relative to other ML models. A similar drawback of being confined to a single model type applies to model-specific methods. Model-agnostic approaches, on the other hand, offer explanation flexibility - unbounded by a specific explanation format - and representation flexibility - allowing for diverse feature representations, akin to the represented model. Within this category, a subdivision into local and global approaches further distinguishes these methods [3].

Global techniques represent the average behavior of a ML model, and are frequently exhibit as expected values grounded in the data distribution. Such methodologies prove exceptionally valuable when the aim pertains to comprehending fundamental data mechanisms or debugging a model [3]. Conversely, local methods aim to explain individual predictions.

In this Thesis only local model-agnostic explanation methods will be deeply explored, with its theoretical basis described following the work by Christoph Molnar [3]. These methods were selected since they are believed to be good options to fit well to deep neural networks and suitable for TS regression problems, that is the domain of the problem this work is hoping to solve. It is noteworthy that even though SHAP is a post hoc attribution method that can help models to obtain local explainability elucidating individual predictions, being therefore considered a local explanation technique, it can also provide global explanations. These local and global interpretation methods of SHAP will be further explored in Chapter 4, as well as LIME.

3.1 LIME

Local surrogate models are interpretable analytical tools employed to explain individual predictions of black box ML models. The concept, pioneered in the study entitled ”Why Should I Trust You?: Explaining the Predictions of Any Classifier” [1] by Ribeiro et al. in 2016, introduced a specific implementation of these local surrogate models known as LIME.

LIME, a popular interpretability technique, approximates any complex ML model with a local, inter-

pretable model, elucidating each specific prediction [2]. These surrogate models are tailored to mimic prediction patterns generated by the underlying black box model. In contradiction to training a global surrogate model, the LIME framework concentrates on training local surrogate models tailored to explain singular predictions [3].

The initial step involves disregarding the training dataset, allowing unrestricted exploration of the black box model. The primary objective is to understand the rationale behind the ML model's precise prediction. LIME conducts experiments by presenting diverse inputs to the black box model and observing the resulting predictions. This process generates a novel dataset, comprising altered samples along with corresponding black box predictions. LIME then utilizes this dataset to train an interpretable model, weighted by the proximity of these new instances to the original instance of interest [2]. The learned model's purpose is not to provide a comprehensive global approximation but rather to accurately replicate local black box model predictions (local fidelity) [3].

Mathematically, the formulation of local surrogate models subject to interpretability constraint can be articulated as follows:

$$explanation(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g) \tag{3.1}$$

Here, the loss function L assesses the proximity of the explanation to the prediction of the original model f, while $\Omega(g)$ maintains the simplicity of the model g (e.g., linear regression model) by favoring fewer features. The set G encompasses potential explanation models, including all conceivable linear regression models, for example. g operates based on the presence or absence of interpretable components. Notably, not all models within G are necessarily simple enough to be considered interpretable. To assess this, $\Omega(g)$ acts as a measure of complexity (as opposed to interpretability) for the explanation $g \in G$. For instance, in the context of linear models, $\Omega(g)$ might denote the count of non-zero weights, whereas for decision trees, it could signify the depth of the tree. Furthermore, the proximity metric π_x establishes the scope of the neighborhood around instance x considered for explanation. The measure L evaluates the dissimilarity of g in approximating f within the defined locality of π_x. To strike a balance between interpretability and local fidelity, the goal is to minimize L while ensuring that $\Omega(g)$ remains sufficiently low for human interpretability. This trade-off between the model fidelity and the complexity is a fundamental challenge encountered in LIME. This framework optimizes the loss component exclusively, leaving the determination of complexity to the user, which can be achieved by specifying the maximum number of features the model is allowed to utilize [3]. This approach renders LIME adaptable, allowing its application with various sets of interpretative models G, fidelity metrics L (such as MSE), and complexity assessments denoted by Ω [4]. The process of developing surrogate models at a local level involves the following steps [3]: **1)** Selection of the instance of interest, requiring an explanation for its black box prediction. **2)** Introducing slight dataset modifications (perturbations) and obtaining black box predictions for these new points. **3)** Assigning varying weights to the new data points based on their proximity to the

instance of interest. **4)** Constructing an interpretable model with these weighted instances and training it using the modified dataset. **5)** Interpreting the prediction outcome by understanding the local model.

Transitioning to the diversification of data variations, the approach hinges on the data type — namely text, images, or tabular data. For text and images, a solution involves activating or deactivating individual words or superpixels. In the case of tabular data, the LIME technique generates fresh samples through the individual perturbation of each feature. Perturbations are drawn from a normal distribution characterized by the feature's mean and standard deviation. Pertaining to LIME samples, instead of being drawn around the instance of interest, they originate from the centroid of the training data. This particular aspect introduces complications, but it increases the likelihood of disparities between predictions for certain sample points and the specific data point of interest. Consequently, LIME gains the capacity to acquire a degree of insight [3].

3.1.1 Advantages

Even in the event of substituting the underlying ML model, the utilization of an identical localized and interpretable model for explication remains feasible, and LIME distinguishes itself by being one of the scarce methodologies functional across tabular data, textual content, and images [3].

In situations involving short trees, the resulting explanations are selective and possibly contrastive. Consequently, they yield explanations that are easily comprehensible to humans, which is good in situations where the intended audience lack expertise or is constrained by time. [3].

The fidelity metric (assessing the interpretable model's proximity to black box predictions) provides a reliable indicator of the interpretable model's dependability in explaining black box predictions in the vicinity of the specific data instance of interest [3].

3.1.2 Disadvantages

While LIME is powerful and user-friendly, it is not without its limitations. A theoretical examination of LIME conducted in 2020 [35] underscored its importance and meaningfulness. However, this analysis also revealed that sub optimal parameter selections might cause LIME to miss out on crucial features [2].

The accurate characterization of the neighborhood remains a complex challenge when employing LIME alongside tabular data. This particular concern stands as the biggest obstacle within the domain of LIME, necessitating a prudent approach to its utilization [3].

Improvements could be made to the existing LIME implementation regarding the sampling methodology. Presently, data points are drawn from a Gaussian distribution, without due regard for inter-feature correlations. Consequently, improbable data points might emerge, subsequently employed to construct local explanation models [3].

A substantial problem surfaces concerning the instability inherent in the explications. As evidenced in [36], explanations between closely situated points yielded considerable discrepancies in simulated scenarios. This instability engenders skepticism, causing difficulty in trusting the explanations, and demanding a specially sensitive interpretation [3].

3.2 DeepLIFT

Deep Learning Important FeaTures (DeepLIFT) stands as a widely recognized recursive prediction explanation technique tailored for DL, conceptualized by Avanti Shrikumar and colleagues in 2017 [37]. This method explains the deviation from the output's reference value, Δt, concerning the disparity from the input's reference value, Δx_n. This difference enables the method to transmit an important signal, even in scenarios where the gradient is zero. Moreover, the use of a continuous reference difference helps alleviate gradient irregularities associated with bias terms.

DeepLIFT operates by decomposing a neural network's output prediction concerning a specific input. It achieves this by backpropagating the contributions of all neurons in the network to each feature of the input. The activation of every neuron is compared with its reference activation, and contribution scores are assigned based on the reference difference.

DeepLIFT relies on a mathematical property, expressed by the formula down below, which embodies a "summation to delta" principle:

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t \tag{3.2}$$

In this equation, t signifies the activation of a specific interest neuron, while $x_1, ..., x_n$ denote neurons within an intermediate layer essential for computing t. Considering t_0 as the reference activation for t, the term Δt represents the reference difference from this baseline and is calculated as $\Delta t = t - t_0$. $C_{\Delta x_i \Delta t}$ quantifies the contribution score for Δx_i by indicating how much of reference difference of t can be attributed to the reference difference in x_i. DeepLIFT employs a linear composition rule to linearize the nonlinear elements within a neural network [4].

3.3 Shapley Values

A prediction arises through the concept of considering every individual feature value within the case as an active participant in a hypothetical scenario resembling a game, where the ultimate prediction becomes the awarded sum. The equitable distribution of this 'reward' across the distinct features is guided by Shapley values, a technique originating from the domain of coalitional game theory, first introduced by Lloyd Shapley in 1953 [38].

The curiosity lies in understanding the influence of individual features on data point predictions. In a linear model, the computation of these individual effects is straightforward. Below illustrates the prediction of a linear model for a specific data instance:

$$\widehat{f}(x) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p \tag{3.3}$$

where x is the instance for which the contributions will be computed. Each x_j is a feature value, with $j = 1, ..., p$. The β_j represents the weight assigned to feature j.

The ϕ_j, indicating the contribution of the j^{th} feature on the prediction $\widehat{f}(x)$, is calculated as follows:

$$\phi_j(\widehat{f}) = \beta_j x_j - E(\beta_j X_j) = \beta_j x_j - \beta_j E(X_j) \tag{3.4}$$

where $E(\beta_j X_j)$ represents the mean effect estimate pertaining to feature j. Essentially, it quantifies the deviation between the feature's effect and the average effect. Remarkably, this method reveals the extent of each feature's impact on the prediction. When the sum of all the feature contributions for a given instance is computed, the following is obtained:

$$\begin{aligned} \sum_{j=1}^{p} \phi_j(\widehat{f}) &= \sum_{j=1}^{p} (\beta_j x_j - E(\beta_j X_j)) \\ &= (\beta_0 + \sum_{j=1}^{p} \beta_j x_j) - (\beta_0 + \sum_{j=1}^{p} E(\beta_j X_j)) \\ &= \widehat{f}(x) - E(\widehat{f}(X)) \end{aligned} \tag{3.5}$$

This outcome represents the predicted value for data point x adjusted by the average predicted value. It is worth noting that feature contributions can take on negative values.

The question is if this approach can be applied universally, irrespective of the model type. The prospect of having a model-agnostic tool is appealing, but however, since other model types typically lack analogous weights, a different solution must be explored. The Shapley value is a solution which offers a way to compute feature contributions for individual predictions for any ML model.

The Shapley value, as defined by the value function val assigned to players within the set S, signifies the contribution of a feature value to the overall payout. This contribution is meticulously determined by weighting and summing it across all possible combinations of feature values:

$$\phi_j(val) = \sum_{S \subseteq \{1,...,p\} \setminus \{j\}} \frac{|S|!(p - |S| - 1)!}{p!} (val(S \cup \{j\}) - val(S)) \tag{3.6}$$

where S denotes a subset encompassing the features utilized in the model, x represents the feature values vector of the instance being explained, and p denotes the total number of features. $val_x(S)$

corresponds to the prediction for feature values situated in the set S, effectively marginalizing those features not included in S.

$$val_x(S) = \int \widehat{f}(x_1, ..., x_p) d\mathbb{P}_{x \notin S} - E_X(\widehat{f}(X)) \tag{3.7}$$

It is worth noting that multiple integrations are executed for each feature outside of set S. For instance, consider a ML model operating with four features: x_1, x_2, x_3 and x_4. When assessing the prediction for the coalition S, which comprises the feature values x_1 and x_3, the process bears a resemblance to the feature contributions in a linear model:

$$val_x(S) = val_x(\{1, 3\}) = \int_{\mathbb{R}} \int_{\mathbb{R}} \widehat{f}(x_1, X_2, x_3, X_4) d\mathbb{P}_{X_2 X_4} - E_X(\widehat{f}(X)) \tag{3.8}$$

It is important to distinguish between the various contexts in which the term $value$ is employed:

- **Feature value** pertains to the numerical or categorical value of a feature within a given instance;
- **Shapley value** represents the feature's contribution to the prediction;
- **Value function** characterizes the payout function for coalitions of players, more specifically, feature values.

The Shapley value stands out as the only attribution approach that satisfies a set of essential properties, namely **Efficiency**, **Symmetry**, **Dummy** and **Additivity**. These properties collectively provide a robust definition of equitable payout:

- **Efficiency**: The sum of feature contributions must precisely match the difference between the prediction for x and its average:

$$\sum_{j=1}^{p} \phi_j = \widehat{f}(x) - E_X(\widehat{f}(X)) \tag{3.9}$$

- **Symmetry**: It is expected that two feature values, denoted as j and k, yield identical contributions when they equally influence all possible coalitions. This equivalence is represented by the condition

$$val(S \cup \{j\}) = val(S \cup \{k\}) \tag{3.10}$$

for all

$$S \subseteq \{1, ..., p\} \setminus \{j, k\} \tag{3.11}$$

Consequently, when this condition is met, it follows that

$$\phi_j = \phi_k \tag{3.12}$$

- **Dummy**: A feature j that exerts no discernible influence on the predicted outcome regardless of the coalition it joins, it is anticipated to possess a Shapley value of 0. This is confirmed when the following condition holds true

$$val(S \cup \{j\}) = val(S) \tag{3.13}$$

for all

$$S \subseteq \{1, ..., p\} \tag{3.14}$$

Resulting in

$$\phi_j = 0 \tag{3.15}$$

- **Additivity**: For a game with collective payouts represented as $val + val^+$, the corresponding Shapley values can be represented as:

$$\phi_j + \phi_j^+ \tag{3.16}$$

Consider this scenario: a random forest is trained, where predictions are derived from the average of numerous decision trees. The concept of Additivity offers assurance that when evaluating the Shapley value for a specific feature value, there is a flexibility to compute it separately for each tree, take their average, and thereby determine the Shapley value for that feature within the random forest.

To compute the exact Shapley value, it is necessary to assess all conceivable combinations of feature values, both with and without the inclusion of the j^{th} feature. However, when dealing with a substantial number of features, finding an exact solution becomes increasingly challenging due to the exponential growth in the number of potential combinations as more features are introduced. In an attempt to address this issue, Strumbelj et al. (2014) [39] introduced an approximation method employing Monte-Carlo sampling:

$$\widehat{\phi}_j = \frac{1}{M} \sum_{m=1}^{M} (\widehat{f}(x_{+j}^m) - \widehat{f}(x_{-j}^m)) \tag{3.17}$$

where $\widehat{f}(x_{+j}^m)$ represents the prediction for x, but with the substitution of certain feature values by those from a randomly selected data point z, except for the specific value associated with feature j. The x-vector x_{-j}^m closely resembles x_{+j}^m, although x_j^m is also derived from the sampled z. These M new instances assimilate with "Frankenstein's Monster", constructed by merging elements from two different instances. It is important to note that in the subsequent algorithm, the order of features remains unchanged; each feature retains its original position within the vector when passed to the predict function. However, the order is strategically employed as a "trick" in this context. By assigning new orders to the features, a random mechanism is created, facilitating the assembly of these "Frankenstein's Monsters". Features situated to the left of feature x_j retain their values from the original observations, while features

to the right obtain their values from a randomly selected instance.

Approximate Shapley estimation for a single feature value involves the following components:

- **Outcome**: Shapley value corresponding to the j^{th} feature value
- **Required**: Number of iterations denoted as M, specific instance of interest x, feature index j, data matrix X, and the ML model represented as f
  - For each $m = 1$ to M:
    - Select a random instance labeled as z from the data matrix X
    - Create a random permutation o of the feature values
    - Arrange instance x as follows: $x_o = (x_{(1)}, ..., x_{(j)}, ..., x_{(p)})$
    - Arrange instance z as follows: $z_o = (z_{(1)}, ..., z_{(j)}, ..., z_{(p)})$
    - Formulate two novel instances:
      - With feature j: $x_{+j} = (x_{(1)}, ..., x_{(j-1)}, x_{(j)}, z_{(j+1)}, ..., z_{(p)})$
      - Without feature j: $x_{-j} = (x_{(1)}, ..., x_{(j-1)}, z_{(j)}, z_{(j+1)}, ..., z_{(p)})$
    - Compute the marginal contribution: $\phi_j^m = \widehat{f}(x_{+j}) - \widehat{f}(x_{-j})$
- Determine the Shapley value as the average: $\phi_j(x) = \frac{1}{M} \sum_{m=1}^{M} \phi_j^m$

First, an instance of interest denoted as x should be selected, as well as a feature labeled as j, and also the number of iterations M should be decided. For every iteration, a random z instance is drawn from the dataset, and a random arrangement of the features is generated. Next, two new instances are formed by merging values from the x instance and the z sample. One of these new instances, x_{+j}, represents the focal instance, but it substitutes all values in the feature order beyond j with feature values from z. The other new instance, x_{-j}, is identical to x_{+j} but additionally replaces feature j with the value corresponding to feature j from the z sample. The discrepancy in the prediction from the black box model is calculated as follows:

$$\phi_j^m = \widehat{f}(x_{+j}) - \widehat{f}(x_{-j}) \tag{3.18}$$

All the discrepancies are then averaged, yielding the result:

$$\phi_j(x) = \frac{1}{M} \sum_{m=1}^{M} \phi_j^m \tag{3.19}$$

It is important to note that this averaging inherently assigns different weights to the samples based on the probability distribution of X. This entire procedure must be iterated for each of the features in order to obtain all the Shapley values.

3.3.1 Advantages

The concept of fairly distributing the difference between a prediction and the average prediction across the feature values of an instance, known as the Efficiency property of Shapley values, sets it apart from alternative techniques like LIME. Unlike LIME, which lacks the assurance of equitable feature distribution in predictions, the Shapley value stands out as potentially the exclusive approach capable of delivering a complete explanation. Particularly in contexts mandating explainability, such as the EU's "right to explanations", the Shapley value could potentially emerge as the sole legally compliant methodology due to its robust theoretical foundation and equitable effect distribution.

The Shapley value introduces the possibility of contrastive explanations, enabling comparisons not just against the dataset's average prediction but also against subsets or even individual data points. This contrastive capability distinguishes it from local models like LIME.

The Shapley value enjoys the unique distinction of being the only explanation method grounded in a rigorous theoretical framework, supported by axioms such as efficiency, symmetry, dummy, and additivity. In contrast, methods like LIME rely on an assumption of local linear behavior in ML models without a comprehensive theoretical foundation.

3.3.2 Disadvantages

The computation time demands of the Shapley value are substantial. In nearly all real-world scenarios, achieving an exact solution is practically implausible due to computational costs. The exact calculation of the Shapley value is computationally expensive due to the existence of a vast number of potential coalitions (2^k) deriving from feature value combinations. Additionally, simulating the "absence" of a feature requires drawing random instances, which elevates the variance in Shapley value estimates. The exponential magnitude of possible coalitions is managed through coalition sampling and the imposition of an iteration limit denoted as M. The reduction of M can expedite computations but concurrently amplifies Shapley value variance. There is no good rule of thumb for the selection of an appropriate number of iterations M, and so there is a need to balance between accurate Shapley value estimation (larger M) and computational efficiency (smaller M).

It is important to note that the Shapley value can be subject to misinterpretation. It should not be misconstrued as the difference of the predicted value after excluding from the model training. The proper interpretation of the Shapley value is as follows: Given the present set of feature values, it quantifies a feature value's contribution to the difference between the actual prediction and the mean prediction.

An additional drawback is that computing the Shapley value for a new data instance necessitates access to the dataset itself. Access to the prediction function alone is insufficient, as it mandates the replacement of segments within the target instance with values derived from randomly selected instances

within the dataset. This requirement is only avoidable if synthetic data instances resembling actual ones can be generated.

3.4 SHAP

SHAP, an abbreviation for SHapley Additive exPlanations as introduced by Lundberg and Lee in their work [40], presents a technique aimed at explaining individual predictions. The foundation of SHAP rests upon the optimal Shapley values derived from game theory explained in Section 3.3.

The authors of SHAP proposed KernelSHAP (Linear LIME + Shapley Values), an alternative methodology grounded in kernels, offering a novel approach for approximating Shapley values. This innovation draws inspiration from local surrogate models. Additionally, the introduction of TreeSHAP is noteworthy — an efficient strategy for estimating Shapley values in the context of tree-based models. SHAP also offers DeepSHAP (DeepLIFT + Shapley Values), a high-speed approximation algorithm for SHAP values in DL models that builds on a connection with DeepLIFT described in [40]. The implementation here differs from the original DeepLIFT by using a distribution of background samples instead of a single reference value, and using Shapley equations to linearize components such as max, softmax, products, divisions, etc [41]. The significance of SHAP extends to encompass numerous global interpretability techniques founded on the aggregation of Shapley values. This chapter serves to comprehensively explain both the novel estimation methodologies and the global interpretation approaches.

3.4.1 Definition

The primary objective of SHAP involves explaining the prediction generated for a given instance x through the computation of the individual contributions made by each feature to this prediction. This method of explanation involves the computation of Shapley values from coalitional game theory. In this context, the features of a data instance effectively assume the roles of participants within a coalition. These Shapley values offer insights into how to equitably distribute the prediction amongst the various features. A participant could refer to a solitary feature value, such as in the case of tabular data, or to a group of feature values. This concept finds relevance in scenarios like image explication, wherein pixels can be grouped into superpixels, and the resultant prediction can be appropriately distributed among them. A distinctive aspect introduced by SHAP is the representation of the Shapley value explanation as an additive method for attributing features, effectively akin to a linear model. This point of view establishes a connection between LIME and the concept of Shapley values. The SHAP framework precisely

outlines this mode of explanation as follows:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{3.20}$$

where g is the explanation model, $z' \in \{0,1\}^M$ symbolizes the coalition vector, with M representing the upper limit on coalition size, and $\phi_j \in \mathbb{R}$ signifying the feature attribution for a feature j, known as the Shapley values. The term *coalition vector* in the SHAP paper is referred to as "simplified features". This nomenclature selection seems to originate from its relevance, particularly in scenarios such as image data, where pixel-level representation gives way to aggregate formations like superpixels.

It proves insightful to regard the z's as descriptors of coalitions: Within the coalition vector, an entry with a value of 1 designates the presence of the corresponding feature value, while a value of 0 denotes its absence. The procedure to compute Shapley values involves simulating scenarios in which certain feature values are "present" while others remain "absent." The concept of depicting coalitions as a linear model serves as a strategic approach to compute the ϕ's. When considering an instance of interest, labeled as x, the coalition vector x' becomes an array entirely composed of 1's, implying the presence of all feature values. This leads to the following simplified formulation:

$$g(x') = \phi_0 + \sum_{j=1}^{M} \phi_j \tag{3.21}$$

Prior to delving into the specifics of the estimation procedure, it is essential to explore the properties of the ϕ values.

Exclusive adherence to Efficiency, Symmetry, Dummy, and Additivity criteria is a distinct peculiarity of Shapley values. These identical criteria are also fulfilled by the SHAP methodology, given its computation of Shapley values. Notable differences between the properties asserted by SHAP and the foundational Shapley values are evident in the SHAP paper. Within the SHAP documentation, a trio of desirable properties is outlined:

1) Local Accuracy

Consider the function approximation as follows:

$$\widehat{f}(x) = g(x') = \phi_0 + \sum_{j=1}^{M} \phi_j x'_j \tag{3.22}$$

Should ϕ_0 be designated as $E_X(\widehat{f}(x))$ and assigned a unit value to each x'_j, the resulting outcome aligns with the principle of Shapley efficiency (despite employing an alternative nomenclature and referencing

the coalition vector):

$$\widehat{f}(x) = \phi_0 + \sum_{j=1}^{M} \phi_j x'_j = E_X(\widehat{f}(x)) + \sum_{j=1}^{M} \phi_j \tag{3.23}$$

2) Missingness

$$x'_j = 0 \Rightarrow \phi_j = 0 \tag{3.24}$$

The concept of Missingness dictates that when a feature is missing, its associated attribution is deemed as zero. It is worth noting that x'_j pertains to the coalitions wherein a value of 0 signifies the non-existence of a feature value. Within coalition representation, all feature values x'_j corresponding to the instance under consideration should be set to 1. A 0 in this context would imply the absence of the respective feature value for the instance in question. This property is not among the conventional Shapley values' properties, but in the context of SHAP its necessity is highlighted. Lundberg coined it a "minor book-keeping property". In theory, a missing feature might possess an arbitrary Shapley value without undermining the local accuracy property, as it gets multiplied by $x'_j = 0$. The Missingness property ensures that absent features receive a Shapley value of 0. In practical scenarios, this is only relevant for features that remain constant.

3) Consistency

Consider the function $\widehat{f}_x(z') = \widehat{f}(h_x(z'))$, while z'_{-j} signifies the condition $z'_j = 0$. For any pair of models denoted as f and f' meeting the criteria:

$$\widehat{f'}_x(z') - \widehat{f'}_x(z'_{-j}) \geq \widehat{f}_x(z') - \widehat{f}_x(z'_{-j}) \tag{3.25}$$

for all inputs within the set $z' \in \{0,1\}^M$, then it follows that:

$$\phi_j(\widehat{f'}, x) \geq \phi_j(\widehat{f}, x) \tag{3.26}$$

The principle of consistency says that if a model undergoes alteration such that the marginal contribution of a feature value increases or remains unchanged (regardless of other features), the Shapley value will similarly experience an increase or remain steady. The Shapley properties Linearity, Dummy and Symmetry are consequences of the Consistency principle, as described in the Appendix of Lundberg and Lee [40].

3.4.2 KernelSHAP

KernelSHAP computes the impact of individual feature values on the prediction for a given instance x. The KernelSHAP methodology consists of a series of stages: **1)** Generating sample coalitions denoted as z'_k, where $z'_k \in \{0,1\}^M, k \in \{1,...,K\}$ (where 1 signifies the presence of the feature in the coalition

and 0 denotes absence). **2)** Determining predictions for each z'_k. This involves converting z'_k into the original feature space and subsequently employing the model $\widehat{f} : \widehat{f}(h_x(z'_k))$ **3)** Evaluating the significance (weight) of each z'_k using the SHAP kernel. **4)** Constructing a weighted linear model. **5)** Returning the Shapley values ϕ_k, derived from the coefficients of the linear model.

A novel approach involves the generation of diverse coalitions using a series of iterative coin tosses until a sequence of alternating 0's and 1's emerges. To illustrate, consider the vector $(1, 0, 1, 0)$, denoting the inclusion of the first and third features in a coalition. The dataset for the regression model is then formed by sampling K such coalitions. The objective of the regression model is to predict outcomes for coalitions. However, it is important to note that the model remains untrained on these binary coalition patterns, rendering it incapable of generating predictions. To transition from sets of feature value coalitions to valid data instances, a function denoted as $h_x(z') = z$ is essential, where $h_x : \{0, 1\}^M \rightarrow \mathbb{R}^p$. Within this function, 1's are linked to corresponding values from a target instance x, which necessitates explanation. Meanwhile, in the context of tabular data, 0's are mapped to values from a different instance randomly drawn from the dataset. This equivalence signifies that the absence of a feature value is synonymous with its substitution by an arbitrary value sourced from the data. A visualization in the context of tabular data can be observed in the following figure (Figure 3.1), which explains the transformation from coalitions to feature values. The function denoted as h_x corresponds to the process of assigning a coalition to a valid instance. In the context of the current features (1), h_x operates by associating with the feature values of x. In situations where features are missing (0), h_x is responsible for linking to the values of an arbitrarily selected data instance.

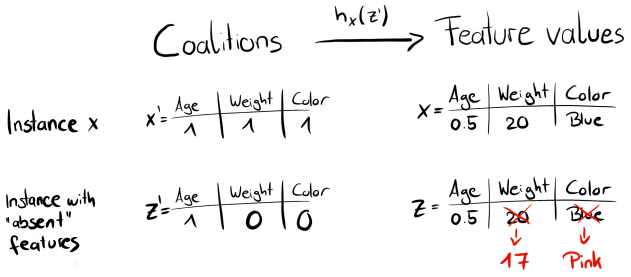

Figure 3.1: Example of a tabular data explanation of the transformation from coalitions to feature values. Source: [3]

In the context of tabular data analysis, the treatment of feature X_j and X_{-j} (referring to the other features) by h_x is conducted as if they are independent entities. This involves the incorporation of the marginal distribution through the following expression:

$$\widehat{f}(h_x(z')) = E_{X_{-j}}[\widehat{f}(x)] \tag{3.27}$$

During the process of drawing samples from the marginal distribution, the dependence structure be-

tween present and absent features is disregarded. Consequently, KernelSHAP encounters a parallel limitation shared with various methods rooted in permutation-based interpretation. This limitation is associated with the overemphasis on improbable instances during estimation, leading to potential unreliability in outcomes. It is imperative, however, to access the marginal distribution through sampling. To address this, a strategic shift towards sampling from the conditional distribution is advocated, introducing alterations to the value function and thereby transforming the underlying framework to which Shapley values are the solution. Consequently, the interpretation of Shapley values undergoes a transformation; for instance, a feature that the model may not have used could attain a non-zero Shapley value when operating under the influence of conditional sampling. In contrast, under the framework of the marginal game, the Shapley value assigned to such a feature would consistently be 0, as any other scenario would infringe the Dummy axiom.

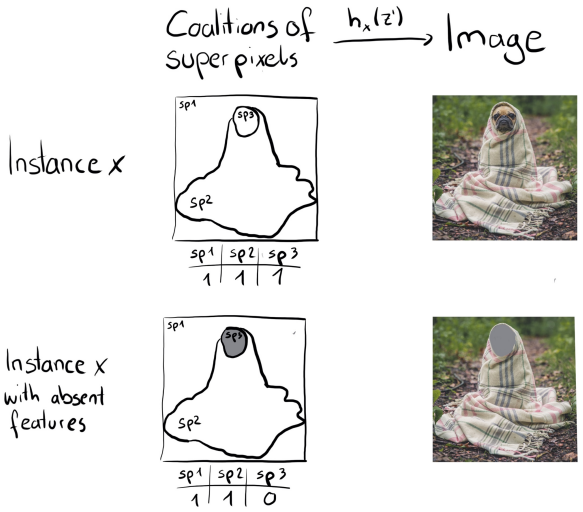

Figure 3.2: Example of an image-based data potential mapping function. Source: [3]

For image-based data, a visual representation of a potential mapping function is depicted in Figure 3.2. The function denoted as h_x performs the task of associating coalitions of superpixel groupings (referred to as sp) with images. These superpixels represent clusters of pixels within the image. In cases involving active features (designated as 1), function h_x yields the relevant segment from the initial image. Conversely, in situations involving inactive features (marked as 0), h_x renders the respective region in shades of gray. Alternatively, a plausible approach could involve setting the region to an average color derived from neighboring pixels or employing analogous strategies.

The primary distinction from LIME lies in how instances are assigned weights within the regression model. While LIME bases its weighting on proximity to the original instance, with more 0's in the coalition vector resulting in smaller weights, SHAP takes a different approach. SHAP assigns weights to sam-

pled instances according to the weight the coalition would receive in Shapley value estimation. In this scheme, smaller coalitions (those with few 1's) and larger coalitions (those with many 1's) are assigned the highest weights. This weighting strategy is rooted in the idea that the most insights into individual features are gained when it is possible to analyze their effects in isolation. If a coalition comprises a single feature, the examination of its isolated primary effect on the prediction can be done. Conversely, if a coalition includes all but one feature, the total effect of that feature is understandable by encompassing both its primary effect and feature interactions. However, when a coalition consists of half the features, the ability to discern the contribution of individual features diminishes, as there exist numerous potential coalitions with half of the features. To establish a weighting method compliant with the Shapley value concept, Lundberg et al. introduced the SHAP kernel:

$$\pi_x(z') = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)} \tag{3.28}$$

In this context, M represents the upper limit for coalition size, while $|z'|$ denotes the count of active features within instance z'. Lundberg and Lee have illustrated that employing linear regression in conjunction with this kernel weighting mechanism produces Shapley values. If one were to employ the SHAP kernel with LIME on the coalition data, LIME would also be capable of estimating Shapley values.

A more strategic approach to coalition sampling can be adopted. The allocation of sampling effort should be more discerning, favoring the smallest and largest coalitions, which collectively carry the bulk of the significance, taking up the majority of the weight. To enhance the accuracy of Shapley value estimation, it can be allocated a portion of the sampling budget K towards including these high-impact coalitions, as opposed to random sampling. The initial step involves considering all feasible coalitions comprising 1 and $M-1$ features, resulting in a total of $2M$ coalitions. If there is sufficient budget remaining (the current budget being $K-2M$), coalitions featuring 2 features and those with $M-2$ features can be included, and so forth. For the remaining coalition sizes, it is conducted sampling with adjusted weighting.

The essential components for constructing the weighted linear regression model are the data, the target variables, and the weighting parameters:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{3.29}$$

The linear model denoted as g is trained through the optimization process applied to the following loss function L:

$$L(\widehat{f}, g, \pi_x) = \sum_{z' \in Z} [\widehat{f}(h_x(z')) - g(z')]^2 \pi_x(z') \tag{3.30}$$

where the training data is denoted as Z. It is possible to encounter the familiar sum of squared errors, a

traditional optimization target for linear models. The model's estimated coefficients, represented as ϕ_j, correspond to Shapley values.

3.4.3 TreeSHAP

Lundberg et al. in their 2018 work [42] introduced TreeSHAP, a modified version of SHAP tailored for tree-based ML models like decision trees, random forests, and gradient boosted trees. TreeSHAP was proposed as a fast, model-specific alternative to KernelSHAP. However, it was discovered that TreeSHAP could yield counterintuitive feature attributions.

TreeSHAP's approach involves defining the value function through conditional expectation, denoted as $E_{X_j|X_{-_j}}(\widehat{f}(x)|x_j)$, instead of employing the marginal expectation. The challenge with the conditional expectation arises when features that have no influence on the prediction function, denoted as f, receive a nonzero TreeSHAP estimation. This phenomenon was demonstrated by Sundararajan et al. in their 2019 study [43] and further elaborated upon by Janzing et al. in the same year [44]. This nonzero estimate can occur when the feature in question exhibits correlation with another feature that indeed impacts the prediction.

In terms of computational efficiency, TreeSHAP significantly outperforms KernelSHAP. It accomplishes this by reducing the computational complexity from $O(TL2^M)$ to $O(TLD^2)$, where T stands for the number of trees, L represents the maximum number of leaves within any tree, and D signifies the maximum depth of any tree.

TreeSHAP relies on the conditional expectation $E_{X_j|X_{-_j}}(\widehat{f}(x)|x_j)$ for estimating effects. To provide an intuitive understanding of how the expected prediction for a single tree, an instance x, and a feature subset S can be computed: if one were to condition on all features, assuming S comprises all features, then the prediction from the node where instance x resides would constitute the expected prediction. Conversely, if the prediction was not conditioned on any feature (i.e., S is empty), the weighted average of predictions from all terminal nodes would be used. When S contains some, but not all, features, predictions from unreachable nodes are disregarded. Here, "unreachable" signifies that the decision path leading to these nodes conflicts with the values in x_S. From the remaining terminal nodes, predictions are averaged with weights based on node sizes (i.e., the number of training samples in each node). The mean of these remaining terminal nodes, weighted by the number of instances per node, yields the expected prediction for x given S. The challenge arises from the need to apply this procedure for every possible subset S of feature values. TreeSHAP employs a polynomial time computation instead of an exponential one. The fundamental concept is to simultaneously traverse all possible subsets S down the tree. For each decision node, it is essential to keep track of the number of subsets, which depends on the subsets in the parent node and the feature used for the split. For instance, if the initial split in a tree is on feature x_3, all subsets containing feature x_3 will converge to one node (the one linked to x).

Subsets lacking feature x_3 will distribute to both nodes but with reduced weight. It is worth noting that subsets of differing sizes carry different weights, which adds complexity to the algorithm since the algorithm has to keep track of the overall weight of the subsets in each node. Expanding this computation to encompass multiple trees is feasible due to the Additivity property of Shapley values. Consequently, the Shapley values of a tree ensemble constitute the (weighted) average of the Shapley values of the individual trees.

3.4.4 DeepSHAP

DeepSHAP is a local model-specific interpretability method designed to approximate SHAP values for DL models. It represents an enhanced version of the DeepLIFT algorithm where, in a manner akin to KernelSHAP, it is employed an approximation technique for estimating the conditional expectations of SHAP values, utilizing a selection of background samples. This combination of Shapley values and DeepLIFT allows DeepSHAP to take advantage of the extra knowledge about the compositional nature of deep networks to improve computational performance. DeepSHAP combines SHAP values computed for smaller components of the network into SHAP values for the whole network by recursively passing DeepLIFT's multipliers (which are basically SHAP values calculated for a node) backward through the network [4]. By propagating SHAP values from the output layer back to the input layer, DeepSHAP offers a clear and intuitive understanding of each feature's contribution to the model's output. It adapts the SHAP concept for DL models by recursively calculating SHAP values for each neuron, starting from the output layer and considering the neuron's activation function and contribution to the model's output. Notably, in their work [40], Lundberg and Lee demonstrated that DeepLIFT's per node attribution rules (Shrikumar, Greenside, and Kundaje, arXiv 2017) [37] can be tailored to closely approximate Shapley values. The process involves integrating over numerous background samples, allowing the Deep estimates to closely approximate SHAP values. These estimates are structured in such a way that they sum up to the difference between the expected model output on the provided background samples and the current model output $(f(x) - E[f(x)])$ [45] [46]. DeepSHAP ultimately offers transparent explanations for complex DL models while upholding the consistency and fairness principles associated with SHAP values [47].

3.4.5 Advantages

As SHAP performs the computation of Shapley values, it inherits all the merits associated with this concept. SHAP is grounded in a robust theoretical framework rooted in game theory, ensuring its credibility. It ensures equitable allocation of predictive influence across feature values, leading to contrastive explanations that compare predictions against the mean prediction.

Additionally, SHAP bridges the domains of LIME and Shapley values, offering useful insights into both techniques while fostering the consolidation of the interpretable ML field.

Furthermore, SHAP has a fast implementation tailored for tree-based models, a pivotal factor in its widespread acceptance. This expeditious computation capability avoids the primary obstacle to Shapley values adoption — slow computational processes. This computational efficiency enables the calculation of numerous Shapley values necessary for comprehensive global model interpretations. These global insights encompass feature importance, feature dependence, interactions, clustering, and summary plots. Importantly, SHAP ensures the coherence of global interpretations with local explanations, given that Shapley values serve as the "atomic unit" for global interpretations. In contrast, utilizing LIME for local explanations and relying on partial dependence plots and permutation feature importance for global insights lacks a unifying foundation.

3.4.6 Disadvantages

KernelSHAP exhibits a notably reduced computational speed, is slow, rendering its practicality questionable when tasked with the computation of Shapley values across numerous instances. This limitation extends to all global SHAP methodologies, which necessitate the calculation of Shapley values for an extensive array of instances.

A noteworthy aspect of KernelSHAP is its neglect of feature dependence. This issue is not unique to KernelSHAP but is encountered in several other permutation-based interpretation techniques as well. The common approach of substituting feature values with those from random instances is generally viable due to the ease of random sampling from the marginal distribution. However, when features are correlated or dependent, this approach leads to putting excessive importance to improbable data points. An effective resolution to this challenge is offered by TreeSHAP, which explicitly models the conditional expected prediction.

Nonetheless, it is important to acknowledge that TreeSHAP can generate feature attributions that may appear counterintuitive. This outcome is a result of TreeSHAP's approach to addressing extrapolation challenges by modifying the value function, thereby introducing slight alterations to the interpretation process. Consequently, features that employ no recognizable influence on predictions can be assigned non-zero TreeSHAP values.

It is worth noting that the limitations associated with Shapley values are equally applicable to SHAP. Shapley values can be misinterpreted, and their computation necessitates access to the underlying data, with TreeSHAP being a notable exception.

4

Methodology

Contents

This chapter presents in a more technical way the available infrastructure of both post hoc explanation methods, LIME and SHAP, for the purpose of this work, previously stated in Chapter 3, therefore involving the utilization of MTS data in a DL model for regression. It is going to be introduced the functions and visualization tools utilized, their purpose and how they were implemented. Based on the knowledge acquired in Chapter 3, these methods are reliable and efficient, capable of offering explanations that pinpoint the key features influencing the predictions made by the model.

4.1 LIME

The LIME library provides a special explainer tailored for keras-style recurrent neural networks (RNNs) called RecurrentTabularExplainer. This explainer is a Python class that extends the core LimeTabularExplainer, which is specifically dedicated for two-dimensional tabular datasets. RecurrentTabularExplainer enables to work with input data in the form of [samples, time steps, features]. This is achieved through reshaping training data and feature labels, transforming them into a representation like $(val1_{t1}, val1_{t2}, val1_{t3}, ..., val2_{t1}, ..., valn_{tn})$, where $val1_{t1}$ stands for the value of the initial feature at the first time step of the given sample [48]. This explainer operates in two modes, regression and classification, but only the first one is useful for this Thesis. So, the regression mode serves as the foundational framework for model interpretation. Although the LIME library allows data visualization, its capabilities in this regard are very limited when compared to SHAP. An essential method of the Explainer object, *show_in_notebook()*, elucidates the model's decision-making process for specific predictions by highlighting feature contributions. For a concrete visualization example, refer to Figure 4.1.

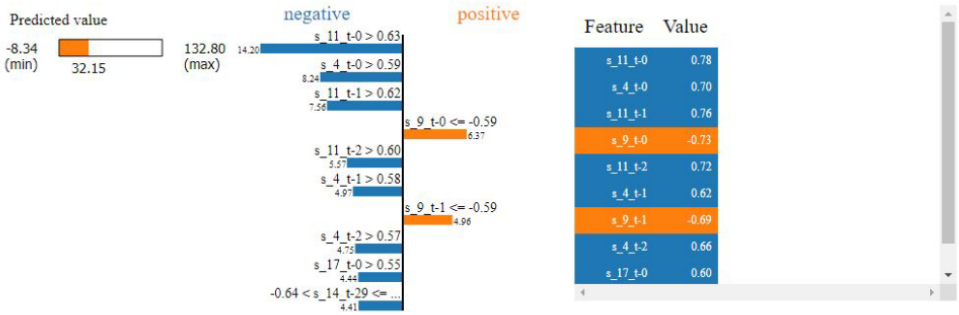

Figure 4.1: Example of an explanation generated by LIME for predicting RUL. Source: [4]

Within the visualization, there are three distinct components: a dynamic progress bar, a bar chart, and a data summary. The progress bar visually demonstrates the range within which the values fluctuate, alongside the actual predictions. Meanwhile, the bar chart employs a vertical axis to denote the various features and a horizontal axis to indicate their respective forces. Positive values, represented in an orange hue, signify features that increase the value of the prediction, whereas negative values, depicted

in blue, signify features that diminish the prediction value. The data summary, presented in tabular form, comprises the actual values of the top N data points. The specific value of N, which determines the number of features displayed, is adjustable. In this example, the RUL value predicted by the model is 32.15, and the range in which it varies is between -8.34 and 132.8. The values of the sensors on the right side indicate that blue is the main color, and the degrading performance of sensors $s11$ and $s4$ are detected for several time stamps. These sensors can be indicated as the principal factors in predicting the low RUL value.

Although tested in this work, this explanation method was considered less complete than SHAP, which led to the utilization of SHAP only.

4.2 SHAP

In Chapter 3, three methods based on SHAP were detailed, but only one of them proves applicable for interpreting the DL model for the MTS prediction. TreeSHAP is only tailored for tree-based ML models. KernelSHAP explainer can not be computationally configured for a GRU model. This is due to its requirement of one- or two-dimensional data inputs to be passed through the prediction model, such as [samples, features], whereas a GRU network necessitate data in a specific matrix format: [samples, time steps, features]. Here, the first dimension signifies samples, the time steps dimension delineates every step for each sample, and features denote the attributes for each time step.

Consequently, the choice gravitated towards DeepSHAP as the explanatory tool. Implemented through the DeepExplainer from the SHAP package [41], this method demands an initial background dataset. This dataset serves as the foundation for approximating the Shapley values and informs the integration process. A detailed description of the implementation of SHAP Deep Explainer, also stating the passed and generated data, is pointed out in Section 5.3.2.

Beyond merely calculating SHAP values, SHAP library also facilitate visualizations, enabling to discern pivotal relationships within the model. In doing so, it is possible to gain a better understanding of how these models work. Subsequent sections delve into the insights derived from the generated plots.

As an example, considering a trained random forest classifier comprising 100 trees for the task of predicting cervical cancer risk to explain individual predictions SHAP will be used. Given that a random forest constitutes an ensemble of decision trees, the expeditious TreeSHAP estimation method over the more time-consuming KernelSHAP method is the better option. It is noteworthy that this instance deviates from the conventional approach by employing the marginal distribution instead of the conditional distribution, a detail described in the package documentation but not found in the original research paper. By adhering to the marginal distribution in this context, the interpretation aligns with the principles elucidated in Section 3.3 [3].

4.2.1 Force Plot

Here, feature attributions, such as Shapley values, are portrayed as 'forces'. Each feature value is akin to a force that can either augment or diminish the prediction. The prediction starts from a base value $E[f(x)]$, which, in the case of Shapley values, corresponds to the mean of all output values of the model on the training data and it represents the predicted outcome if none of the features were used [4]. In the graphical representation, each Shapley value is symbolized as a bar, exerting either a positive (increasing) or negative (decreasing) influence on the prediction. Each bar corresponds to a feature's importance value, the color of the bar indicates its effect on the output (red is positive and blue is negative) and its size relates to the magnitude of that effect. These forces effectively balance each other to attain the final prediction for the specific data instance.

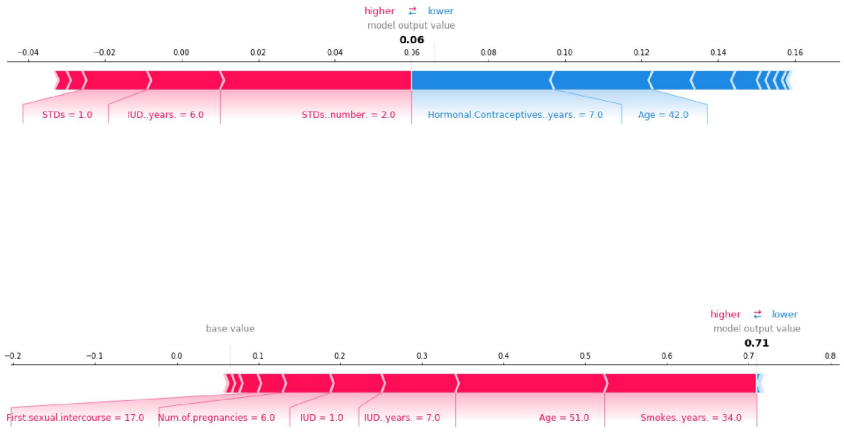

Figure 4.2: Example of SHAP force plots explanations of predicted cancer probabilities for two individuals from a cervical cancer dataset. Source: [3]

In Figure 4.2, it is possible to see force plots visualizing SHAP explanations of predicted cancer probabilities for two individuals from a cervical cancer dataset. The reference point, denoting the mean predicted probability, stands at 0.066. The first female displays a diminished predicted risk of 0.06. Factors associated with elevated risk, like STDs, are offset by mitigating elements, such as age. On the other hand, the second female exhibits an elevated predicted risk of 0.71. Her 51 years of age and 34 years of smoking contribute to the escalation in her predicted cancer susceptibility. These explanations pertained to the predictions for each data point, individual predictions. Shapley values possess the potential for aggregation into global explanations. Executing the SHAP method on every individual case will generate a Shapley value matrix. This matrix encompasses a row for each specific data point and a column for each

feature. The interpretation of the entire model can be made by examining the Shapley values within this matrix [3].

The previous example utilized the SHAP force plot using two-dimensional data. When applied to three-dimensional data, SHAP generates a distinctive force plot, as illustrated in Figure 4.3. In this scenario, each time step is graphed along the x-axis, while the corresponding forces are represented on the y-axis.

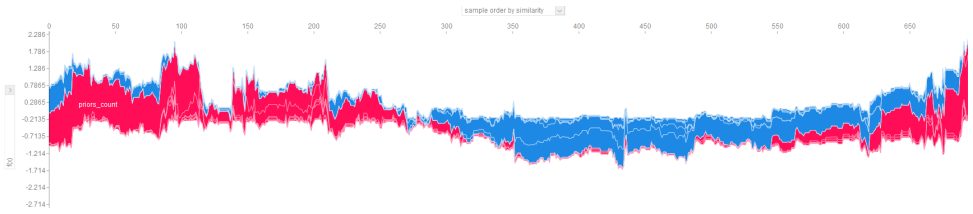

Figure 4.3: Example of the SHAP force plot with three-dimensional data where the horizontal axis depicts the time steps for the previously trained random forest model used in predicting cervical cancer. Source: [5]

4.2.2 Feature Importance

The concept behind SHAP feature importance is straightforward: Features exhibiting substantial Shapley values in absolute terms carry importance. To obtain a global understanding of importance, it must be calculated the mean absolute Shapley values per feature across the dataset using the formula:

$$I_j = \frac{1}{n} \sum_{i=1}^{n} |\phi_j^{(i)}| \tag{4.1}$$

Subsequently, the features are sorted in descending order of significance and plotted.

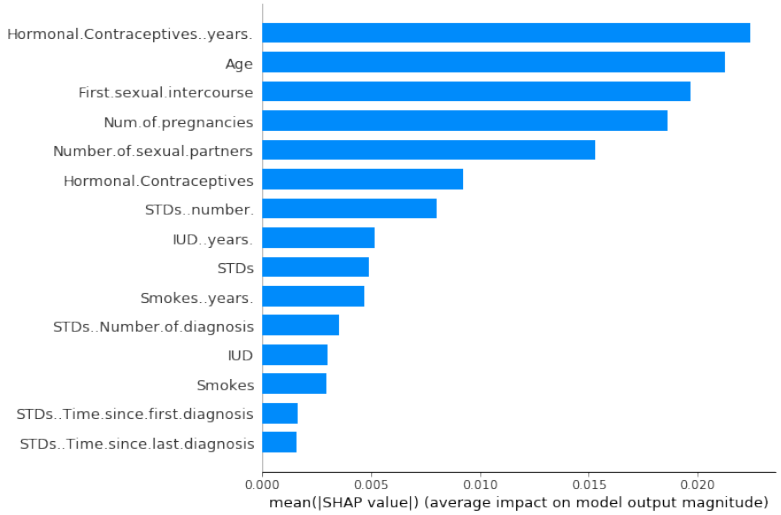

Figure 4.4: Example of the SHAP feature importance plot for the previously trained random forest model used in predicting cervical cancer. Source: [3]

Figure 4.4 illustrates the SHAP feature importance for the previously trained random forest model used in predicting cervical cancer. SHAP feature importance, quantified as the average absolute Shapley values, revealed that the duration of hormonal contraceptive use emerged as the dominant factor, as the most important feature, resulting in an average shift of 2.4 percentage points (0.024 on the x-axis) in the predicted absolute cancer probability.

While the feature importance visualization serves its purpose, it lacks any additional insights beyond the importance values. To generate a more comprehensive and informative visualization, the next step involves examining the summary plot.

4.2.3 Summary Plot

The synthesis of feature importance and their effects is encapsulated within the summary plot. Every data point within this visual representation corresponds to a Shapley value, uniquely characterizing both a feature and an instance. The vertical positioning is dictated by the feature, while the Shapley value defines the horizontal axis. The color gradient mirrors the feature's value, ranging from minimal to maximal. In cases of overlap, data points are subtly displaced along the y-axis, revealing the dispersion of Shapley values per feature. The order of the features is governed by their respective degrees of importance.

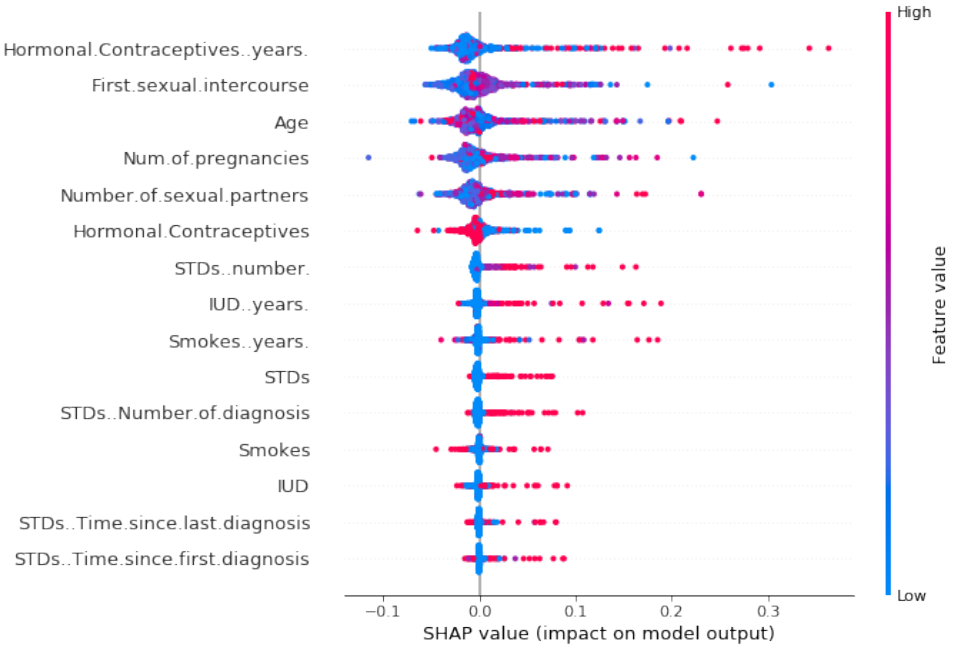

Figure 4.5: Example of the SHAP summary plot for the previously trained random forest model used in predicting cervical cancer. Source: [3]

Observing Figure 4.5, SHAP summary plot reveals that a shorter duration of hormonal contraceptive

use is associated with a decreased predicted cancer risk, while an extended history of usage is linked to an elevated risk. It is important to emphasize that all observed effects merely characterize the model's behavior and do not inherently imply causation in real-world scenarios.

4.2.4 Waterfall Plot

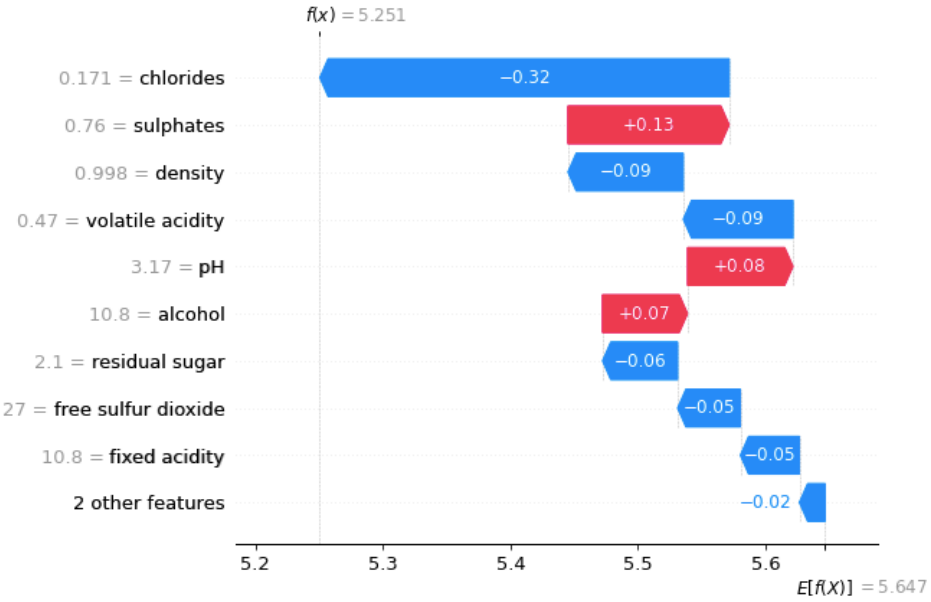

Figure 4.6: Example of a SHAP waterfall plot. Source: [6]

The graphical representation shown in Figure 4.6, known as the SHAP Waterfall plot, offers an alternative way to illustrate SHAP values, and they are designed to display explanations for individual predictions [49]. It conveys comparable insights to a force plot, specifically for a singular time step. These SHAP values represent the extent of influence each factor exerted on the model's prediction relative to the average prediction [4]. The bottom of the plot starts as the expected value of the model output ($E[f(x)]$), and then each row shows how the positive (red) or negative (blue) contribution of each feature moves the value from the expected model output over the background dataset to the model output for this prediction [49]. Notably large positive or negative SHAP values denote substantial influence on the model's prediction.

5

Experimental Setup

Contents

This section delves into the analysis of the experimental setup, describing the data and model used and their implementation, as well as the implementation of DeepSHAP. It further explores two proposed experiments that consisted in providing reduced input data during inference, with the purpose of potentially validating the results obtained from applying DeepSHAP in the black box model.

5.1 Caenorhabditis Elegans

The study of the human brain and its intricate nervous system presents an intellectually demanding and stimulating task, and is definitely one of the greatest challenges in computational neuroscience. One of the primary difficulties stems from the remarkable versatility of the human brain, capable of performing a myriad of functions, coupled with its astonishing complexity. With approximately 8.61×10^{10} neurons [50], it stands as one of the largest complex systems available for study. This complexity becomes even more evident when comparing the number of neurons in the human nervous system to other living species' nervous systems, like the hermaphrodite *Caenorhabditis Elegans*, a roundworm of about 1 mm in length that has 302 neurons [51].

The *Caenorhabditis Elegans*, or *C. elegans* for short, has the potential to be valuable in acquiring a deeper understanding of the foundational aspects of neural dynamics and overall brain organization. In order to create an approach to model peripheral input-output behavior of complex nervous systems, Mestre, in his work from 2021 [52], tested ML techniques on their ability to construct data-driven black box models of peripheral input-output behavior of this nematode's nervous system, and also underwent evaluation for their precision in replicating system behavior. One of the ML techniques that performed better was the GRU, and was chosen to be used in this work.

In summary, due to the complexity of the task of modeling the human nervous system, the *C. elegans* was picked for this thesis because of its simplicity, commonly used as a benchmark in computational neuroscience, and for which the connectome is completely mapped, illustrating all of the neuronal connections in the nervous system. The model utilized in [53] further facilitated the generation of responses in the roundworm's nervous system under diverse stimuli.

5.2 Forward Crawling Motion

This section delves into the presentation of the data used in this work, discussing the model employed in its generation, a work realized by Barbulescu and Silveira [53]. Notably, the model and data used in this Thesis align with those in a prior work by Barbulescu, Mestre, and Silveira [54], and the data utilized in this Thesis was constructed by Mestre in a work from 2021 [52]. Follows a description of the way it was built and the specific components of the nervous system relevant to the Forward Crawling Motion (FCM)

behavior under scrutiny.

To ensure the efficacy of training and evaluating the ML model utilized in this thesis, it was essential to find a dataset detailing the peripheral input-output behavior of the *C. elegans*. Leveraging the accessibility of 3D neuron reconstructions and computational models for this nematode, synthetic simulated data emerged through the deployment of the NEURON simulator [55] and the same model used in previous research [54]. To evaluate the model's performance in replicating a specific behavior of the *C. elegans*, a dataset mirroring the FCM scenario was generated. This dataset underwent partitioning into three distinct segments: one for training the ML model, another for model validation, hyperparameter tuning, and assessing the influence of time step variations on the model, and a third set to evaluate the performance of the ML model.

This behavior impacts more than a hundred neurons within the *C. elegans* nervous system, predominantly motor neurons [54]. However, to mitigate computational complexities, this thesis focuses on a subset of neurons, a choice also applied in [52]. To balance the need for accuracy with computational efficiency, a sample comprising 16 neurons was chosen: AS1, AS7, DA1, DA9, DB1, DB5, DD1, DVA, LUAL, PHCL, PLML, PVCL, PVR, SIBVL, VB1, and VD1. These were selected due to their robust responses during the observed behavior.

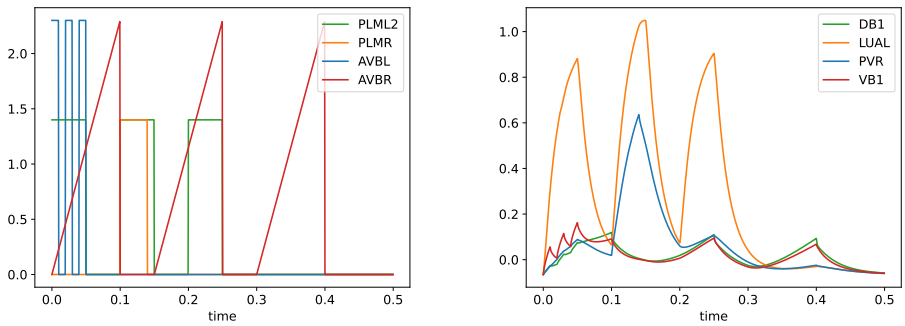

Figure 5.1: Example of inputs (left) and outputs (right) for a test sequence of **FCM-20**.

To optimize the hyperparameters of the model, a foundational dataset, denoted as **FCM-20**, was created to explore the nematode's behavior. This dataset allocated $50\,\%$ of its data to the training set and $25\,\%$ each to the validation and test sets. Comprising 40 distinct sequences, each sampled at intervals of $0.5\,\text{ms}$, the dataset was generated through a diverse range of stimuli, and the sequences were randomly divided across datasets. The stimuli employed were a mix of random pulses and ramps, serving as inputs for the PLM and AVB neurons within the *C. elegans* nervous system. The inputs used for the PLM neurons were capped at $1.4\,\text{nA}$, while AVB neurons received inputs not exceeding $2.3\,\text{nA}$. To minimize computational expenses during testing, only a subset of output neuron data was utilized in [52], and in this work, more specifically the data from neurons DB1, LUAL, PVR, and VB1. Figure 5.1

displays a sample sequence alongside its corresponding inputs and outputs.

5.3 Implementation Settings

5.3.1 Model

In this section, a description of how the ML model was implemented is provided.

The implementation of the GRU was done using python [56] and its libraries like keras [57], since they provide an easy framework for creating models with multiple layers of different kinds. The GRU model is composed by two different layers, the main recurrent layer, which the model is named after, and a small dense layer at the end, which converts the output of the recurrent layer to the output of the model. Due to the model output requirements posed by the explanation methods some changes had to be made to the original GRU code. As stated in Section 4.2, a GRU network needs data in a specific matrix format, more specifically [samples, time steps, features], and for that reason SHAP DeepExplainer was picked as the explanatory tool. However, to be able to use it, the model output has to be one- or two-dimensional, and so, a *Flatten* layer had to be added to the previous 3D model output in order to reduce the dimensions of the output. Consequently, instead of sticking to the initial output using 4 features (Figure 5.1), it was decided to create 4 models, each with 1 feature of output, as can be visualized in Figure 5.2.

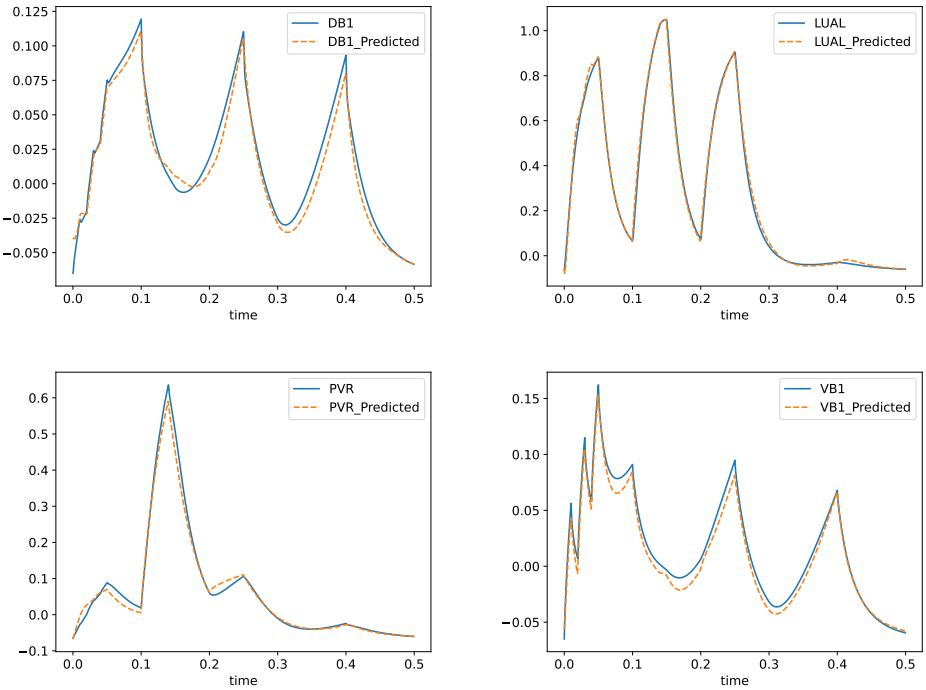

Figure 5.2: Example of outputs predicted for features DB1, LUAL, PVR and VB1 for a test sequence of **FCM-20**.

For the training procedure, the model was trained for 1000 epochs with the optimization method Adam [58]. For model evaluation, Mean Squared Error (MSE) was the chosen regression metric. MSE computes the mean of the squared difference between each pair of corresponding data points from the two sets, using Equation (5.1). Therefore, the difference between a pair of corresponding data points is inflated when this difference is larger and it is lowered even more when the difference is already small. This detail allows the models that use the MSE as the loss function during the training procedure to improve on data points where the error is large, while also giving a more exact information on the correctness of the results, since larger errors get more punished and lower errors less punished.

For the evaluation of the results after model training, Root Mean Squared Error (RMSE), calculated as in Equation (5.2), is used, since the errors are compared also in terms of their magnitude and usage of values with the same units as the ones of the data being predicted allows for an easier comparison.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \tag{5.1}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2} \tag{5.2}$$

In Equation (5.1) and Equation (5.2), N refers to the number of samples in the sequence predicted by the model, y_i refers to the expected results at time step i, and $\hat{y_i}$ refers to the results predicted by the model at time step i.

Regarding hyperparameters optimization, it was made using **FCM-20**, where 4 input neurons are stimulated while the output voltage of other 4 neurons is measured. There are 3 hyperparameters to be optimized: the batch size that controls the number of sequences the model forward propagates before updating the parameters, the learning rate that defines how much the network changes its parameters on each update, and the size of the hidden layer. The latter is the last one being optimized, aiming to understand how the RMSE changes with the different sizes experimented for the recurrent layer and also how small of a model can be used without loosing prediction capability. From the two other parameters, it is chosen to optimize first the learning rate, fixing a size for the hidden layer of 4, while fixing the batch size as 32. In order to find the learning rate that best suits the model and data being used, different values were experimented: 0.05, 0.01, 0.005 and 0.001. The value of the learning rate for which the model performed better was 0.01, and so it was the chosen learning rate for the Adam optimizer. Regarding the batch size, due to its cost, it was chosen a batch size of 32 in order to maintain a low computational cost. Since the objective is to have a model as reduced as possible relatively to the number of parameters to train, the optimal choice was using 4 as the size of the recurrent layer, totaling 125 parameters. The picked values lead to a good performance of the GRU model to accurately reproduce the behavior of the *C. elegans* nervous system. An example of the validation set of the outputs obtained for the GRU model

can be found in Figure 5.3.

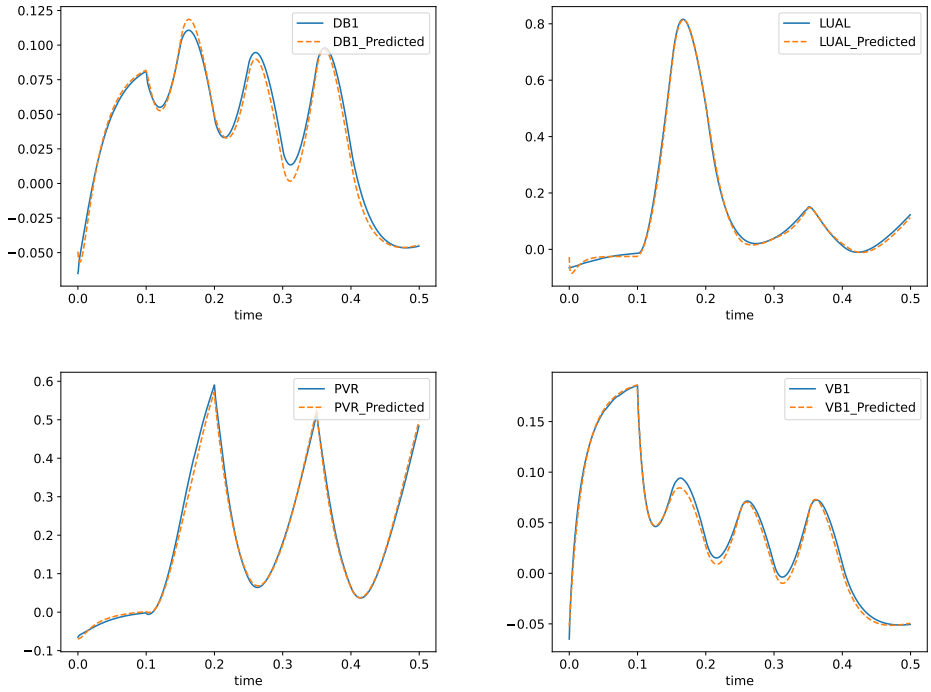

Figure 5.3: True outputs and predicted outputs for one of the examples of the validation set of **FCM-20**.

5.3.2 DeepSHAP

As stated previously in Section 4.2, the selected explanatory tool was DeepSHAP, which is implemented through the DeepExplainer from the SHAP package. This method demands an initial background dataset, which are just data samples from the training set, that serves as the foundation for approximating the Shapley values and informs the integration process. By integrating over many training samples, the Deep estimates approximate SHAP values such that they sum up to the difference between the expected model output on the passed training samples and the current model output $(f(x) - E[f(x)])$ [45, 46].

Regarding the selected data samples from the training set used in this work, there are some important thoughts needed to be discussed. The complexity of the method scales linearly with the number of data samples. Passing the entire training dataset gives very accurate expected values, but can be unreasonably expensive. The variance of the expectation estimates scale by roughly $\frac{1}{\sqrt{N}}$ for N data samples. Even though computationally time expensive, it was decided to try to obtain a very good estimate of the expected values, and so the entire training dataset, which has shape $(20, 1000, 4)$, is passed to the DeepExplainer. For comparison reasons, a dataset composed only by a single row of the training data, returning shape $(1, 1000, 4)$, was also computed.

The expected value, in the context of SHAP, is the average prediction made by the model when it sees a typical or representative dataset. This value is calculated based on the predictions made by the model on the data samples provided to the explainer. These data samples from the training set serve as a reference point to establish what the model typically predicts in the absence of any specific features being present or absent. Internally, the explainer feeds each data point from the data samples through the DL model. It calculates predictions for each data point using the wrapped model, which is a modified version of the model that allows for input perturbations and recording of the model's behavior for these perturbed inputs. After obtaining predictions for all data points in the data samples, the explainer calculates the average (expected) prediction across these data points. The expected value provides a baseline for understanding the impact of individual features on a prediction. When the SHAP values are positive for a feature for a specific prediction, it means that the feature's presence increases the prediction compared to the expected value. Conversely, if the SHAP values are negative for a feature for a specific prediction, it means that the feature's presence decreases the prediction compared to the expected value. The expected value in this thesis has shape (1000,), each data point corresponding to each time step.

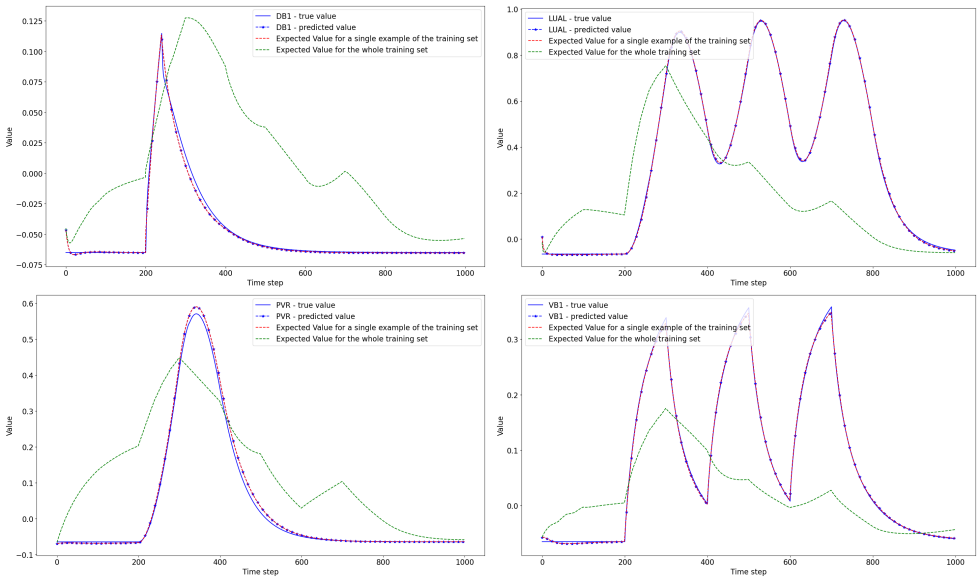

Figure 5.4: Comparison of Expected Value for a single example of the training set, Expected Value for the whole training set, predicted value for a specific example of the training data (the one passed to the Expected Value) and corresponding true input value, for each output neuron.

Observing Figure 5.4, which is an example of an expected value based on a single example of the training data and another one based on the whole training data, for each of the output neurons and additionally plotting its true and predicted values, some thoughts can be explored. Two conclusions one can come to are that the expected values are different, and that the expected value using a single example of the training data mimicks the model prediction for the same specific row of the training set,

that is also being used as the data samples passed to the explainer. So, this would have been a good choice of expected value if the goal was to explain that single example of data, but since the objective is to gain a general knowledge and explain every example of the dataset, a better idea might be to use more data samples from more than one single example of the training set. If instead other examples of the training data are plotted, as can be observed in Figure 5.5, different conclusions can be drawn. This means that, for the remaining examples of the training set, the expected value for a single example of the training set and the SHAP values calculated with it, will not be the best in trying to explain the remaining data. Consequently, the explainer for the whole training set, which obviously takes more data into account than the one for a single example of the training set, is chosen to be used in this work, since it will explain better the data overall.

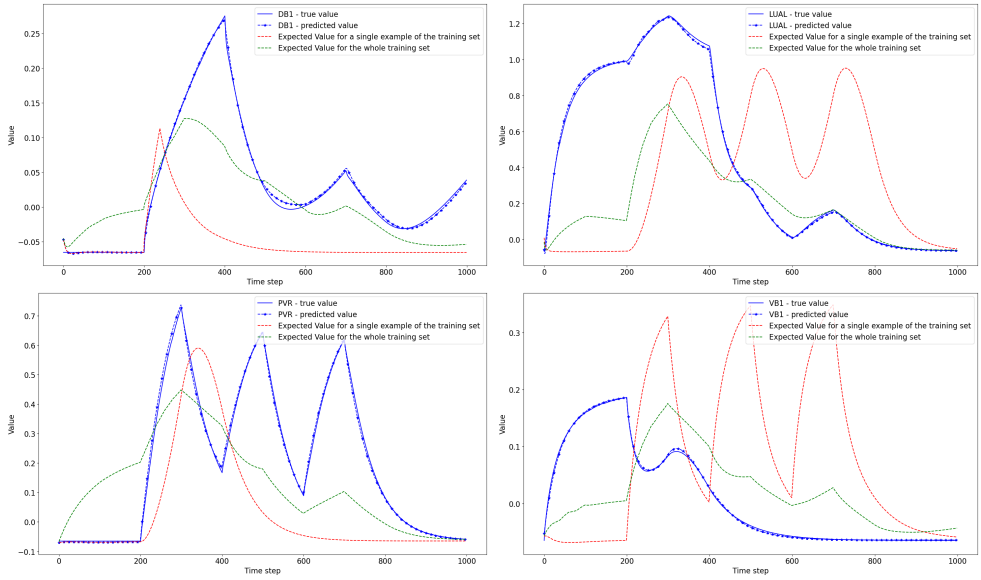

Figure 5.5: Comparison of Expected Value for a single example of the training set, Expected Value for the whole training set, predicted value of a random example of the training set and corresponding true input value, for each output neuron.

Concerning the SHAP values function, it returns approximate SHAP values for the model applied to the test data or unseen data, which are calculated through the explainer created earlier. In this work, the entire test data, which has shape $(10, 1000, 4)$, was used to calculate the SHAP values, returning shape $(1000, 10, 1000, 4)$. This way, it was possible to explain all of the predictions needed using SHAP values. It is noteworthy that, although the whole test data is passed for the calculation of the SHAP values, they will explain each of the 10 examples of the test set separately, and not all at the same time. SHAP values are explored in depth in Section 6.1.

5.4 Experiments

5.4.1 Experiment without 1 neuron

It is often the case that even though all inputs to a system are by definition external to the system, one might not be able to measure or observe all of these outside stimuli given to the system. Therefore, this experiment tests the ability of the models to accurately learn the behavior of the system using less inputs than the ones actually given. To try and understand which neurons had the most importance for the model an experiment divided in 4 cases was made. It essentially consisted in modifying the test part of the dataset, more concretely the input one, zeroing the data corresponding to one neuron while maintaing the other 3 untouched, and then proceeding to do the inference on each dataset. There are 4 different datasets: **FCM_without_AVBL**, **FCM_without_AVBR**, **FCM_without_PLML2**, and **FCM_without_PLMR**. The intuition behind this experiment was to better comprehend the weight that each neuron exerted on the models, to test how the lack of input information affected the models, comparing the results of this experiment with the ones using the original dataset. The plots in Figure 5.2 describe an example of a prediction of features DB1, LUAL, PVR and VB1, using a test sequence of the original dataset.

An analysis of the results for this 4 case experiment without one neuron, where it was predicted the output of features DB1, LUAL, PVR and VB1 in the *C. elegans* nervous system, is presented in Section 6.2. To continue to study this matter and search for more in depth answers, another experiment follows.

5.4.2 Experiment without a pair of neurons

Following the previous experiment (Section 5.4.1), one thought that emerged was that maybe the use of 3 input features could be reduced to the utilization of 2 input features in the test sequences. The reason for this relates to the chance that using a specific pair of neurons could be the cause behind good predictions, and leaving only one neuron out is not enough to perceive that.

This experiment evaluates the performance of the model when: the information of the input on the PLM neurons is not available, the information of the input on the right neurons is missing, the information of the input on the left neurons is nowhere to be found, and the information of the input on the AVB neurons is inaccessible. Following this, 4 additional datasets were built based on joining the input features that had some correlation: **FCM_without_PLM** zeroing PLMR and PLML2, **FCM_without_R** zeroing AVBR and PLMR, **FCM_without_L** zeroing AVBL and PLML2, and **FCM_without_AVB** zeroing AVBR and AVBL. An analysis of the results for this 4 case experiment without a pair of neurons, where it was predicted the output of features DB1, LUAL, PVR and VB1 in the *C. elegans* nervous system, is presented in Section 6.3.

6

Results

Contents

This chapter presents the results from the utilization of DeepSHAP to explain the model, and the insights gained from the two experiments. Employing DeepSHAP, several different visualizations were plotted, the majority related to the sum of SHAP values throughout all time steps, to gain global insights, but also summary plots explaining individual time steps. For the experiments, conclusions were drawn by looking at the models' output predictions, that used less input data during inference. In the end, there is room for a discussion about the results.

6.1 Results from DeepSHAP

With the intention of trying to use SHAP to explain the GRU model and achieve some conclusions, visualizations related to SHAP values were plotted. For each output neuron, five different plots were introduced, using an example of the test set of **FCM-20**. The first one illustrates the sum of the SHAP values of each input feature through the 1000 time steps. The second is similar to the first one, but applying the sum of the absolute values of the SHAP values instead. The third plot represents the sum of the SHAP values of each input feature through the 1000 time steps but overlapped within the same time span, together with the true value of the output feature. The fourth plot contains two bar plots, one representing the sum of the SHAP values of each input feature through the 1000 time steps, and the other doing the same but using absolute values. The fifth one consists of two summary plots visualizing SHAP values in the time step when the output value is maximum, only differing in the visualization technique but representing essentially the same.

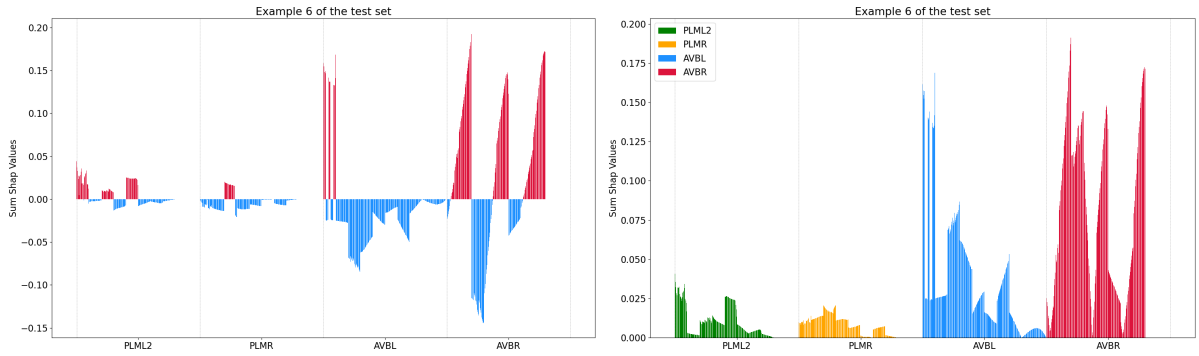

Figure 6.1: Sum of the SHAP values (left) and sum in absolute values of the SHAP values (right) of each input feature through the 1000 time steps for the neuron DB1 for one example of the test set of **FCM-20**.

For DB1, the AVB neurons are clearly more impactful than the PLM neurons, as can be observed in the right plot of Figure 6.1, since the SHAP values of the former have a higher magnitude than the SHAP values of the latter, with AVBR having some advantage over AVBL. Another observation is that all of the features present both positive and negative SHAP values throughout the 1000 time steps (left plot of Figure 6.1), varying in different moments, indicating that the impact of the features on the prediction is

not consistent across all time steps. Furthermore, this might indicate that the model considers different aspects of these features at different time steps. AVBR has positive and negative SHAP values, but there are more positive ones and they are more meaningful (larger magnitude) than the negative ones. AVBL, although having some positive meaningful SHAP values in the first time steps, has mainly negative ones, but less meaningful. In order to better understand all of these SHAP values in the context of predicting the neuron DB1, the next plot (Figure 6.2) is helpful.

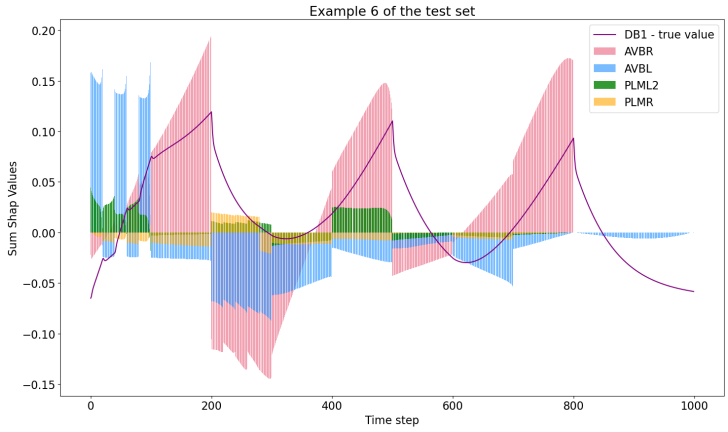

Figure 6.2: Sum of the SHAP values of each input feature through the 1000 time steps overlapped within the same time span, together with the true value of the output feature, for the neuron DB1 for one of the examples of the test set of **FCM-20**.

Analyzing Figure 6.2, it is possible to perceive the influence that the AVB neurons have on the output, specially AVBR. There is an interpretable relationship between the AVB input features and the output variable. When the SHAP values of the AVB neurons are positive (positive impact of the input features), particularly AVBR, the true output value tends to increase. Conversely, the output variable tends to decrease when these SHAP values are negative (negative impact of the input features), following the ”waveform” of the SHAP values. The model appears to be capturing the impact of these features (AVB neurons) accurately, since changes in these features' SHAP values correspond to changes in the true output value.

Taking a look at the example of inputs in Figure 5.1, which is the same example used here, there are similarities to the SHAP values that can be pointed out. In the initial time steps, the shape of the input AVBL is visible in the SHAP values of AVBL, reaching the value of 0.15. Then, the three ”climbs” of the SHAP values of AVBR, peaking in values above 0.15, are also evident on the input AVBR. In time steps 200, 500 and 800, the declines on the AVBR inputs from $2.4\,$nA to $0\,$nA provoke the fast switch on the corresponding SHAP values from very positive (0.2 in time step 200) to very negative (-0.15 in time step 300), and the output also follows. The AVB neurons, and specially AVBR, might have a threshold effect. For AVBR, when the input feature's value is greater than zero, it positively influences the model's predictions. However, when the feature's value is exactly zero, it negatively impacts the predictions.

AVBL also presents a similar relationship with the output variable, but is less meaningful. This suggests that the presence of any non-zero value on these features has a positive effect on the predictions, while a zero value has a detrimental impact.

It is evident that the AVB neurons are important for the model's prediction, since their impact influence the model's output, and so when explaining the model or making decisions based on its predictions, the AVB neurons should be a focal point. Moreover, the consistent relationship between the input features' SHAP values and the true output values provides strong evidence of the features' importance and their impact on the model's predictions. This understanding can enhance the trust in the model's predictions, the insights derived from its behavior and the overall explainability of the model.

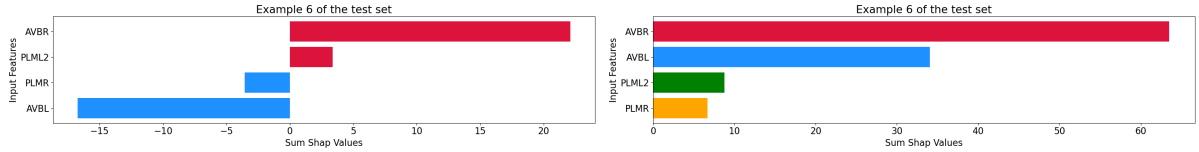

Figure 6.3: Bar plot of the sum of SHAP values (left) and of the sum of the absolute values of SHAP values (right) of each input feature through the 1000 time steps for the neuron DB1 for one of the examples of the test set of **FCM-20**.

By looking at the right plot of Figure 6.3 it is possible to observe the same thing about the AVB neurons as previously stated, these features present the largest absolute sum of SHAP values over all time steps, which means that they are fulcral in predicting the output neuron DB1. When analyzing the regular sum of SHAP values over all time steps (left plot of Figure 6.3), it is comprehended that AVBR has mainly positive SHAP values, and the majority of SHAP values of AVBL are negative. As seen before, this is due to the presence of a large portion of input values equal to zero (Figure 5.1).

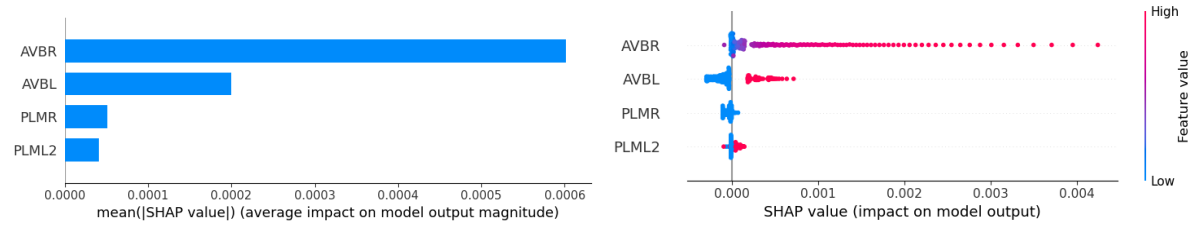

Figure 6.4: Summary plots of SHAP values of each input feature in the time step when the output value is maximum (200) for the neuron DB1 for one of the examples of the test set of **FCM-20**.

The left plot of Figure 6.4 represents the importance of the features exhibiting substantial SHAP values in absolute terms in time step 200, when the true output value achieves its maximum point. Not surprisingly, the AVB neurons are the key features and AVBR stands out. The right plot of Figure 6.4 reveals that in time step 200, higher input values of AVBL and AVBR cause higher SHAP values.

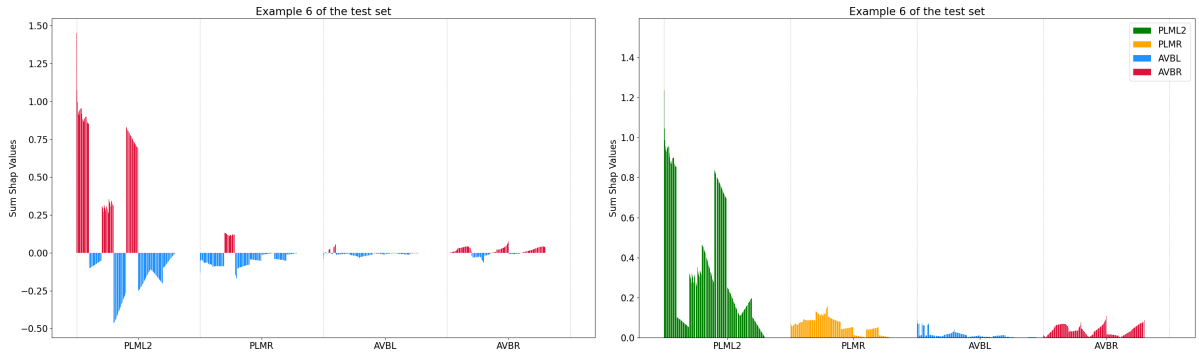

Figure 6.5: Sum of the SHAP values (left) and sum in absolute values of the SHAP values (right) of each input feature through the 1000 time steps for the neuron LUAL for one of the examples of the test set of **FCM-20**.

In the case of LUAL (Figure 6.5), PLML2 is clearly the most impactful feature, and the positive SHAP values have larger magnitude than the negative ones. PLMR is the second most impactful feature, although far from PLML2, and is mainly composed by negative SHAP values. Conversely to DB1, the AVB neurons present much smaller SHAP values than the PLM neurons, with AVBL achieving the lowest ones. Similar insights from the analysis of DB1 can be drawn in terms of all of the features presenting both positive and negative SHAP values across all time steps, and in order to better understand all of these SHAP values in the context of predicting the neuron LUAL, the next plot (Figure 6.6) is helpful.

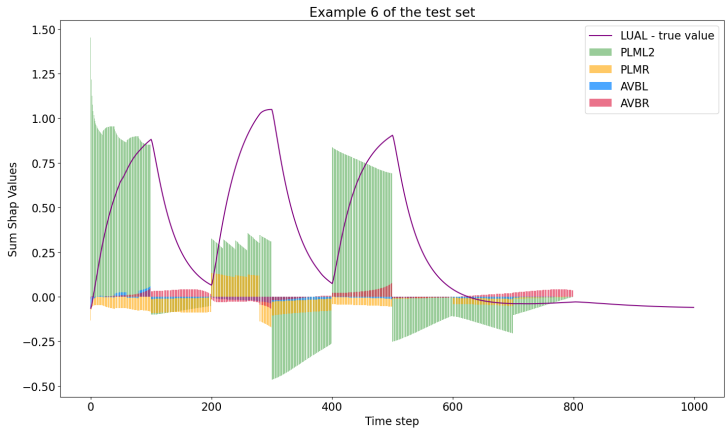

Figure 6.6: Sum of the SHAP values of each input feature through the 1000 time steps overlapped within the same time span, together with the true value of the output feature, for the neuron LUAL for one of the examples of the test set of **FCM-20**.

Observing Figure 6.6, PLML2 is the most influential neuron to the output, since the output follows the positive and negative SHAP values of PLML2. In time step 300, the LUAL output achieves its highest value (above 1.00), while the PLM neurons present positive SHAP values. The contrary is observed when the PLM neurons have both negative values, the output of LUAL decreases. This is also observed in the analysis of DB1, although with different features. These examples show the impact that the PLM

neurons have on LUAL.

Looking at Figure 5.1, a similar insight from DB1 can be drawn in this instance. For the PLM neurons, when the input features' values are greater than zero, it positively influences the model's predictions. However, when the features' values are exactly zero, it negatively impacts the predictions. This suggests that the presence of any non-zero value on these features has a positive effect on the predictions, while a zero value has a detrimental impact.

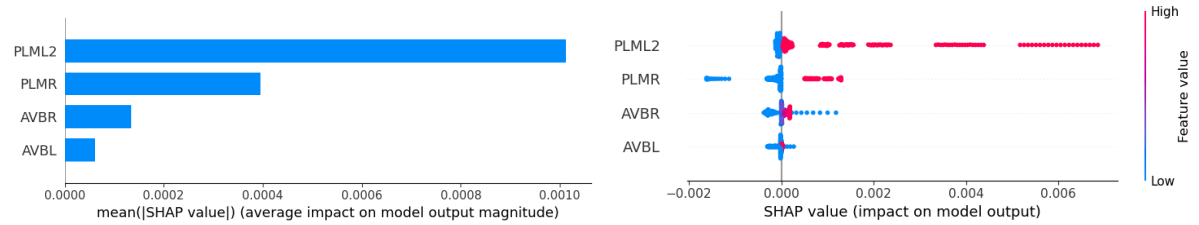

Figure 6.7: Summary plots of SHAP values of each input feature in the time step when the output value is maximum (300) for the neuron LUAL for one of the examples of the test set of **FCM-20**.

The left plot of Figure 6.7 represents the importance of the features exhibiting substantial SHAP values in absolute terms in time step 300, when the true output value achieves its maximum point. Without any doubt, the PLM neurons are the most important features and PLML2 stands out. The right plot of Figure 6.7 reveals that in time step 300, higher input values of the PLM pair cause higher SHAP values. An interesting observation can be made in the AVBR feature, where higher input values provoke positive SHAP values but with small magnitude, but lower input values produce positive and negative SHAP values, the positive ones being more positive than the ones perceived by the higher input values.

Regarding PVR and VB1, their analysis are similar to the ones stated in LUAL and DB1, respectively, and to make these conclusions more concise they will not be thoroughly discussed. For PVR the most important and impactful pair of neurons are the PLM, with PLMR presenting more impact than PLML2. The relationship with the true output value is essentially the same, the PLM neurons are important for the model's prediction since their impact influence the model's output, and so when explaining the model or making decisions based on its predictions, the PLM neurons should be taken into account first. For VB1, the most impactful pair of neurons are the AVB. The relationship with the true output value is very similar to the one analyzed in DB1. All of the figures regarding the results from the utilization of DeepSHAP on every example are plotted in Appendix A.

In summary, this analysis using the SHAP values leads to the conclusion that some inputs, or pair of inputs, are more relevant than others for the models to predict the output voltage of some neurons in the *C. elegans* nervous system. The **AVB** pair are the key features for DB1 and VB1. For LUAL and PVR, the **PLM** pair are the most important features, with PLML2 standing out in the former and PLMR in the latter. With the goal of validating the conclusions obtained using SHAP, two different experiments, that involve reducing the input data during inference, were put into practice in the next sections.

6.2 Results from Experiment without 1 neuron

The plots in Figures 6.8 to 6.11 represent the 4 case experiment without one neuron where it was predicted the output of features DB1, LUAL, PVR and VB1 in the *C. elegans* nervous system, ordered respectively.

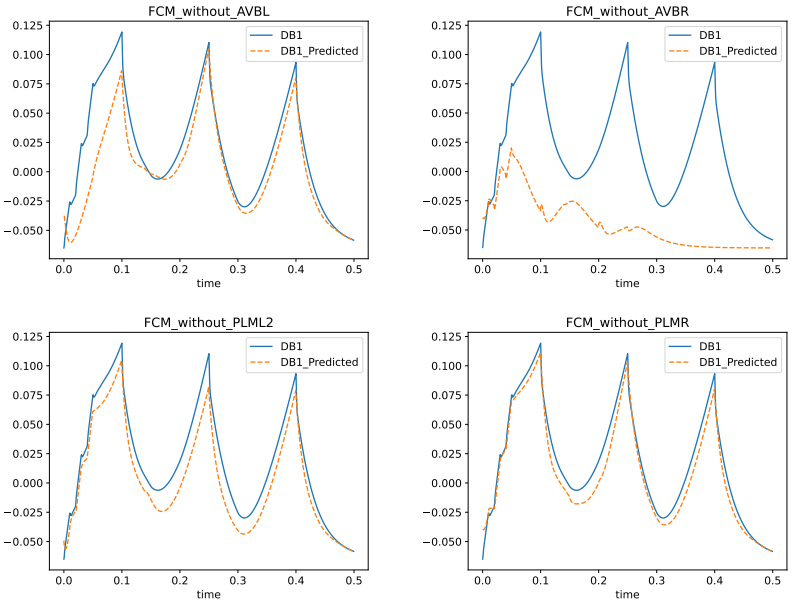

Figure 6.8: Example of outputs predicted for feature DB1 without one different input neuron at each time for a test sequence of the novel datasets stated above, ordered respectively.

By looking at the display of plots some conclusions can be drawn. An obvious one is that the overall best predictions of the output voltage of the 4 output neurons are made by the model using the 4 input features (Figure 5.2), which was expected.

In Figure 6.8 one of the plots stands out negatively for having the worst output prediction, but, in reality, this has a positive meaning. The prediction in question is plotted by the model that is not using the input feature AVBR in the inference, denoting that **AVBR** is actually a key feature here, since the other 3 models did not fail to predict the output voltage of DB1. The closest one is the model without AVBL, then the model not using PLML2, and the best model does not utilize the input feature PLMR. This last one is the model which replicates the output voltage of the neuron DB1 in the system more accurately.

In Figure 6.9, once again and even more obviously, one of the plots is significantly different from the others. This time, the model that does not have the input feature **PLML2** is giving the worst output predictions. If there is no AVBR or AVBL in the test sequences for the inference, just one at a time and not both concurrently, there is also almost no difference when comparing to the inference using the original dataset. Not having PLMR is just slightly worse than the better ones.

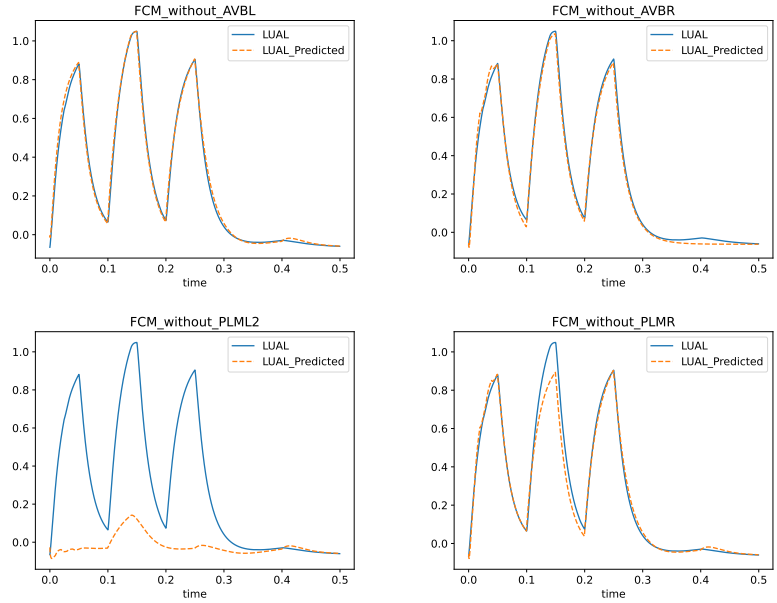

Figure 6.9: Example of outputs predicted for feature LUAL without one different input neuron at each time for a test sequence of the novel datasets stated above, ordered respectively.

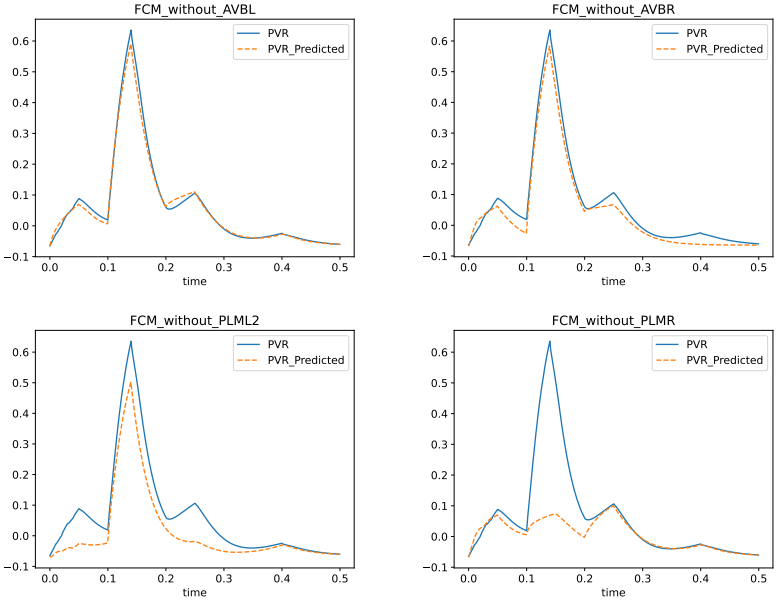

Figure 6.10: Example of outputs predicted for feature PVR without one different input neuron at each time for a test sequence of the novel datasets stated above, ordered respectively.

The third example (Figure 6.10) is different to some degree than the first two. Here, once more, two plots give good predictions (model without AVBR and model without AVBL), a third one predicts acceptably (model without PLML2), and the last one fails completely to predict the output voltage of the PVR neuron (model without **PLMR**). In this last one, despite not giving acceptable output predictions, it

is possible to perceive that there are time intervals with good predictions overall, from time step 0 to time step 200 and then from time step 400 until the end, only predicting badly from time step 200 to time step 400.

Looking back to the third plot that predicts acceptably, there is potentially an inverse of the worst plot here, because where one model predicts acceptably well, the other one is bad, and vice-versa.

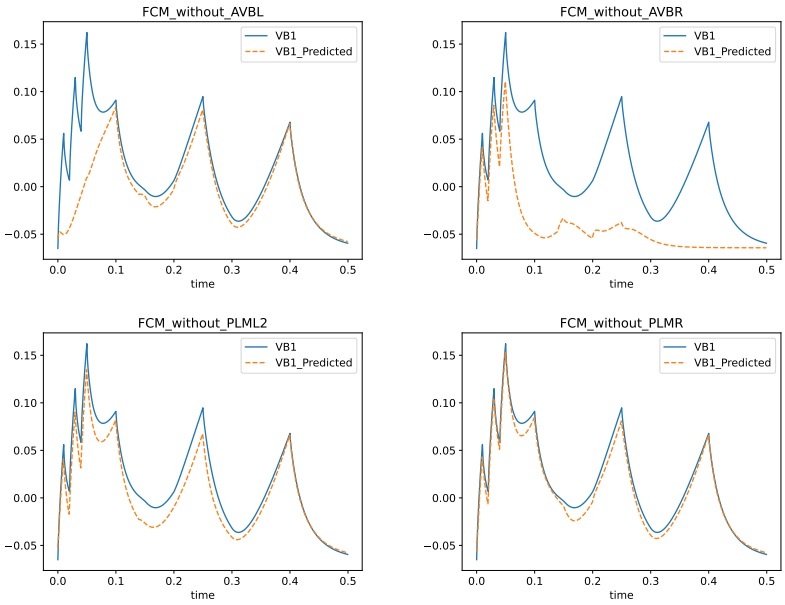

Figure 6.11: Example of outputs predicted for feature VB1 without one different input neuron at each time for a test sequence of the novel datasets stated above, ordered respectively.

In Figure 6.11 similar conclusions to the ones in the previous can be taken. The model that does not have PLML2 and the model that does not have PLMR give good predictions, but the results from the model that does not have AVBL (good prediction after time step 200) and the results from the model that does not have **AVBR** (good prediction until time step 100) are more or less the opposite of each other.

In summary, this experiment leads to the conclusion that some inputs, or pair of inputs, are more relevant than others for the models to predict the output voltage of some neurons in the *C. elegans* nervous system. The **AVB** pair are the key features for DB1 and VB1, with AVBR having more impact than AVBL. For LUAL and PVR, the **PLM** pair are the most important features, with PLML2 being more important in the former case and PLMR in the latter.

6.3 Results from Experiment without a pair of neurons

The plots in Figures 6.12 to 6.15 represent the 4 case experiment without a pair of neurons where it was predicted the output of features DB1, LUAL, PVR and VB1 in the *C. elegans* nervous system, ordered

respectively. Looking at the plots, one can say that there are good, bad and acceptable predictions, and the predictions corresponding to each output feature can be more or less ranked from best to worst.

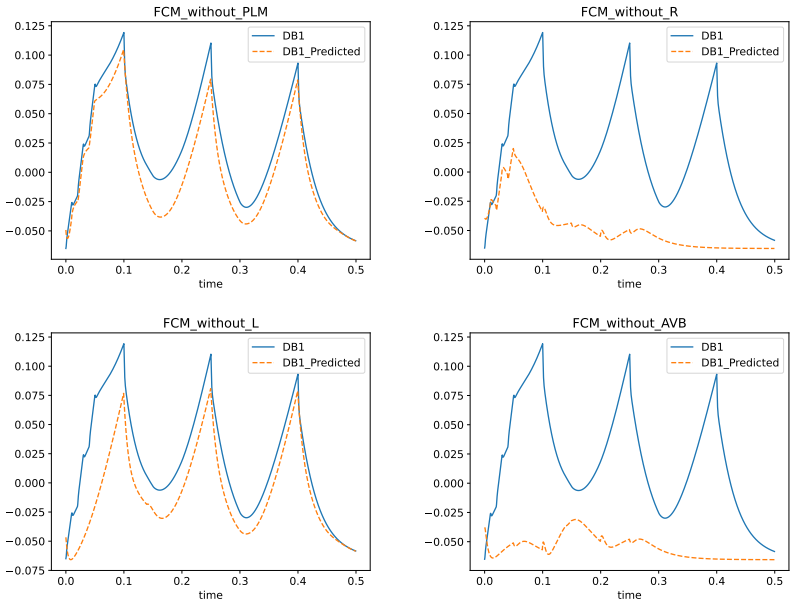

Figure 6.12: Example of outputs predicted for feature DB1 without a different pair of input neurons at each time for a test sequence of the novel datasets stated above, ordered respectively.

Starting with the analysis of the plots of the outputs predicted for feature DB1 (Figure 6.12), the finest predictions are achieved by the model that only uses AVBL and AVBR (plot 1 of Figure 6.12), although they are not quite as good as the model without PLMR or the model without PLML2 (refer to Figure 6.8, plots 4 and 3, respectively). In this case, one thought might be that neither PLML2 or PLMR have much importance, because without them the models can predict the output voltage of DB1 successfully.

Then, the model using the pair AVBR and PLMR (plot 3 of Figure 6.12) give acceptable predictions, but worse than the model without PLML2 and a bit worse than the model without AVBL (refer to Figure 6.8, plots 3 and 1, respectively). From this, a possible deduction might be that PLMR and AVBL do not have much influence, and that the shared input feature, **AVBR**, is a key feature, as can be observed in plot 2 of Figure 6.8.

The last two plots are bad, one of them can even be considered super poor. In the model that employs AVBL and PLML2 (plot 2 of Figure 6.12), the predicted value line tries to catch the true value in the very beginning but from time step 100 forward is really bad. The prediction is considerably worse than in the model without PLMR, but very similar to the model where AVBR is kept with value zero (refer to Figure 6.8, plots 4 and 2, respectively). In this case, **AVBR** is the input feature with the most importance of all, since the prediction improves significantly when that input feature is added to the model with AVBL and PLML2, as can be seen in plot 4 of Figure 6.8.

The last and worse plot is observed in the model giving use to PLML2 and PLMR (plot 4 of Figure 6.12), but if the input feature AVBR is added to these previous two, as can be seen in plot 1 of Figure 6.8, the prediction improves greatly. The same thing is not observed if the input feature AVBL is added to the model using the PLM pair (plot 2 of Figure 6.8). From analyzing the plots of the outputs predicted for feature DB1, the main conclusion is that the **AVB** pair are the key features, and AVBR is the main contributor for this.

The next analysis segment is focused on the predictions for output feature LUAL. In this instance, two of the plots are good and the other two are terrible. The best one, which has a very good prediction line, corresponds to the model that uses the PLM pair (plot 4 of Figure 6.13). So, one can conclude that AVBL and AVBR are not important whatsoever, which can be proven by looking at the plots of the model without AVBL and at the plots of the model without AVBR (refer to Figure 6.9, plots 1 and 2, respectively).

The second best model is composed by AVBL and PLML2 (plot 2 of Figure 6.13), and taking into consideration that it is also pretty good, one possible assumption could be that the only input feature in common, **PLML2**, is the reason behind the good predictions. This can be proven by the fact that the prediction of the model that does not include PLML2 (refer to plot of 3 Figure 6.9) is really bad.

Additionally, the last two plots, that are terrible, are the ones corresponding to the model that does not include PLML2. From analyzing the plots of the outputs predicted for feature LUAL, the main conclusion is that the **PLM** pair are the key input features, with PLML2 being the biggest contributor for that.

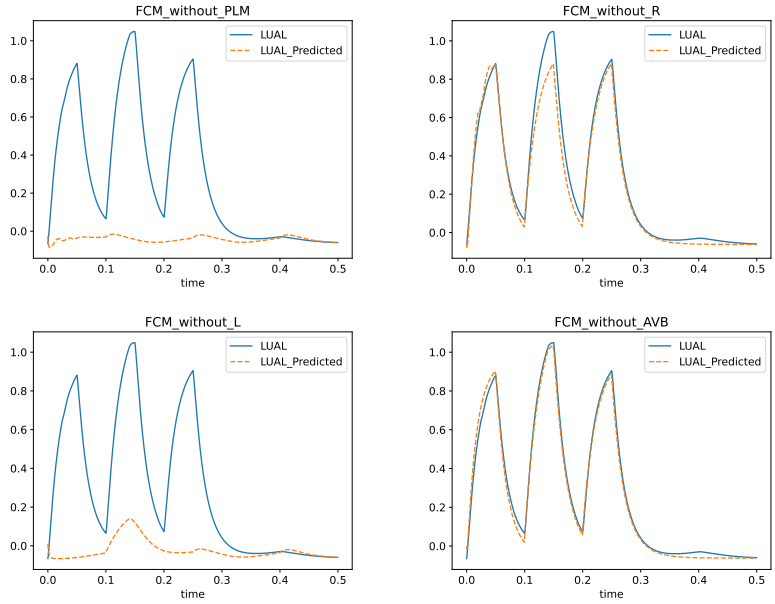

Figure 6.13: Example of outputs predicted for feature LUAL without a different pair of input neurons at each time for a test sequence of the novel datasets stated above, ordered respectively.

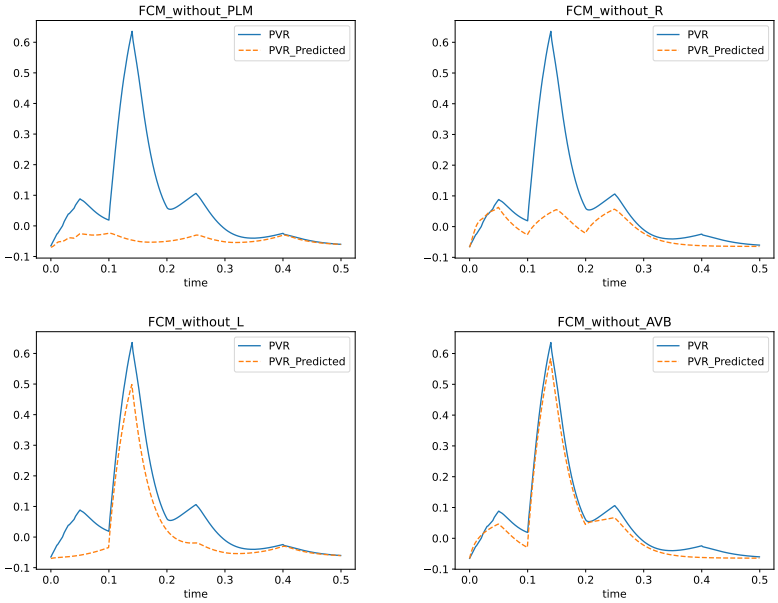

Figure 6.14: Example of outputs predicted for feature PVR without a different pair of input neurons at each time for a test sequence of the novel datasets stated above, ordered respectively.

Relatively to the output predictions for feature PVR, the models that contain **PLMR** give the most favorable predictions (refer to Figure 6.14, plots 4 and 3, in order), and the models without that input feature predict very poorly (refer to Figure 6.14, plots 2 and 1, in order).

This can be further manifested discerning Figure 6.10, where the exclusively bad plot is given by the model in the absence of PLMR (plot 4). With that said, it is clear that PLMR is a very important input feature in this instance, and the **PLM** pair of neurons are the key input features.

Based on Figure 6.15, the worst predictive performance plot is provided by the model that makes use of the input features PLML2 and PLMR (plot 4), which translates in them not having any influence here.

Conversely, the best predictive performance plot is given by the model that uses the input features **AVBL** and **AVBR** (plot 1).

The remaining plots behave inversely, one model (AVBL+PLML2) performs well until time step 100 and poorly afterwards, the other model (AVBR+PLMR) shows an unsatisfactory prediction line until time step 200 but has a good performance subsequently.

Since the **AVB neurons** are the most important features in this case, looking at the plots of the model without AVBL (plot 1) and of the model without AVBR (plot 2) in Figure 6.11, the latter performs better in the initial time steps, while the former does it greater after time step 200. And so, there is not a single most important input feature in this instance. Instead, the **AVB** pair are the key input features in this case.

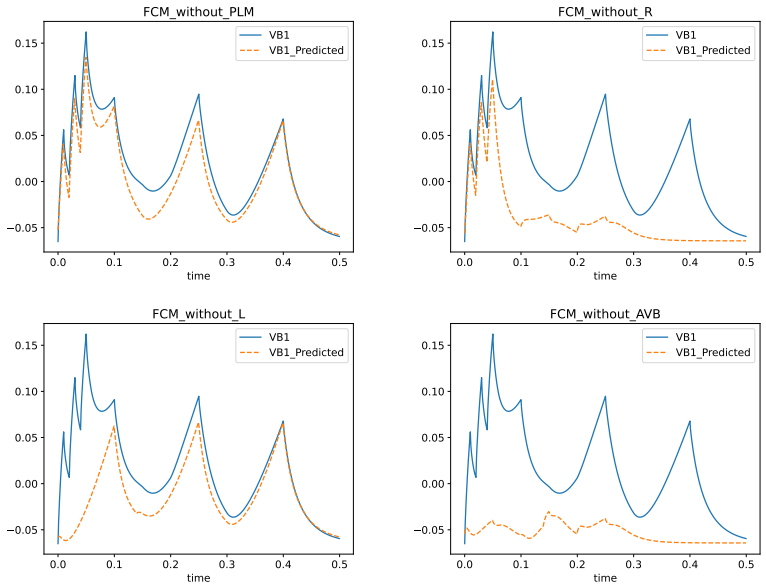

Figure 6.15: Example of outputs predicted for feature VB1 without a different pair of input neurons at each time for a test sequence of the novel datasets stated above, ordered respectively.

In summary, this experiment leads to the conclusion that, once again, some inputs, or pair of inputs, are more relevant than others for the models to predict the output voltage of some neurons in the *C. elegans* nervous system: The **AVB** pair are the key input features for DB1, with AVBR being the most important feature; The **PLM** pair of neurons are the key input features for LUAL, with PLML2 being the most important feature for predicting the output voltage of that neuron; For PVR, the **PLM** pair of input features are the key input features, and PLMR is the main contributor for that; The **AVB** pair are the most important features for the output neuron VB1.

6.4 Discussion about the Results

Comparing the conclusions of the experiments (Sections 5.4.1 and 5.4.2) with the ones from using SHAP (Section 6.1), it is perceivable that the assumptions are in agreement. However, it is essential to remember that this experiment only uses one out of ten examples of the testing data, and it is not guaranteed that the deductions taken from this example are replicated in the remaining, meaning that global conclusions must take into consideration all of the examples. To keep these conclusions concise, it is not possible to analyse all of existing examples one by one, but all of the plots considering the whole test data are located in Appendix A. After a careful analysis of the DeepSHAP results using the whole test set, it is safe to say that the conclusions from Section 6.1 are reflected in all of the examples. Consequently, the results from the two experiments can corroborate and validate the insights drawn from applying DeepSHAP to the black box model.

7

Conclusions and Future Work

Contents

7.1 Conclusions

This Thesis delved into XAI and ML Interpretability and its application for TS and MTS regression models. Since it is a relatively new area of research, XAI has been mostly focused on more intuitive applications like natural language processing and computer vision, and not so much on TS data. While there is research done in XAI for TS data, it is most of the time centered on classification and forecasting tasks.

This work focused on the analysis of post hoc XAI approaches for TS data, and achieving explainability in a MTS regression problem, understanding the decision-making process of the underlying DL model as well as draw actionable insights. Post hoc explanation methods were chosen over intrinsic ones, since these are not restricted by the type of model being used.

The context of the problem centers around trying to understand the foundational aspects of neural dynamics and overall brain organization, utilizing a roundworm named *Caenorhabditis Elegans* for which the connectome is completely mapped, illustrating all of the neuronal connections in the nervous system. The used dataset is composed essentially by responses of the roundworm's nervous system under diverse stimuli, where both inputs and outputs are TS data with multiple features. The used DL model was a GRU, and was used to predict the output voltage of a single neuron, being utilized 4 times each to predict a different neuron. Then, a XAI technique was applied to the black box model, SHAP, for creating explanations. SHAP was the chosen technique to be applied, since the literature review revealed that it is the most complete post hoc local explainability method, albeit being computationally expensive. It provides both global and local explanations, the global ones explain the model and its decisions in a general way, and the local ones explain the behavior of the model for a single input. It highlights how important each feature in each time step in the input was to the prediction. Lastly, two experiments were proposed, that consisted in providing less input data during inference, with the goal of trying to understand which features were more important in different contexts, and comparing these results to the ones achieved by applying the explanation method and validate them.

With the application of SHAP it was possible to achieve global and local explainability for the 4 models. By analyzing SHAP values across all data points, it was possible to gain insights into the global behavior of the DL models, understanding which features are consistently important for different inputs. SHAP provided local explainability since it is possible to understand the reason for a specific prediction, utilizing SHAP visualization tools like force plot. The results from the two experiments further validated the results achieved by SHAP, where some inputs, or pair of inputs, are more relevant than others for the models to predict the output voltage of some neurons in the *C. elegans* nervous system.

7.2 Future Work

Taking into account all of the work done in this Thesis, there are some points that can be improved. In a future work, regarding DeepSHAP, interactions between features could have been explored in depth, which would have probably unveiled new insights and increased the explainability of the model. Furthermore, LIME could have been explored as much as SHAP was, since in other publications it was considered a good explanation model to obtain explainability for models using TS data. Also, more DL models could have been explored and their results compared to the ones achieved using GRU. Moreover, different behavior scenarios could be tested in future work.

XAI for MTS regression problems is an uncovered area of research, but there is a lot of potential knowledge that can be extracted from these black box models to further provide understanding of the models and trust to end-users, and as future work researchers should start giving it more attention.

Bibliography

[1] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1135–1144.

[2] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, p. 18, 2020.

[3] C. Molnar, *Interpretable Machine Learning*, 2nd ed., 2022.

[4] N. Jakubiak, "Analysis of explainable artificial intelligence on time series data," Ph.D. dissertation, UPC, Facultat d'Informàtica de Barcelona, Departament de Ciències de la Computació, Oct 2022.

[5] B. Hayes, "Ben Hayes - Interpreting Machine Learning with SHAP — benhay.es."

[6] C. K. Dataman, "The SHAP with More Elegant Charts — medium.com."

[7] D. A. Hod, "Council Post: Explainable AI: The Importance Of Adding Interpretability Into Machine Learning — forbes.com."

[8] O. O. D. Science, "The Importance of Explainable AI — medium.com."

[9] S. T. P. Limited, "What is Explainable AI? Which industries are meant for XAI? — seaflux.tech."

[10] G. D. P. Regulation, "General data protection regulation (gdpr)," *Intersoft Consulting, Accessed in October*, vol. 24, no. 1, 2018.

[11] T. Rojat, R. Puget, D. Filliat *et al.*, "Explainable artificial intelligence (xai) on timeseries data: A survey," 2021.

[12] B. Lim, S. Arık, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.

[13] W. E. Marcílio and D. M. Eler, “From explanations to feature selection: assessing shap values as feature selection mechanism,” in *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2020, pp. 340–347.

[14] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, “Gene selection for cancer classification using support vector machines,” *Machine Learning*, vol. 46, no. 1, pp. 389–422, Jan 2002.

[15] T. Guo, T. Lin, and N. Antulov-Fantulin, “Exploring interpretable LSTM neural networks over multi-variable data,” in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 2494–2504.

[16] K. E. Mokhtari, B. P. Higdon, and A. Başar, “Interpreting financial time series with shap values,” in *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON ’19. USA: IBM Corp., 2019, p. 166–172.

[17] O. Ozyegen, I. Ilic, and M. Cevik, “Evaluation of interpretability methods for multivariate time series forecasting,” *Applied Intelligence*, vol. 52, no. 5, pp. 4727–4743, Mar 2022.

[18] T. Miller, “Explanation in artificial intelligence: Insights from the social sciences,” *Artificial Intelligence*, vol. 267, pp. 1–38, 2019.

[19] R. Marcinkevics and J. E. Vogt, “Interpretability and explainability: A machine learning zoo mini-tour,” *CoRR*, vol. abs/2012.01805, 2020.

[20] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, “Machine learning interpretability: A survey on methods and metrics,” *Electronics*, vol. 8, no. 8, 2019.

[21] L. H. Gilpin, D. Bau, B. Z. Yuan *et al.*, “Explaining explanations: An overview of interpretability of machine learning,” in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018, pp. 80–89.

[22] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning,” 2017.

[23] Z. C. Lipton, “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.” *Queue*, vol. 16, no. 3, p. 31–57, jun 2018.

[24] A. Adadi and M. Berrada, “Peeking inside the black-box: A survey on explainable artificial intelligence (xai),” *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[25] B. Kim, R. Khanna, and O. O. Koyejo, “Examples are not enough, learn to criticize! criticism for interpretability,” in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[26] T. Begley, T. Schwedes, C. Frye, and I. Feige, “Explainability for fair machine learning,” *CoRR*, vol. abs/2010.07389, 2020.

[27] S. Padmanabhuni, “Model Agnostic vs Model Specific Explainability — srinivaspadmanabhuni.”

[28] C. Grady, “Institutional review boards: Purpose and challenges,” *Chest*, vol. 148, no. 5, pp. 1148–1155, Nov. 2015.

[29] M. Robnik-Šikonja and M. Bohanec, *Perturbation-Based Explanations of Prediction Models*. Cham: Springer International Publishing, 2018, pp. 159–175.

[30] J. a. Bento, P. Saleiro, A. F. Cruz, M. A. Figueiredo, and P. Bizarro, “Timeshap: Explaining recurrent models through sequence perturbations,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 2565–2573.

[31] G. Jiang, F. Zhuang, B. Song, J. Li *et al.*, “SeqSHAP: Subsequence level shapley value explanations for sequential predictions,” 2023.

[32] A. Naghibzadeh-Jalali, A. Schindler, B. Haslhofer, and A. Rauber, “Machine learning interpretability techniques for outage prediction: A comparative study,” in *PHM papers PROCEEDINGS 2020*, 2020, pp. 1–10, prognostics and Health Management Europe ; Conference date: 27-07-2020 Through 31-07-2020.

[33] U. Schlegel, D. L. Vo, D. A. Keim, and D. Seebacher, “Ts-mule: Local interpretable model-agnostic explanations for time series forecast models,” in *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, M. Kamp, I. Koprinska, A. Bibal *et al.*, Eds. Cham: Springer International Publishing, 2021, pp. 5–14.

[34] S.-C. Lu, C. L. Swisher, C. Chung *et al.*, “On the importance of interpretable machine learning predictions to inform clinical decision making in oncology,” *Frontiers in Oncology*, vol. 13, 2023.

[35] D. Garreau and U. von Luxburg, “Explaining the explainer: A first theoretical analysis of lime,” in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 1287–1296.

[36] D. Alvarez-Melis and T. S. Jaakkola, “On the robustness of interpretability methods,” *CoRR*, vol. abs/1806.08049, 2018.

[37] A. Shrikumar, P. Greenside, and A. Kundaje, “Learning important features through propagating activation differences,” in *Proceedings of the 34th International Conference on Machine Learning*,

ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 3145–3153.

[38] H. Kuhn and A. Tucker, *Contributions to the Theory of Games (AM-28), Volume II*, ser. Annals of Mathematics Studies. Princeton University Press, 2016.

[39] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowledge and Information Systems*, vol. 41, no. 3, pp. 647–665, Dec 2014.

[40] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[41] "GitHub - shap/shap: A game theoretic approach to explain the output of any machine learning model. — github.com."

[42] S. M. Lundberg, G. G. Erion, and S. Lee, "Consistent individualized feature attribution for tree ensembles," *CoRR*, vol. abs/1802.03888, 2018.

[43] M. Sundararajan and A. Najmi, "The many shapley values for model explanation," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 9269–9278.

[44] D. Janzing, L. Minorics, and P. Bloebaum, "Feature relevance quantification in explainable ai: A causal problem," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 2907–2916.

[45] "shap.DeepExplainer x2014; SHAP latest documentation — shap-lrjball.readthedocs.io."

[46] "Deep Explainer (Deep Shap) — arize.com."

[47] B. J. Aboze, "A Comprehensive Guide into SHAP Values — deepchecks.com."

[48] "lime package x2014; lime 0.1 documentation — lime-ml.readthedocs.io."

[49] "waterfall plot x2014; SHAP latest documentation — shap.readthedocs.io."

[50] F. A. Azevedo, L. R. Carvalho, L. T. Grinberg *et al.*, "Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain," *Journal of Comparative Neurology*, vol. 513, no. 5, pp. 532–541, 2009.

[51] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner, “The structure of the nervous system of the nematode caenorhabditis elegans,” *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, vol. 314, no. 1165, pp. 1–340, Nov. 1986.

[52] G. L. C. Mestre, “Modeling C. Elegans Nervous System’s Behavior using Machine Learning Techniques,” Master’s thesis, 2021.

[53] R. Barbulescu and L. M. Silveira, “Black-box model reduction of the c. elegans nervous system,” in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, 2021, pp. 4174–4179.

[54] R. Barbulescu, G. Mestre, A. L. Oliveira, and L. M. Silveira, “Learning the dynamics of realistic models of c. elegans nervous system with recurrent neural networks,” *Scientific Reports*, vol. 13, no. 1, p. 467, Jan 2023.

[55] N. Carnevale and M. Hines, *The NEURON Book*. Cambridge University Press, 2006.

[56] G. Van Rossum, F. L. Drake *et al.*, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995, vol. 111.

[57] C. François, “Keras,” 2015.

[58] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.

A

Additional Images from SHAP Results

A.1 Analysis of PVR

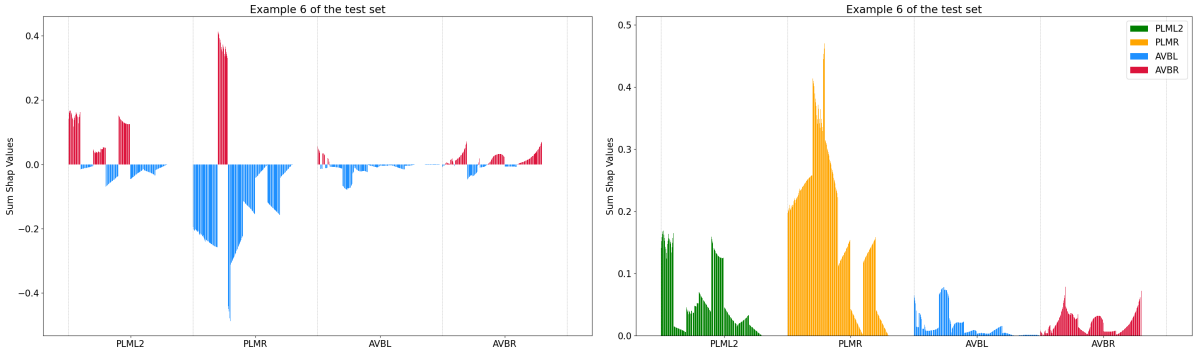

Figure A.1: Sum of the SHAP values (left) and sum in absolute values of the SHAP values (right) of each input feature through the 1000 time steps for the neuron PVR for one example of the test set of **FCM-20**.

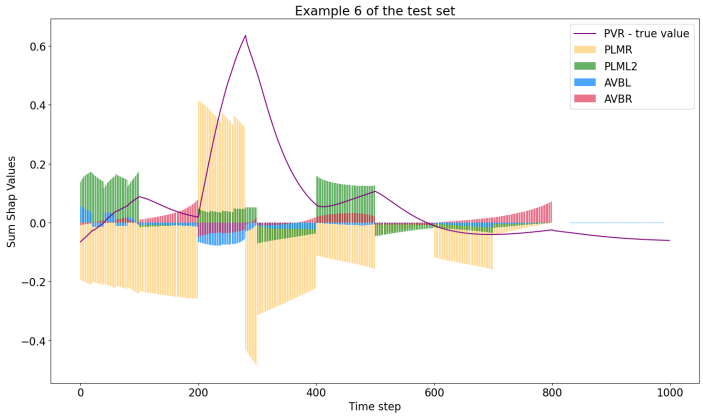

Figure A.2: Sum of the SHAP values of each input feature through the 1000 time steps overlapped within the same time span, together with the true value of the output feature, for the neuron PVR for one of the examples of the test set of **FCM-20**.

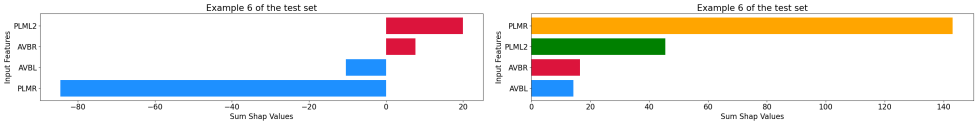

Figure A.3: Bar plot of the sum of SHAP values (left) and of the sum of the absolute values of SHAP values (right) of each input feature through the 1000 time steps for the neuron PVR for one of the examples of the test set of **FCM-20**.

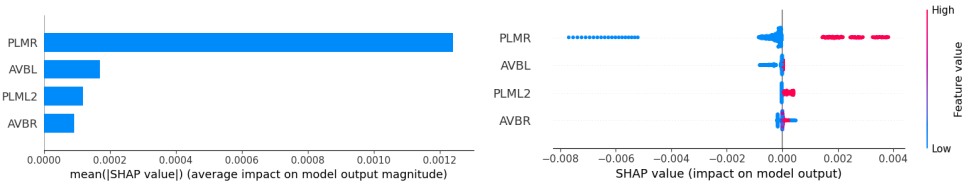

Figure A.4: Summary plots of SHAP values of each input feature in the time step when the output value is maximum (200) for the neuron PVR for one of the examples of the test set of **FCM-20**.

A.2 Analysis of VB1

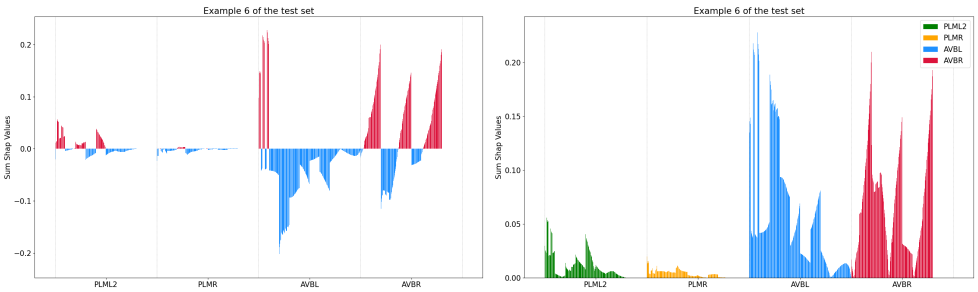

Figure A.5: Sum of the SHAP values (left) and sum in absolute values of the SHAP values (right) of each input feature through the 1000 time steps for the neuron VB1 for one example of the test set of **FCM-20**.

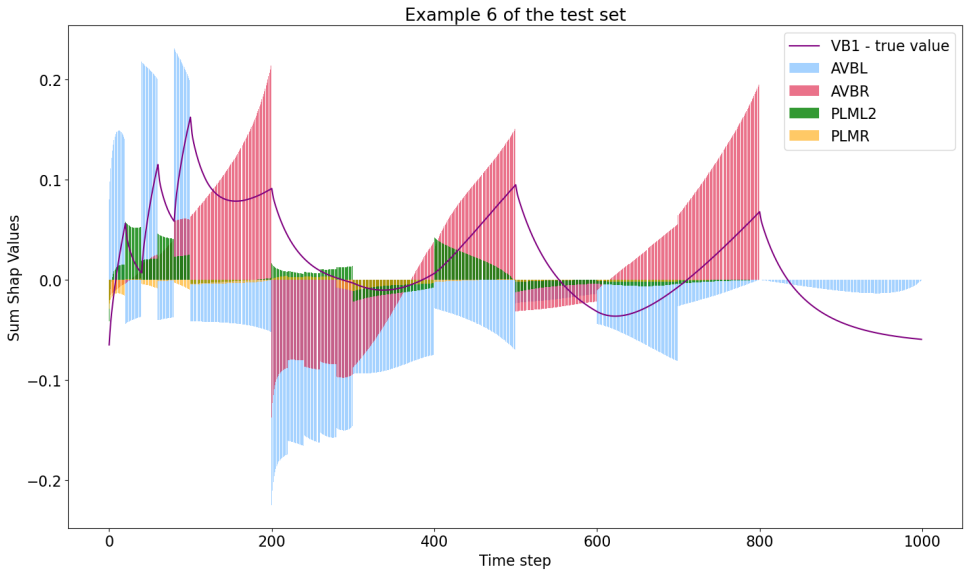

Figure A.6: Sum of the SHAP values of each input feature through the 1000 time steps overlapped within the same time span, together with the true value of the output feature, for the neuron VB1 for one of the examples of the test set of **FCM-20**.

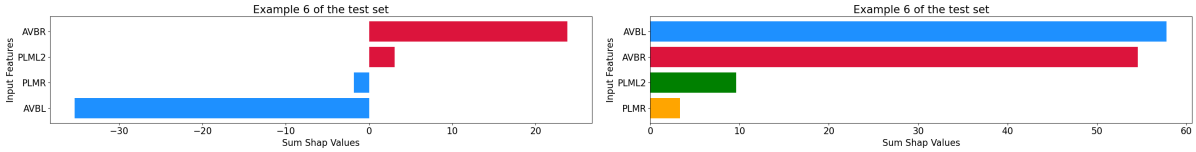

Figure A.7: Bar plot of the sum of SHAP values (left) and of the sum of the absolute values of SHAP values (right) of each input feature through the 1000 time steps for the neuron VB1 for one of the examples of the test set of **FCM-20**.

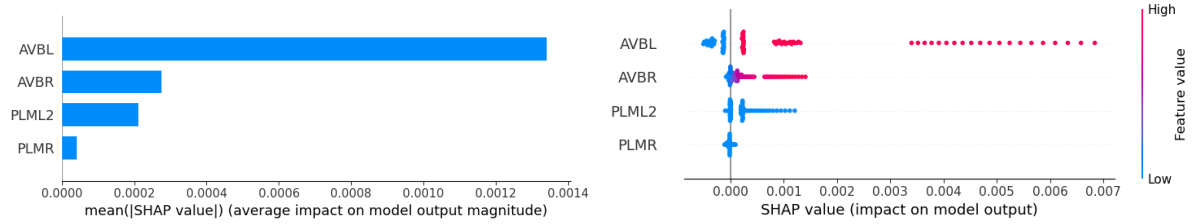

Figure A.8: Summary plots of SHAP values of each input feature in the time step when the output value is maximum (200) for the neuron VB1 for one of the examples of the test set of **FCM-20**.

A.3 Results for every example

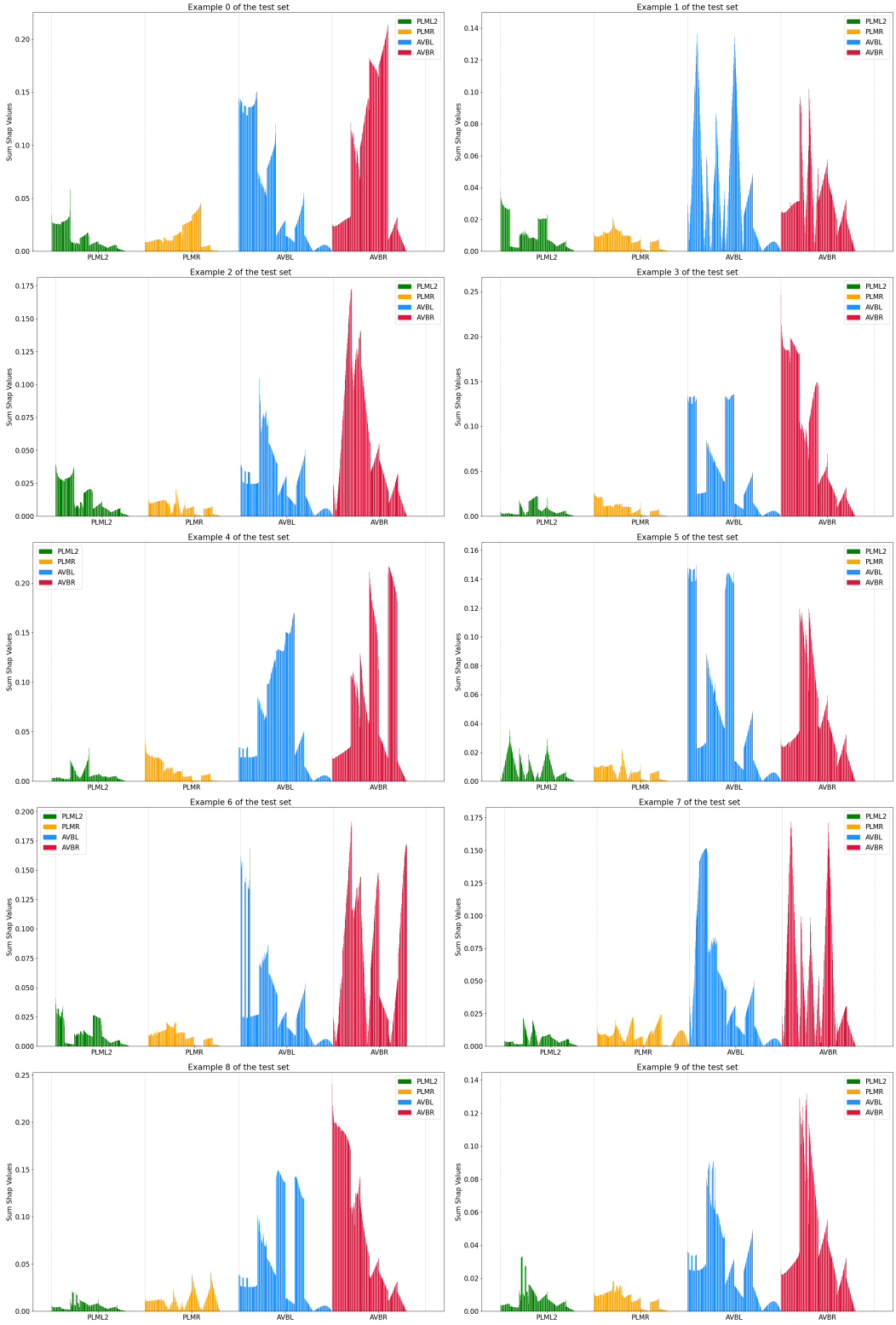

Figure A.9: Sum in absolute values of the SHAP values of each input feature through the 1000 time steps for the neuron DB1 for all of the examples of the test set of **FCM-20**, ordered from left to right and up-down.

Figure A.10: Sum in absolute values of the SHAP values of each input feature through the 1000 time steps for the neuron LUAL for all of the examples of the test set of **FCM-20**, ordered from left to right and up-down.

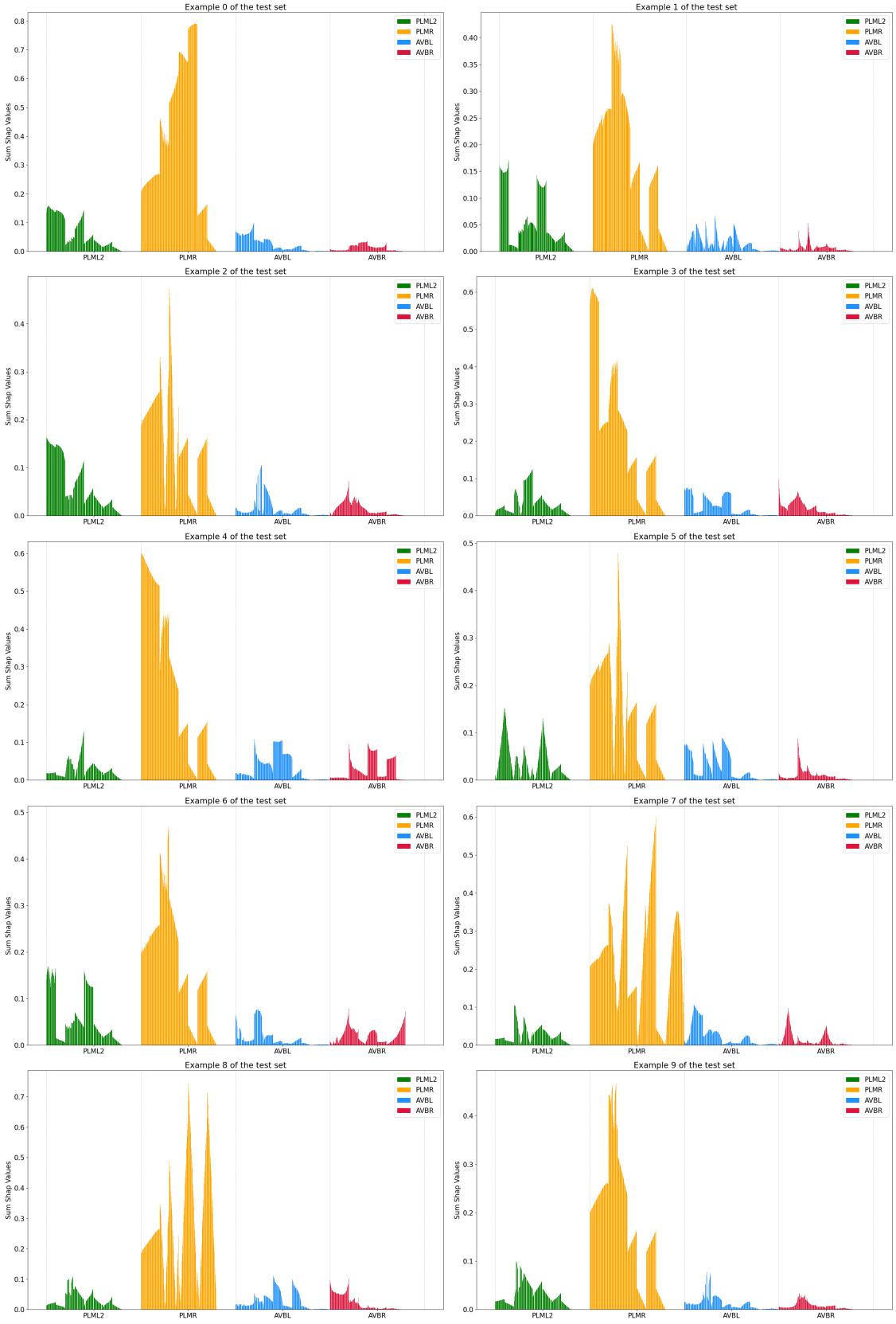

Figure A.11: Sum in absolute values of the SHAP values of each input feature through the 1000 time steps for the neuron PVR for all of the examples of the test set of **FCM-20**, ordered from left to right and up-down.

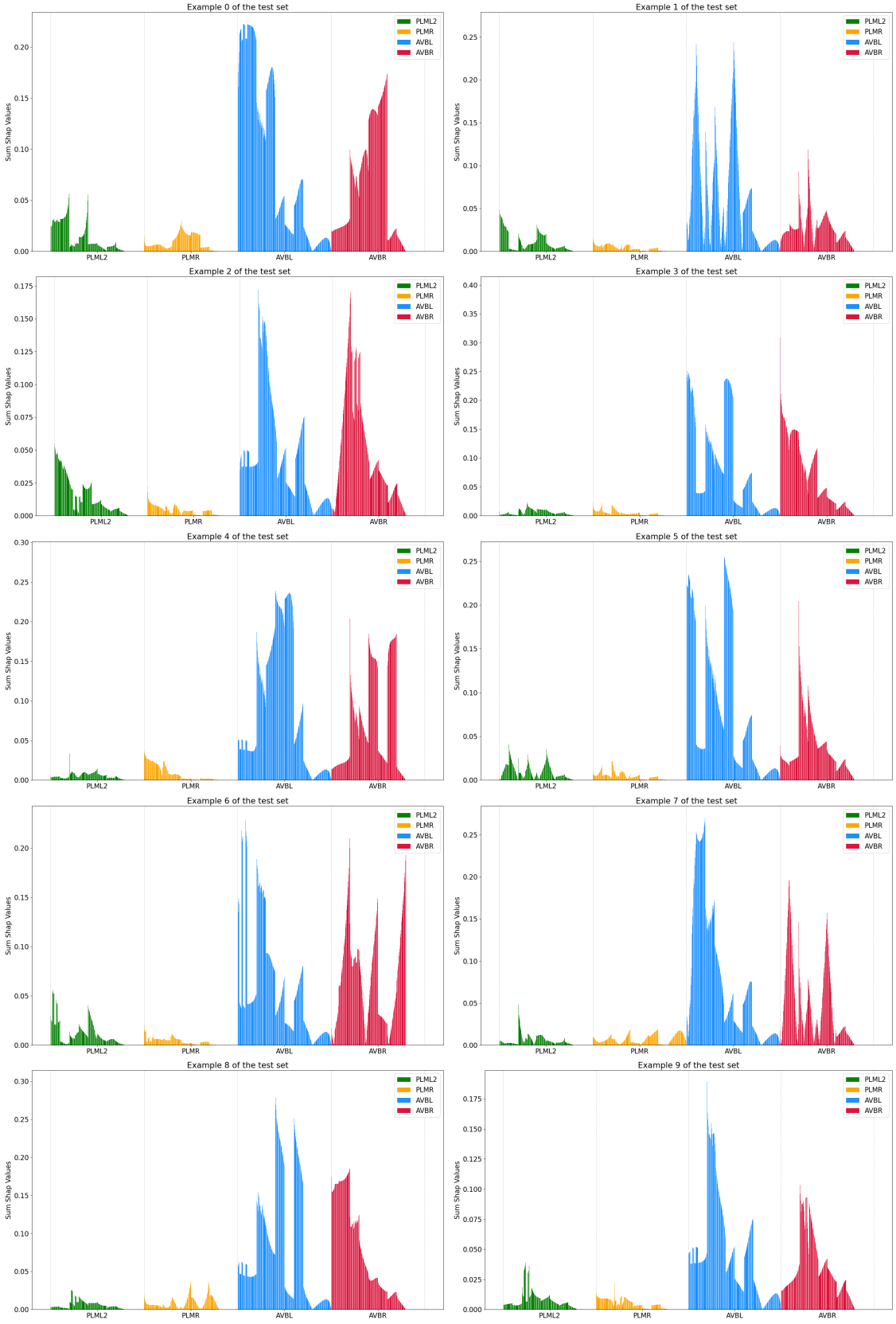

Figure A.12: Sum in absolute values of the SHAP values of each input feature through the 1000 time steps for the neuron VB1 for all of the examples of the test set of **FCM-20**, ordered from left to right and up-down.

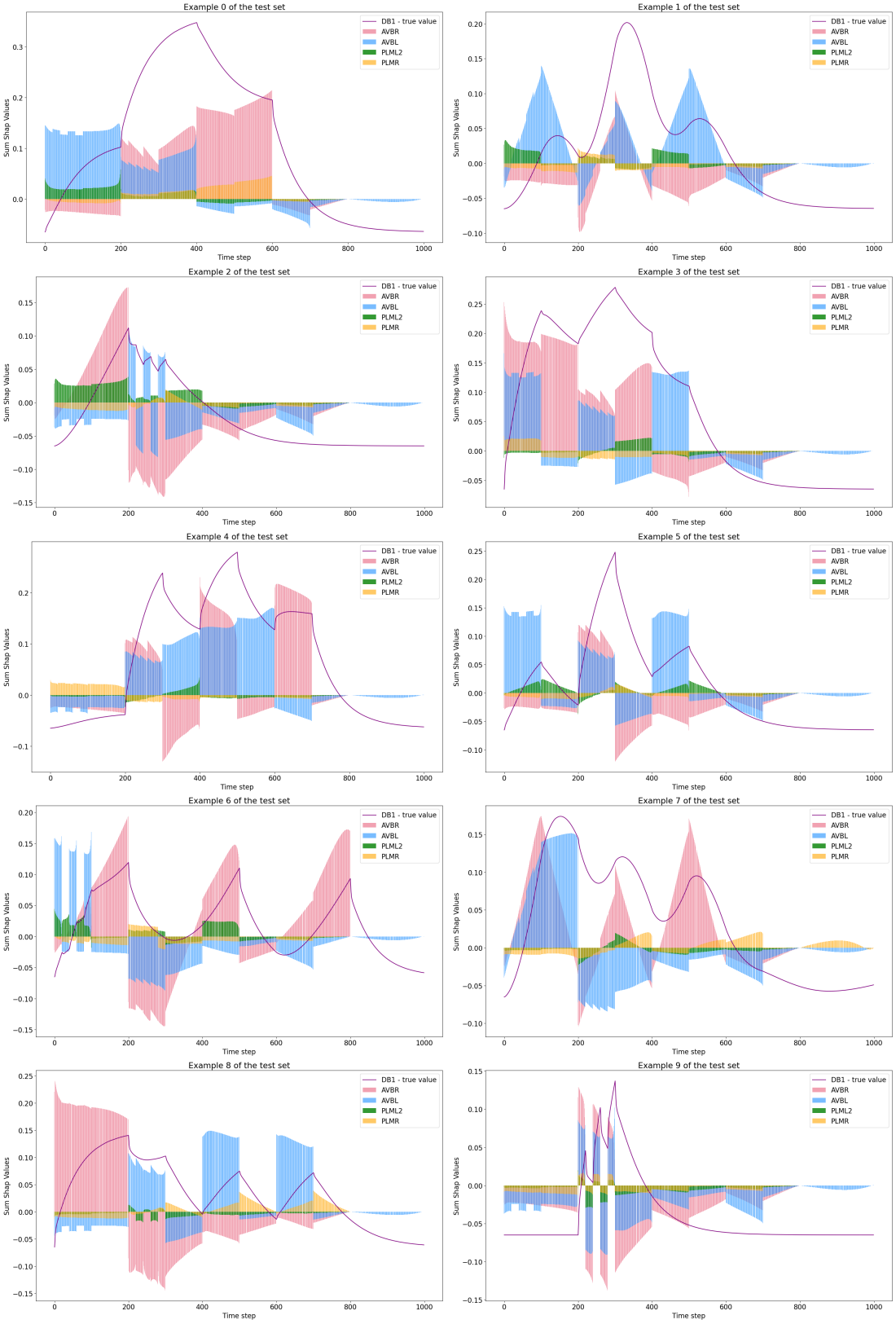

Figure A.13: Sum of the SHAP values of each input feature through the 1000 time steps overlapped within the same time span, together with the true value of the output feature, for the neuron DB1 for all of the examples of the test set of **FCM-20**, ordered from left to right and up-down.

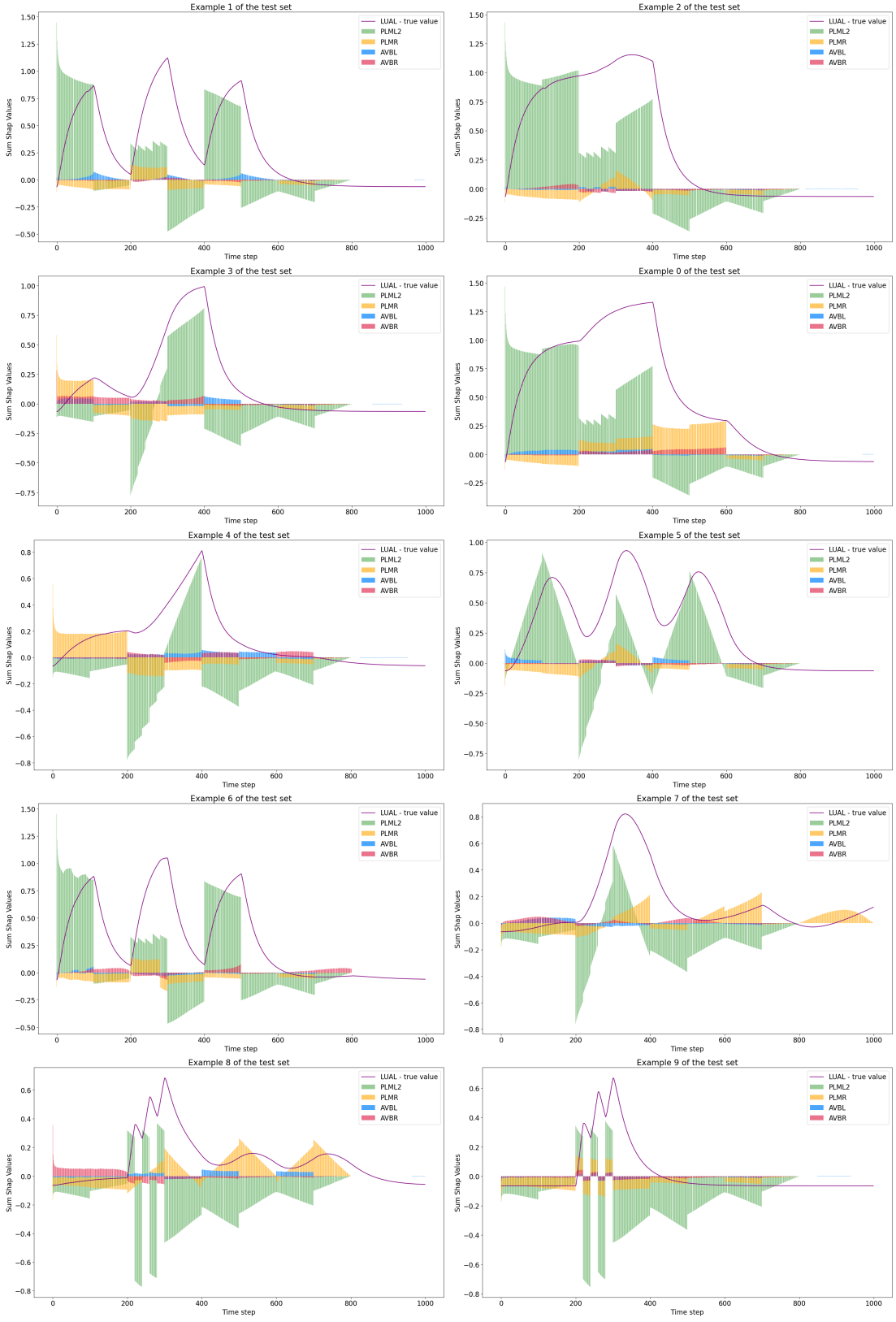

Figure A.14: Sum of the SHAP values of each input feature through the 1000 time steps overlapped within the same time span, together with the true value of the output feature, for the neuron LUAL for all of the examples of the test set of **FCM-20**, ordered from left to right and up-down.

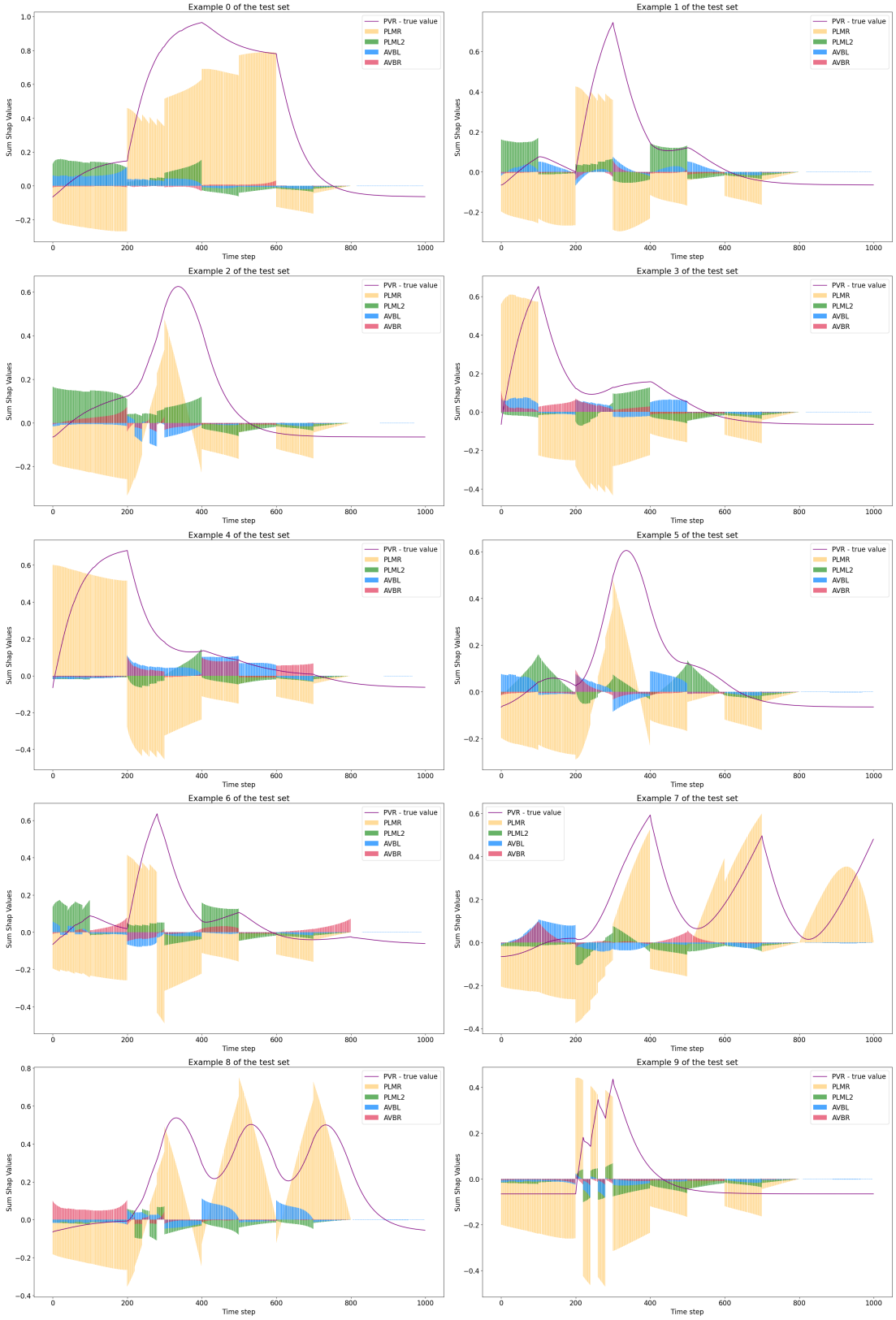

Figure A.15: Sum of the SHAP values of each input feature through the 1000 time steps overlapped within the same time span, together with the true value of the output feature, for the neuron PVR for all of the examples of the test set of **FCM-20**, ordered from left to right and up-down.

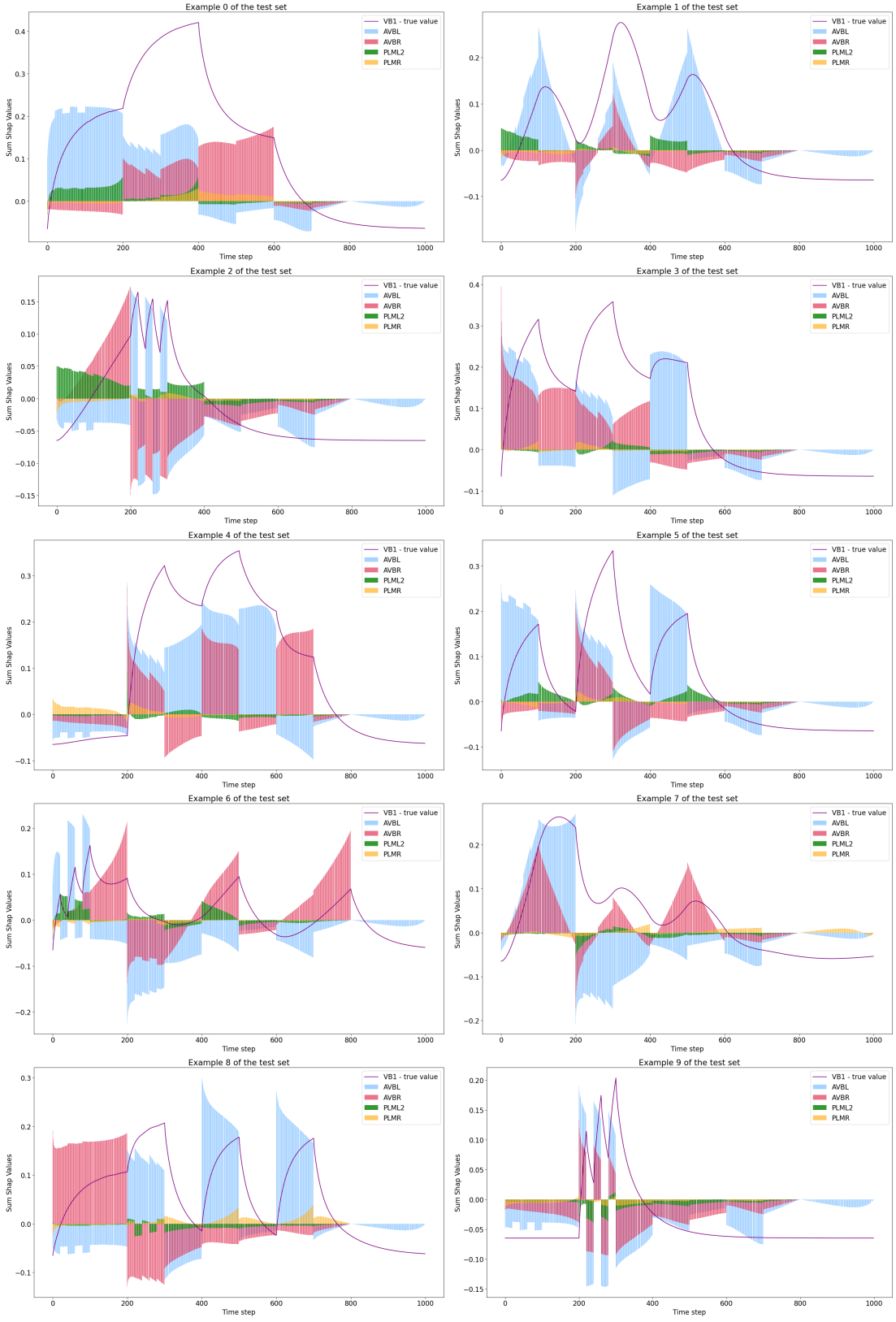

Figure A.16: Sum of the SHAP values of each input feature through the 1000 time steps overlapped within the same time span, together with the true value of the output feature, for the neuron VB1 for all of the examples of the test set of **FCM-20**, ordered from left to right and up-down.