COMILLAS
UNIVERSIDAD PONTIFICIA

ICAI

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

PHD THESIS

EXPLAINABLE ARTIFICIAL INTELLIGENCE (XAI) TECHNIQUES BASED ON PARTIAL DERIVATIVES WITH APPLICATIONS TO NEURAL NETWORKS

Author: Jaime Pizarroso Gonzalo
Supervisors:
Antonio Muñoz San Roque
José Portela González

Madrid
June 2023

A mis padres

Contents

List of Figures

List of Tables

Abstract

As Machine Learning (ML) and Deep Learning (DL) models continue to permeate various aspects of society, there is an increasing demand for interpretability and transparency in their decision-making processes. This demand is fueled by the need to understand, trust, and effectively use these complex, black-box models, particularly in high-stake applications where decisions can have far-reaching consequences. Furthermore, the advancement of interpretability techniques is critical to adhere to the emerging ethical and legal requirements concerning the use of Artificial Intelligence (AI) systems.

Explainable Artificial Intelligence (XAI) has emerged as a promising solution to the opacity of complex models, offering techniques to make their decision-making processes understandable and transparent. Nevertheless, most existing XAI techniques face limitations concerning assumptions on data relationships, computational cost, the trade-off between interpretability and accuracy, and their ability to provide local and global explanations. To address these issues, this thesis introduces novel XAI methods based onpartial derivatives. Unlike existing methods, these techniques provide detailed, local to global level explanations without making assumptions about the relationships between inputs and outputs.

The main contributions of this thesis reside in three newly developed methods: Sensitivity Analysis, α-curves, and the application of the Interaction Invariant designed in Alfaya *et al.* (2023) to ML models, all of which leverage the partial derivatives to offer interpretability of differentiable ML models. Sensitivity Analysis estimates the influence of input variables on the ML model output, offering insights into the most impactful variables. α-curves provide a detailed view of sensitivity variation across the input space, assisting in identifying average and localized high-sensitivity regions. Lastly, Interaction Invariant focuses on detecting interactions between input variables, revealing complex relationships within the data that may influence the model's decision-making process. Collectively, these methods offer a comprehensive understanding of ML models, enhancing the transparency and trustworthiness of AI systems.

The utility and effectiveness of these methods were validated through three real-world use cases including predicting NOx emissions, parkinson disease progression, and turbofan engine Remaining Useful Life (RUL). These applications illustrated how the developed methods can reveal nuanced insights into model behavior, surpassing commonly used XAI techniques by providing coherent and relevant information about the inner workings of the models.

Resumen

A medida que los modelos de Aprendizaje Automático (ML, por su nombre en inglés *Machine Learning*) y Aprendizaje Profundo (DL, por su nombre en inglés *Deep Learning*) continúan permeando diversos aspectos de la sociedad, existe una creciente demanda de interpretabilidad y transparencia en sus procesos de toma de decisiones. Esta demanda está alimentada por la necesidad de comprender, confiar y utilizar eficazmente estos complejos modelos de caja negra, particularmente en aplicaciones de alto riesgo donde las decisiones pueden tener consecuencias de gran alcance. Además, el avance de las técnicas de interpretabilidad es fundamental para adherirse a los emergentes requisitos éticos y legales concernientes al uso de los sistemas de Inteligencia Artificial (IA).

La Inteligencia Artificial Explicable (XAI, por su nombre en inglés *Explainable Artificial Intelligence*) ha surgido como una solución a la opacidad de los modelos complejos, ofreciendo técnicas para hacer comprensibles y transparentes estos modelos. Sin embargo, la mayoría de las técnicas existentes de XAI enfrentan limitaciones con respecto a las suposiciones que deben hacer sobre las relaciones entre los datos, el costo computacional, la compensación entre la interpretabilidad y la precisión, o su capacidad para proporcionar explicaciones locales y globales. Para abordar estos problemas, esta tesis introduce nuevos métodos de XAI basados en derivadas parciales. A diferencia de los métodos existentes, estas técnicas proporcionan explicaciones detalladas, desde un nivel local hasta global, sin hacer suposiciones sobre las relaciones entre las entradas y las salidas.

Las principales contribuciones de esta tesis residen en tres métodos recién desarrollados: Análisis de Sensibilidad, curvas$-\alpha$ y la aplicación del Invariante de Interacción diseñado en Alfaya et al. (2023) a modelos de ML, todos los cuales aprovechan las derivadas parciales para ofrecer interpretabilidad de los modelos de ML diferenciables. El Análisis de Sensibilidad estima la influencia de las variables de entrada en la salida del MLP, ofreciendo información sobre las variables más importantes. Las curvas$-\alpha$ proporcionan una visión detallada de la variación de la sensibilidad a través del espacio de entrada, ayudando a identificar regiones localizadas de alta sensibilidad. Por último, el Invariante de Interacción se centra en la detección de interacciones entre las variables de entrada, revelando relaciones complejas en los datos que pueden influir en la predicción del modelo. En conjunto, estos métodos ofrecen una comprensión integral de los modelos de Aprendizaje Automático, mejorando la transparencia de los sistemas de IA.

La utilidad y efectividad de estos métodos se validaron a través de tres casos de uso del mundo real, que incluyen la predicción de emisiones de NOx, la progresión de la enfermedad de Parkinson y la vida útil restante de motores turbofan. Estas aplicaciones evidenciaron cómo los métodos desarrollados pueden mostrar información detallada sobre el comportamiento del modelo, superando las técnicas más utilizadas de explicabilidad de IA al proporcionar información coherente y relevante sobre el funcionamiento interno de los modelos.

1 Introduction

When you are inspired by some great purpose, some extraordinary project, all your thoughts break their bonds.
Patanjali (1^{th} Century BC)

This first chapter introduces the rationale behind this thesis as well as its main objectives. In addition, it provides the reader with a general overview of the organization and the outline of the dissertation in order to make it easier to follow.

1.1. Motivation

In recent years, Machine Learning (ML) and Deep Learning (DL) have gained significant attention in the field of Artificial Intelligence (AI). ML, as a subfield of AI, focuses on developing algorithms that can learn patterns from data without explicit programming. It encompasses various approaches such as supervised learning, unsupervised learning, and reinforcement learning (Goodfellow *et al.*, 2016). Deep Learning (DL), a subset of ML, utilizes artificial neural networks (ANNs) to mimic the learning abilities of the human brain, enabling the modeling of complex relationships and achieving remarkable performance in various domains (LeCun *et al.*, 1998). The success of ML and DL models has led to their widespread adoption across different domains, demonstrating superior performance in tasks such as object detection in images (Ghasemi *et al.*, 2022), natural language processing (Lauriola *et al.*, 2022), or predictive maintenance (Serradilla *et al.*, 2022). The primary allure of NNs lies in their ability to learn and model intricate patterns and nonlinear relationships in large and high-dimensional datasets, thereby offering superior predictive performance. NNs, particularly those with multiple hidden layers or Deep Neural Networks (DNNs), are notably powerful, exhibiting an unparalleled ability to perform highly sophisticated tasks (LeCun *et al.*, 1998).

The complexity that affords NNs their remarkable capabilities, however, also contributes to their principal limitation: the black box problem. Black box refers to systems where the internal workings are not understandable or interpretable by human users. In the context of NNs, this means that while we can input data and obtain outputs from the network, understanding the internal decision-making process—how the network arrived at a particular output from a given input—is not straightforward. This opacity is due to the intricate inner structure of the network. As of now, there is no definitive method to explain these models, and often several methods are required to understand the model inner processes. This lack of interpretability is a significant hurdle to their widespread use in many sectors, especially those where decision transparency is crucial, such as healthcare (Miotto *et al.*, 2017), finance (Mirestean *et al.*, 2021), and cybersecurity (Pereira and Thomas, 2020). The ability to interpret and understand the decision-making process of these models is paramount in building trust and allowing human users to appropriately use and manage AI systems.

Explainable Artificial Intelligence (XAI) has emerged as a vital research area aiming to address the interpretability challenge in ML and DL models. XAI techniques provide insights into the decision-making process of complex models, making them more transparent and interpretable. These techniques have been successfully applied for various applications in different fields. For example, in healthcare, XAI has been utilized to interpret the predictions of disease diagnosis models, facilitating doctors to understand the reasoning behind the model's diagnosis decisions, and thus, enhancing trust in these models (Caruana *et al.*, 2015). In finance, XAI techniques are used to elucidate credit scoring and financial fraud detection models. By providing understandable and intuitive explanations, these techniques assist financial institutions in understanding the reasoning behind credit decisions or identifying potential fraudulent transactions (van den Berg and Kuiper, 2020). As for cybersecurity, XAI techniques play a critical role in interpreting the decisions made by intrusion detection systems. They help security analysts to understand the rationale behind the alerts generated by these systems, facilitating the process of threat identification and mitigation (Srivastava *et al.*, 2022). In these ways, XAI techniques contribute to enhancing trust, improving transparency, and facilitating better decision-making across diverse domains.

While existing XAI methods have significantly improved the interpretability of ML and DL models, they face several limitations that hinder their effectiveness and broad application. The challenges associated with these techniques include high computational costs, inconsistency, a lack of robustness and difficulty in handling high-dimensional inputs (Molnar, 2022; Lipton, 2018). For example, local approximation methods such as LIME may produce varying explanations due to different perturbations used in the process (Ribeiro *et al.*, 2016). Additionally, they often grapple with the curse of dimensionality, failing to deliver meaningful interpretations for high-dimensional inputs or even unfeasible to be applied in large datasets (Molnar, 2022).

These persistent challenges often cost the practitioner the opportunity to use complex models, such as Neural Networks, in their business applications, often deriving to simpler models such as Linear Regression or Logistic Regression models. These models are highly interpretable and are already accepted by the regulators in critical sectors, but they lack the modeling capacities of NN models (Lipton, 2018). In this context, even designing new XAI methods to explain the simplest NN model, the Multi-Layer Perceptron (MLP), and overcoming existing limitations would be a valuable contribution. Explaining MLP models with enhanced XAI methods would facilitate their adoption in critical sectors by providing regulators with the interpretability they require without sacrificing the modeling power of NNs. Therefore, there is a need for new XAI methods that overcome these limitations while retaining the benefits of existing techniques. Based on the review of existing literature (Nielsen *et al.*, 2021; Arras *et al.*, 2022), the analysis of partial derivatives of the output variables with respect to the input variables of ML models offers a promising direction for new XAI methods. These methods, with their light-weight calculation, are expected to maximize the information retrieved from the model, thus improving both the quality and efficiency of the explanation process.

This dissertation, therefore, aims to contribute to the field of XAI by proposing novel methods based on partial derivatives that provide robust, coherent, and accurate explanations for ML and DL models. These techniques are designed to give detailed insights into the influence of individual input features on model predictions, thereby enhancing the interpretability of ML and DL models. By developing and employing these new XAI methods, we aim to outperform existing approaches in terms of explanation quality and applicability to real-world applications, without compromising the model's predictive performance.

1.2. Thesis objectives

The primary objective of this thesis is to contribute to the development of Explainable Artificial Intelligence (XAI) methods that can facilitate comprehensive understanding of Neural Network (NN) models, focusing primarily on Multi-Layer Perceptron (MLP) models. The choice of the MLP model is driven by its widespread use in diverse domains due to its ability to model complex, non-linear relationships, and the potential generalizability of the insights gained from this model to other types of NN models. Our research endeavors to derive explanations that provide nuanced insights into the relationship between inputs and outputs, reveal feature importance, and detect feature interactions in MLP models. The developed methods aim to offer both global and local interpretability, while ensuring computational feasibility, thus enabling a higher degree of transparency, trust, and informed decision-making in AI systems. Hence, the specific objectives pursued in this dissertation are summarized below:

- Development of analytical calculations of first, second and third partial derivatives of a commonly used NN model, the Multi-Layer Perceptron (MLP) model. This approach aims to reduce computational resources and time required, making the XAI methods more practical for real-time applications.

- Development and introduction of novel XAI methods based on partial derivatives that meet the key requirements outlined above. These methods will be designed to reveal intricate details about the input-output relationships, feature importance, and feature interactions in ML models. Special attention will be devoted to ensuring that these methods provide both global and local interpretability. The developed methods should address the need for more comprehensive, nuanced, and computationally efficient explanations in the domain of XAI.

- Validation of the proposed methods by applying it to different real-world datasets obtained from different fields. The performance of the proposed methods must be compared against other reference XAI methods found in the literature, highlighting the improvements made by the techniques developed in this thesis when applied to real-world problems.

1.3. Dissertation outline

This dissertation consists of six chapters including this first introductory one.

In Chapter 2, the focus is on thoroughly exploring the current landscape of Explainable Artificial Intelligence (XAI) techniques, with a specific emphasis on post-hoc explainability methods for Multi-Layer Perceptron models. This chapter not only aims to familiarize the reader with key concepts in XAI and highlight existing methods, but also provides a comprehensive understanding of the need for these techniques. By delving into the motivations behind XAI, the chapter elucidates the critical role played by explainability in addressing the black-box nature of Neural Network models. Moreover, it investigates the limitations of current techniques, thereby setting the stage for the proposal of novel approaches that are more efficient, scalable, and aligned with specific needs. Through a thorough exploration of the taxonomy of XAI and factors guiding the selection of appropriate techniques, this chapter lays the foundation for the subsequent chapters, which build upon the development of advanced methods in the field.

Diving into the main developments of the research, Chapter 3 aims to introduce and thoroughly explain the novel methods proposed in this thesis for enhancing the explainability of ML models, focusing on the MLP models. In this chapter, three new methodologies are comprehensively described: Sensitivity Analysis, $\alpha-$curves, and the application of the Interaction Invariant developed in Alfaya *et al.*, 2023 to ML models. By detailing each method, discussing its strengths, and acknowledging its limitations, the chapter seeks to equip the reader with tools that provide a more granular understanding of model behaviour. The use of synthetic datasets to validate these methods further anchors their practical applicability and effectiveness.

The goal of Chapter 4 is to illustrate the practical utility of the novel methods proposed in this thesis through real-world case studies. It seeks to validate the efficacy of these methods and demonstrate their advantage over traditional XAI techniques in three use cases: predicting NOx emissions in the Boston dataset, forecasting parkinson disease progression, and estimating the Remaining Useful Life (RUL) of turbofan engines using the CMAPSS dataset. The chapter aims to showcase how the developed methods—Sensitivity Analysis based on Partial Derivatives, $\alpha-$curves, and Interaction Invariant—can provide nuanced insights into model behavior, uncovering intricate relationships and feature interactions that commonly used methods may overlook. The overarching objective is to demonstrate that these novel methods can improve the understanding and interpretability of ML models in practical scenarios, enabling more informed and reliable decision-making.

Finally, Chapter 5 provides the concluding remarks of the dissertation, summarizing the contributions and future developments.

2 Explainable Artificial Intelligence (XAI) for Neural Networks: State-of-the-art

Give me six hours to chop down a tree
and I will spend the first four
sharpening the axe.
Abraham Lincoln (1842—1865)

This chapter explores Explainable Artificial Intelligence (XAI) in machine learning, with a focus on Multi-Layer Perceptron (MLP) models. It presents a comprehensive XAI taxonomy, different post-hoc explainability techniques, and an in-depth analysis of various XAI methods.

2.1. Introduction

Machine Learning (ML) and Deep Learning (DL) models have been successfully applied in several applications. Consequently, understanding the foundations upon which these models make predictions becomes a crucial aspect of AI solution development. Users must have confidence in the model's performance on real data, based on meaningful metrics; otherwise, the model should not be utilized (Ribeiro *et al.*, 2016). The process of interpreting how the model works is investigated in a research field called Explainable Artificial Intelligence.

Explainable Artificial Intelligence (XAI) refers to the development of Artificial Intelligence (AI) and ML models that generate transparent and interpretable results to enable humans to understand, trust, and manage the decisions made by these models (Gunning and Aha, 2019). With the increasing adoption of AI and ML systems in various domains, including healthcare, finance, and criminal justice, the demand for explainability has become more critical (Miller, 2019).

As AI and ML models become more complex and increasingly integrated into critical decision-making processes, the need for XAI techniques continues to grow. This section collects the result of an extensive review of the state of the art of XAI methods for NN. Along the section, the terms "Artificial Intelligence Model" and "Machine Learning Model" are used indistinguisable in order to provide a more comprehensive reading. However, it must be noted that the literature differs in the definition of both "Artificial Intelligence" and "Machine Learning", being the latter an application of the former (Ghahramani, 2015).

2.2. Introduction to Machine Learning

Artificial Intelligence (AI) is a branch of computer science that focuses on the creation of intelligent agents or systems that have the ability to perform tasks that usually require human intelligence (Russell and Norvig, 2009). This encompasses capabilities such as learning, reasoning, problem-solving, perception, and language understanding. Machine Learning (ML), a subset of AI, is defined as the study of computer algorithms that improve automatically through experience (Mitchell, 1997). Unlike traditional algorithms or Expert Systems (Jackson, 1986) that follow a determinisaatic set of hand-coded rules, Machine Learning algorithms learn patterns from data and make predictions or decisions based on those patterns.

Machine Learning encompasses three main types of learning:

- **Supervised Learning**: This approach is guided by a known set of output values. It includes tasks such as:
  - Regression: In regression, the goal is to predict a continuous output variable. Examples include predicting housing prices (Kaggle, 2016) or stock prices (Akita *et al.*, 2016).
  - Classification: In classification, the goal is to predict a categorical output variable. Examples include predicting whether an email is spam or not (Metsis *et al.*, 2006), or whether a patient has a certain disease or not (Esteva *et al.*, 2017).
  - Forecasting: In forecasting, historical data is used to predict future trends or events. Examples include predicting electrical demand (Hong and Fan, 2016) or weather patterns (McGovern *et al.*, 2017).

  Some of the most common supervised learning algorithms are Linear Regression (Bishop and Nasrabadi, 2006), Decision Trees (Quinlan, 1986), and Support Vector Machines (SVM) (Cortes and Vapnik, 1995).

- **Unsupervised Learning**: This method discovers the inherent structure in the data without any explicit output guidance. It includes tasks such as:
  - Clustering: In clustering, the goal is to group similar instances together. Examples include grouping customers based on their purchasing behavior (Irani *et al.*, 2016) or grouping documents based on their topics (Blei *et al.*, 2003).
  - Dimensionality reduction: In dimensionality reduction, the goal is to reduce the number of features in the data while retaining as much information as possible. Examples include Principal Component Analysis (PCA) (Bro and Smilde, 2014) and t-distributed Stochastic Neighbor Embedding (t-SNE) (Van der Maaten and Hinton, 2008).
  - Anomaly detection: In anomaly detection, the goal is to identify instances that are significantly different from the rest of the data. Examples include detecting

fraudulent transactions (Maniraj *et al.*, 2019) or detecting faulty equipment in a manufacturing process (J. Liu *et al.*, 2018).

Algorithms like K-means (MacQueen, 1967), Hierarchical Clustering (Sibson, 1973), and Principal Component Analysis (PCA) (Bro and Smilde, 2014) are frequently used in unsupervised learning.

- **Reinforcement Learning**: In this paradigm, the agent takes actions in an environment and receives feedback in the form of rewards or penalties. Reinforcement learning has been successfully applied in domains such as robotics, game playing, and recommendation systems. One notable example is AlphaGo, a computer program developed by DeepMind that used reinforcement learning to defeat the world champion in the ancient Chinese board game Go (Sutton and Barto, 2018; Mnih *et al.*, 2015).

Among the various models used in Machine Learning, Artificial Neural Network (ANN) models stand out as one of the most prominent and widely recognized (Goodfellow *et al.*, 2016). An ANN is composed of layers of interconnected nodes, also known as neurons. Each neuron receives input from other neurons, processes the input using a mathematical function called an activation function, and produces an output that is passed on to other neurons in the next layer. The weights associated with the connections between neurons are adjusted during the learning phase of the model to optimize the network's performance on a given task. One of the key features of ANNs is their ability to approximate complex functions, including non-linear ones, enabling end-to-end learning and superior performance on complex tasks. This is possible because ANNs can learn non-linear transformations of the input data through the composition of multiple non-linear activation functions. The Universal Approximation Theorem, first proven by George Cybenko in 1989, states that a feedforward neural network with a single hidden layer containing a sufficient number of neurons can approximate any continuous function on a compact subset of Euclidean space to arbitrary accuracy (Cybenko, 1989). Moreover, several studies have investigated the approximation power of ANNs, including the recent work by (Montufar *et al.*, 2014), which showed that deep rectifier networks can approximate any continuous function on a compact subset of Euclidean space to arbitrary accuracy, regardless of the number of hidden layers or neurons. Other researchers have explored the relationship between the depth of the network and its approximation properties, with some suggesting that deeper networks may have advantages over shallower ones (Bengio *et al.*, 2013).

One of the most common ANN architectures is the Multilayer Perceptron (MLP) model. A MLP is a feedforward neural network composed of multiple layers of interconnected perceptron units. It consists of an input layer that receives the input data, one or more hidden layers that apply non-linear transformations to the inputs, and an output layer that produces the final output. Despite the advent of more complex neural network architectures for specialized tasks (such as Convolutional Neural Networks (CNNs) (Yamashita *et al.*, 2018) for image data or Recurrent Neural Networks (RNNs) (Sherstinsky, 2018) for sequential data), MLP remains highly relevant due to its simplicity, effectiveness, and wide applicability (Car *et al.*, 2020; Desai and Shah, 2021; Xu *et al.*, 2022). This is even more relevant in tabular data, where the MLP architecture requires less computational resources to obtain an optimal model while retaining a comparable level of accuracy compared to other DL architectures (Brownlee, 2022; Borisov *et al.*, 2022). MLP models trained on tabular data find applications in numerous fields, such as predict electrical demand (Lorencin *et al.*, 2019a) or predictive maintenance (Lorencin *et al.*, 2019b).

Despite the successful implementation of this type of model in the previously mentioned fields, Explainable Artificial Intelligence (XAI) in the context of ANNs is particularly needed due to the inherent opacity and black-box nature of these models. This is specially relevant when using ANNs in critical applications such as credit scoring (Pandey *et al.*, 2017; Riffi *et al.*, 2020), where it is crucial to understand the factors that contribute to the model's predictions and ensure transparency and fairness in the decision-making processes. Therefore, interpretability is an indispensable property for these systems to be reliably integrated into decision-making processes.

2.2.1. Interpretability vs Performance trade-off

In Machine Learning, the pursuit of high-performance predictive models often leads to models that are complex and opaque. These models, although achieving excellent predictive accuracy, can be described as *black boxes* due to their inherently complex and non-transparent structures. On the other hand, simpler models with lower predictive performance often allow for easier interpretability (see section 2.3.1).

The compromise between model performance and interpretability is a common dilemma in the field of Machine Learning. On one hand, we have models like linear regression and decision trees which are straightforward to understand and interpret but may not yield the best performance. On the other hand, we have highly accurate models like deep neural networks, which are notorious for their lack of transparency and difficulty to interpret.

The dilemma between model performance and interpretability aligns with the bias-variance trade-off, another central concept in Machine Learning. Bias refers to the simplifying assumptions made by a model that can result in error due to incorrect assumptions in the learning algorithm. High-bias models, such as linear regression, may overlook the nuanced relations between features and target outputs, leading to underfitting and suboptimal performance. However, these models are often easier to interpret because of their simplicity.

Variance, on the other hand, arises from a model's sensitivity to small fluctuations in the training set. High-variance models tend to capture even the noise from the training data, leading to overfitting. While these models typically perform well on the training data, they may not generalize well to new, unseen data. Complex models, such as deep neural networks, are prone to overfitting due to their capacity to capture intricate relationships in data, which in turn makes them harder to interpret.

Thus, the quest for high performing and interpretable models is intrinsically tied to navigating the bias-variance trade-off successfully.

Figure 2.1 shows where are different ML models in the interpretability vs performance trade-off. The X-axis represents the interpretability of the model and the Y-axis represents the performance of the model. As we move towards more complex models, the interpretability decreases while the performance increases.

Understanding and navigating this trade-off is crucial. However, the importance of interpretability versus performance can vary depending on the application at hand. For tasks where the consequences are significant, such as medical diagnoses, financial risk assessment, or any other decisions where human lives and well-being are at stake, the need for interpretability and transparency might outweigh the desire for the highest possible predictive performance.

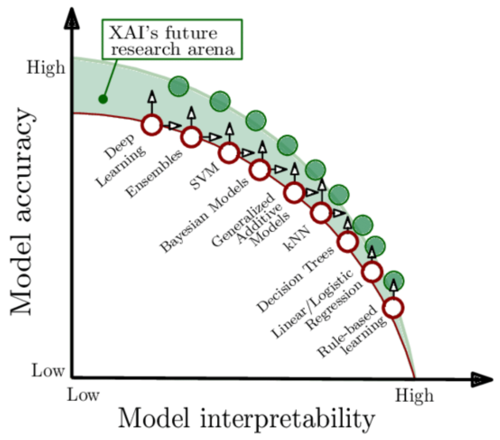


Figure 2.1. Interpretability vs Performance Trade-off. Source: (Arrieta *et al.*, 2020)

Meanwhile, in applications where the consequences of errors are less severe and high predictive performance is paramount, the balance may swing towards more complex, less interpretable models.

The challenge is to find the right balance between interpretability and performance. One potential approach is to create models that are both performant and interpretable, though this remains an active area of research. The ultimate goal is to achieve trustworthy AI, where the model's decisions are both highly accurate and understandable to humans.

XAI methods strive to bridge the gap between performance and interpretability. They aim to make complex models more understandable, which allows for a decrease in the opacity of models without compromising their performance. With the help of XAI, we might be able to mitigate the trade-off, moving towards models that offer both high performance and interpretability.

2.2.2. Importance of XAI

The importance of model accountability has been underscored in numerous contexts, with various incidents highlighting the repercussions of AI decision-making processes not being fully understood or interpretable (Diakopoulos, 2016; Miller, 2019). These incidents have underscored the need for robust strategies to rectify such issues, thereby ensuring the ethical, safe, and accountable application of AI (Mittelstadt *et al.*, 2019).

Understanding the basis of a model's decisions is essential to ensure its reliability and effectiveness. Blindly trusting models can lead to incorrect or undesirable outcomes, as they may base their decisions on irrelevant features, noise, or background information (Selvaraju *et al.*, 2017). The key to addressing these challenges is developing methods for generating explanations, which can help uncover the reasoning behind a model's decisions (Gunning, 2017). By making AI systems more explainable, we can improve their trustworthiness, reliability, and usability in a wide range of applications (Holzinger, 2016). This leads us to the importance of Explainable Artificial Intelligence, which aims to address these concerns and make AI systems accountable.

As AI and ML models become more integrated into various aspects of our lives, ensuring their decisions are interpretable is crucial. XAI techniques play a key role in order to achieve the following goals:

1. **Enhancing trust and user acceptance:** By offering comprehensible explanations for AI decisions, users' trust in these systems is improved. This elevated trust not only propels wider adoption of AI technology but also facilitates its utilization across a myriad of domains (Mittelstadt *et al.*, 2019). For instance, when a medical diagnosis AI system provides a result, if it can also explain its reasoning in a manner that a doctor can understand, the doctor is more likely to trust and use the system in clinical settings.

2. **Promoting Effective Human-AI Collaboration:** XAI serves as an interface that encourages efficient collaboration between human experts and AI systems. By offering insights into each other's reasoning processes, it cultivates a mutual understanding, leading to improved decision-making, particularly in intricate domains (Gunning, 2017). In the finance field, an AI system that predicts stock market trends can collaborate more effectively with financial analysts if it can elucidate the factors influencing its predictions, allowing analysts to combine their domain expertise with the AI's insights.

3. **Guaranteeing Regulatory Compliance:** Legal frameworks, such as the European Union's General Data Protection Regulation (GDPR), mandate AI systems to provide explanations for their decisions, especially when impacting individuals. XAI techniques can facilitate compliance with such regulations, helping AI systems to legally operate within stipulated boundaries (Wachter *et al.*, 2017). For example, in online lending platforms, if an AI system denies a loan application, the GDPR mandates that the applicant has the right to know the reason. XAI can provide clear explanations, ensuring that the platform remains compliant with regulatory requirements.

4. **Debugging and Validation of AI Models:** Explainable AI serves as a critical tool in identifying, understanding, and correcting errors or biases in AI models. By offering insights into the underlying model logic, it can contribute to the development of superior-performing, more accurate, and equitable models (Ribeiro *et al.*, 2016). To illustrate this point, consider a facial recognition system. If it consistently misclassifies individuals from a particular ethnic background, XAI can help pinpoint the reasons for this bias, allowing developers to address and rectify the issue.

5. **Reliability / Robustness:** The goal is to ensure that minor variations in input variables do not drastically influence the prediction output. XAI strives to assure that ML systems are resilient against issues such as noisy inputs, domain shifts, and adversarial attacks, thereby promoting robustness and reliability (Alvarez Melis and Jaakkola, 2018). As an example, in autonomous driving, it's vital that slight changes in environmental conditions (like light or rain) don't drastically alter the AI's decisions. XAI can help in understanding and ensuring the model's consistent performance across varied conditions.

These goals may not be shared by the ML systems being interpreted. For example, in health care, the end goal of the ML system might be to predict accurately if a given patient has a specific illness. This end goal might be in conflict with generating interpretable decissions. Highly interpretable models, such as linear regression and decision trees, are generally easier to understand and explain; however, they may not always provide the best performance, especially in complex tasks with high-dimensional data (Molnar, 2022). On the other hand, more complex models, like Deep Learning, can achieve state-of-the-art performance in various domains, but their decision-making process is often difficult to interpret (Goodfellow *et al.*, 2016).

This trade-off between interpretability and performance raises an important dilemma: should we prioritize the development of highly accurate models, even if they are less transparent and harder to understand? Or should we focus on more transparent models, despite the potential loss in performance? This dilemma is particularly relevant in critical applications, such as healthcare, finance, and criminal justice, where the consequences of incorrect decisions can be severe and far-reaching (Arrieta *et al.*, 2020). Consider the domain of criminal justice, where an ML model might be used to assess the likelihood of recidivism among inmates. A complex machine learning model might promise unprecedented predictive accuracy, aiding judges in making informed decisions about parole and sentencing. However, if this model operates as a "black box" without XAI techniques, it could inadvertently introduce gender or race bias into its predictions. Without transparent explanations of its decision process, biased outcomes might go undetected, perpetuating unjust disparities in the justice system. This underscores the urgent need for interpretable AI to ensure not only accuracy but also fairness and accountability in critical decision-making processes.

2.2.3. Difference between Transparency, Explainability and Interpretability

Transparency, explainability, and interpretability are closely related concepts in the context of XAI. Although the three terms are often used interchangeably in the context of Artificial Intelligence and Machine Learning, they refer to distinct concepts with different implications for the understanding and trustworthiness of AI systems:

- **Transparency** refers to the extent to which the internal mechanisms and processes of a model are open and visible to inspection. A transparent model provides clear visibility into its architecture, parameters, and calculations (Molnar, 2022).
- **Explainability** focuses on providing the reasoning for the predictions or decisions made by a model. It involves the ability to articulate why a particular outcome was produced by the model and to present the factors or features that influenced that outcome (Arrieta *et al.*, 2020). For a model to be explainable, it must be able to provide explanation for its predictions or an additional method can be applied to retrieve these explanations.
- **Interpretability** relates to the ease with which a model's predictions or behaviors can be understood and interpreted by humans (Molnar, 2022). An interpretable model enables users to grasp the underlying logic, relationships, or patterns that the model has learned from the data.

The three terms are crucial aspects of AI systems, as they help users build trust, validate model decisions, and ensure the responsible and ethical use of AI technologies across various domains. Let's consider an example using a medical diagnosis AI system that uses a complex Deep Learning model to predict the probability of a patient having a certain disease based on various health indicators.

In terms of transparency, it pertains to how comprehensible the internal mechanisms and operations of the model are to external scrutiny. A transparent model would offer insights into its intricate architecture, reveal the values of its parameters, and elucidate the computational steps it undertakes. However, due to the complexity of the Deep Learning model, direct access to its parameters or operations might not inherently provide valuable interpretive information to users.

When it comes to explainability, the Deep Learning model should not merely produce predictions, but also offer clear reasoning for those predictions. For instance, if the system predicts a high probability of a patient having a certain disease, explainability would involve articulating why this prediction was made. This might involve highlighting that the prediction was influenced by a peak glucose level of 130 mg/dL or emphasizing the significance of triglyceride levels as a key contributing factor.

The concept of interpretability within the context of the Deep Learning model concerns the degree to which users can comprehend the model's predictions and underlying patterns. An interpretable model enables users to grasp the rationale behind predictions and ubderstabd how specific health indicators contribute to the outcome. In the context of our medical diagnosis AI system, interpretability might involve users understanding why a glucose level of 130 mg/dL indicates a higher probability of the disease, or comprehending how the model has learned to relate triglyceride levels to disease progression.

XAI methodologies have an important role in addressing the challenges of transparency, explainability, and interpretability in ML models. Regarding transparency, XAI approaches can retrieve information about the inner workings of ML models or design new inherent transparent model architectures with greater prediction capabilities than current transparent models. When it comes to explainability, XAI methods can generate justifications for the model's outputs, enabling users to analyze the model behavior. Lastly, regarding interpretability, XAI techniques can facilitate the translation of model outputs into a more intuitive and user-friendly format. By simplifying the prediction results or providing visualizations, they help users, even those without technical backgrounds, to comprehend the results and the model's behavior more easily.

2.3. XAI taxonomy

In this section, we delve into the diverse landscape of Explainable Artificial Intelligence (XAI) by exploring its rich array of taxonomies found in the state of the art. As the need for interpretability in AI systems becomes increasingly significant, numerous taxonomies have emerged (Miller, 2019; Arrieta *et al.*, 2020; Speith, 2022; Chuang *et al.*, 2023), each offering a unique perspective on the classification and categorization of XAI methods. These taxonomies provide valuable frameworks for understanding and organizing the wide range of methods and techniques available under the XAI umbrella. By examining and synthesizing the existing taxonomies, this research aims to establish a comprehensive understanding of the different dimensions and classifications within the field of XAI. This knowledge base serves as a solid foundation for selecting the most appropriate method or technique to address a specific problem, effectively bridging the gap between the complexity of AI systems and the demand for interpretable explanations. XAI methods can be broadly categorized based on the following aspects:

1. **Model Transparency Level:** The level of transparency in models plays a crucial role in determining the interpretability of AI systems. Transparent models, such as linear regression or decision trees, inherently offer high interpretability, as the functional relationships they utilize to make predictions are readily understandable (Lipton, 2018). However, such models might lack sophistication required to model complex data patterns. Conversely, opaque models, typically associated with Deep Learning architectures, excel in capturing these complex patterns but do so in ways that are not easily interpretable due to their 'black-box' nature.

2. **Type of Data:** This criterion focuses on the type of data used by the AI model and how it influences the explainability of the model's decisions. It considers whether the AI

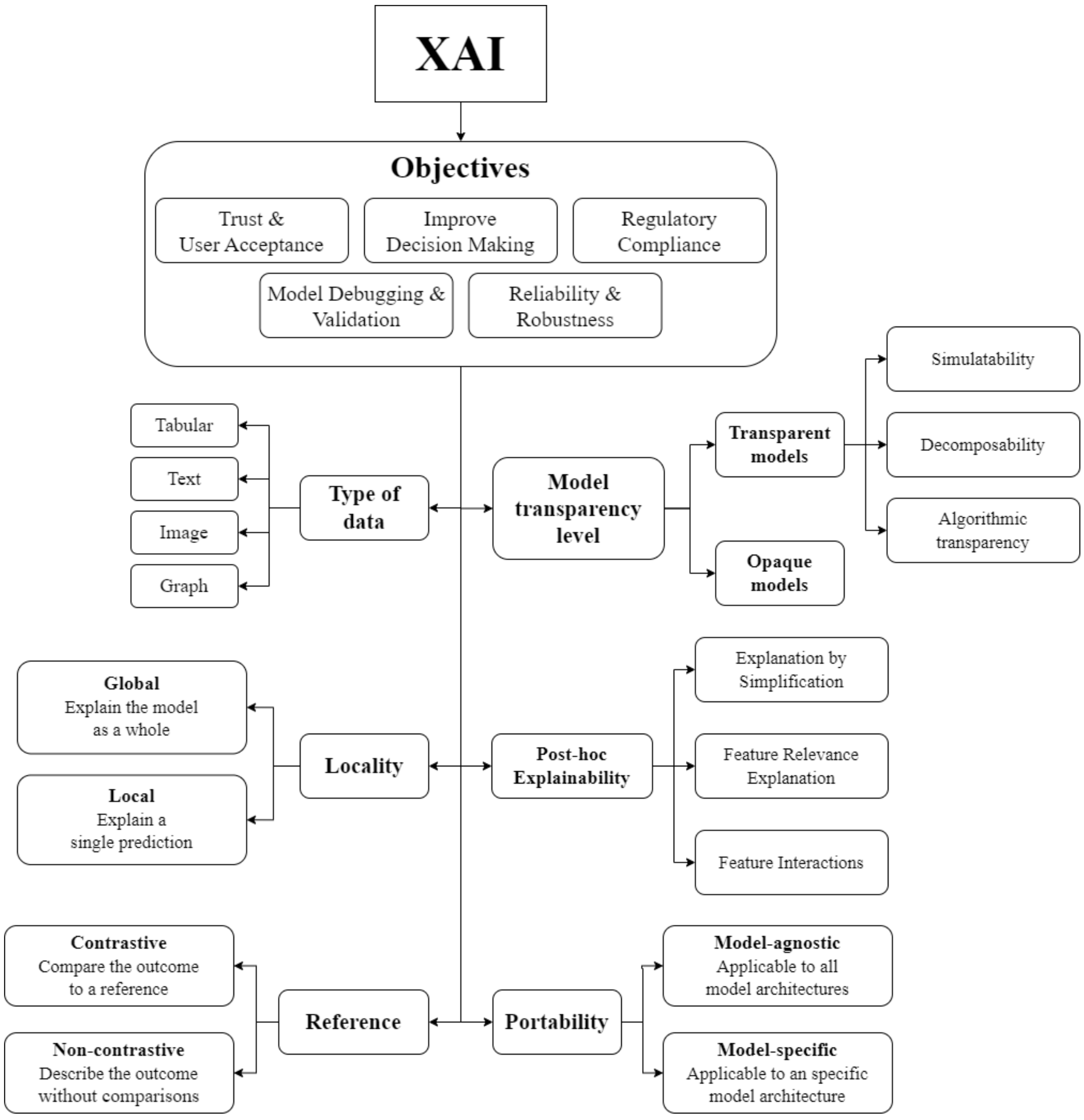


Figure 2.2. XAI objectives and taxonomy criteria described in this document.

system operates on structured data, such as tabular data with well-defined features, or unstructured data, including text, images, audio, or video. For structured data, techniques like Layer-Wise Relevance Propagation or Local Interpretable Model-Agnostic Explanations (LIME) can providing explanations for individual predictions (Rios *et al.*, 2020; Ribeiro *et al.*, 2016). For unstructured data, methods such as feature visualization or saliency-based approaches can highlight relevant regions or segments in the input data that contribute to the model's decision (Simonyan *et al.*, 2013; Olah *et al.*, 2017).

3. **Post-hoc Explainability:** this criteria refers to the retrospective analysis of a Machine Learning model's decision-making process after it has been trained. Some methods, like LIME, aim to create simpler models that approximate the local behavior of the complex ones. Others, like feature relevance explanation techniques, provide explanations by ranking features based on their importance in the decision-making process. Visual explanations, on the other hand, use visual aids to convey how a model arrived at its decisions (Ribeiro *et al.*, 2016).

4. **Portability:** Portability refers to whether an explainable Artificial Intelligence (XAI) method is tailored to a specific model or can be applied across different models. Model-specific techniques are designed for particular types of models and leverage their unique characteristics to generate explanations. For instance, DeepLIFT is an example of a model-specific technique that focuses on neural network models and examines neuron activation relative to a baseline value to provide explanations (Shrikumar *et al.*, 2017). On the other hand, model-agnostic techniques are not specific to any particular model type, offering greater flexibility but potentially providing less precise explanations (Lundberg and Lee, 2017). Partial Dependence Plots (PDP) is a widely used model-agnostic technique that analyze how the model's output responds to variations in a single input (Friedman, 2001), without information about the model's inner processes.

5. **Locality:** XAI techniques can provide either local or global explanations. Local explanations aim to clarify the predictions made for individual instances, as done by methods like anchors (Ribeiro *et al.*, 2018). Global explanations, on the other hand, aim to explain the model's overall behavior, as done by techniques such as permutation importance (Altmann *et al.*, 2010).

6. **Reference:** XAI methods can be contrastive, providing explanations in relation to a specific reference point, or non-contrastive, providing standalone explanations. DeepLIFT, for instance, is a contrastive method as it provides explanations by comparing the neuron activation with a baseline value (Shrikumar *et al.*, 2017). LIME, on the other hand, provide explanations for specific samples without the need of a reference point (Ribeiro *et al.*, 2016).

In the following sections, a detailed explanation is provided for each of the criteria listed.

2.3.1. Transparent vs. Opaque models

Transparent models offer comprehensible information of their inner workings and the decision-making processes by design. Linear regression, decision trees, and rule-based systems exemplify such transparent models, with their calculations and operations being easy to trace and understand (Lipton, 2018). On the other hand, opaque models, commonly complex Machine Learning algorithms like deep neural networks, are more challenging to interpret. The decision-making process in these models is intricate and not directly observable, thereby forming a so-called "black box" (Ribeiro *et al.*, 2016). While they often achieve superior performance, the lack of transparency in these models necessitates the use of post-hoc explainability techniques to understand and interpret their decisions.

2.3.1.1. Transparent models: Level of transparency

In the context of XAI, transparent models exhibit different degrees of interpretability. These degrees can be categorized into three levels: simulatability, decomposability, and algorithmic transparency (Doshi-Velez and Kim, 2017; Arrieta *et al.*, 2020). Each level provides distinct insights into the model's inner workings, aiding in understanding and interpreting the model's decisions.

1. **Simulatability**: Simulatability refers to the ability of a human to mentally simulate the steps and calculations performed by the algorithm, thereby understanding how the model arrives at its output (Doshi-Velez and Kim, 2017). In this category falls, for example,

shallow decision trees and simple rule-based systems. For instance, consider a decision tree model used to predict whether a person would go for a walk based on weather conditions, including Outlook, Humidity, and Wind. The decision tree can be represented as follows:

```
If Outlook = Sunny
    If Humidity <= 75
        Decision: Yes (Go for a walk)
    If Humidity > 75
        Decision: No (Don't go for a walk)

If Outlook = Rainy
    If Wind = Weak
        Decision: Yes (Go for a walk)
    If Wind = Strong
        Decision: No (Don't go for a walk)
```

In this example, we observe that the decision tree model predicts a person would go for a walk if it is sunny and not too humid, or if it is rainy and there is weak wind. However, it is important to note that deep decision trees or extensive rule-based systems fall outside this category, as humans are unable to visualize the entire decision process of these complex models.

2. **Decomposability**: Decomposability refers to the ability to break down a model into its constituent parts or components, enabling separate explanations for each element (Molnar, 2022). In the context of decomposable models, the contribution of individual features, parameters, or components to the final output of the model can be clearly identified and comprehended. This granular understanding facilitates the detection of potential biases, issues, or areas for enhancement within the model's decision-making process. Generalized Additive Models (GAMs) serve as a notable example of decomposable models (Hastie and Tibshirani, 1987). GAMs are flexible regression models that allow for the modeling of non-linear relationships by combining multiple smooth functions of the input features. The individual smooth functions in a GAM provide explainable contributions to the model's predictions, enabling a breakdown of the model's behavior into comprehensible components. For instance, consider a GAM model trained using a dataset following the expression $y = 2 \cdot x_1 + \sin(x_2) + x_3 + \varepsilon$. Figure 2.3 shows plots illustrating the effects of individual predictors on the output in this GAM model. Analyzing these plots allows users to extract information on the relationship between the output and input variables, providing a breakdown of the "black-box" model's behavior into explainable components.

3. **Algorithmic Transparency**: Algorithmic transparency refers to the clarity of the overall process that a model follows to generate its output from the input data (James *et al.*, 2013). In models with high algorithmic transparency, users can comprehend the sequence of steps, calculations, and transformations that the model performs to arrive at its decision. This understanding allows users to assess the model's rationale and identify potential weaknesses or vulnerabilities in the model's decision-making process. Algorithmic transparent models must be completely understandable through mathematical analysis and techniques. An example of an algorithmic transparent model is linear regression (Montgomery *et al.*, 2021). Let's consider a specific example where we want to

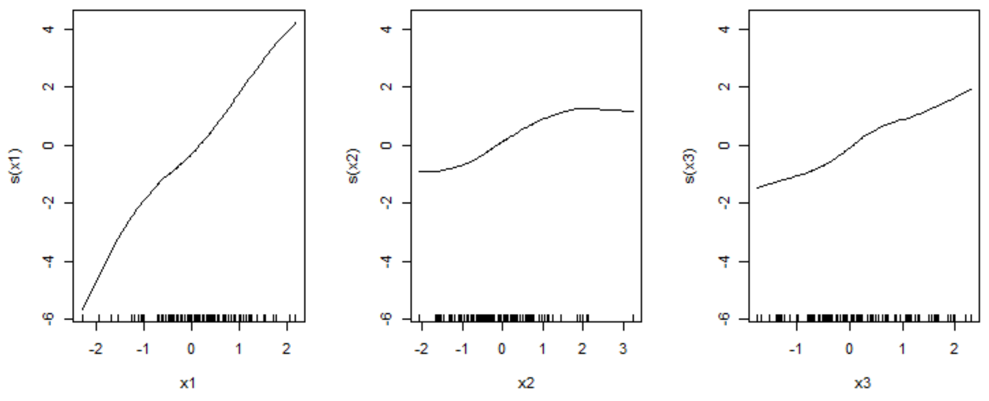


Figure 2.3. Example of plots of individual predictors effect on the output in a GAM model

predict a student's exam score based on the number of hours studied and the amount of sleep obtained the night before the exam. The linear regression model can be expressed as $exam\_score = \beta_0 + \beta_1 \cdot hours\_studied + \beta_2 \cdot hours\_slept$, where β_1 represents the expected change in the exam score for each additional hour of study, and β_2 indicates the expected change for each additional hour of sleep. Although the user may not be able to simulate the mathematical calculations of the expression, analyzing the coefficients of the linear regression model provides insights into the model's behavior and the impact of the input variables on the predicted exam score.

2.3.2. Types of data

Different data types require specialized XAI methods to effectively explain a given model. Some of the most common data types are:

- **Tabular Data**: Tabular data is the most common type of data in Machine Learning and includes structured data with rows and columns (Witten *et al.*, 2016). This type of data encompasses numerical, categorical, binary, and ordinary (ordered) data. Some of the most common methods to analyze tabular data models are LIME and SHAP, which are explained with greater detail in section 2.4.

- **Text Data**: Text data consists of natural language text, which requires specialized XAI techniques that consider the unique properties of language (Ribeiro *et al.*, 2016). Common approaches include attention mechanisms in neural networks (Vaswani *et al.*, 2017), Layer-wise Relevance Propagation (Bach *et al.*, 2015), and visualization of word embeddings (Wang *et al.*, 2018). a

- **Image Data**: Image data requires XAI techniques that can handle high-dimensional data and spatial information (Zeiler and Fergus, 2014). Techniques for explaining image models include saliency maps (Simonyan *et al.*, 2013) and Layer-wise Relevance Propagation (Rios *et al.*, 2020), which help visualize the importance of different regions within an image.

- **Graph Data**: Graph data represents relationships between entities as nodes and edges in a graph (Goyal and Ferrara, 2018). XAI techniques for graph data include Graph Neural Network Layer-wise Relevance Propagation (Schnake *et al.*, 2020).

2.3.3. Post-hoc Explainability Techniques: type of explanation

Post-hoc explainability techniques aims to retrieve information from a model after it has been trained. They are commonly used in opaque models to explain their internal functioning and decision-making process. However, they can be employed on both transparent and opaque models to generate explanations (Vale *et al.*, 2022; Arrieta *et al.*, 2020). Several means may be used to accomplish this task:

1. **Explanation by Simplification:** These techniques aims to elucidate the decision-making mechanism of a complex model by constructing a simpler, interpretable model that closely approximates the predictions of the original model. The simpler model can then be inspected to provide insights into the decision-making process of the complex model (Ribeiro *et al.*, 2016).

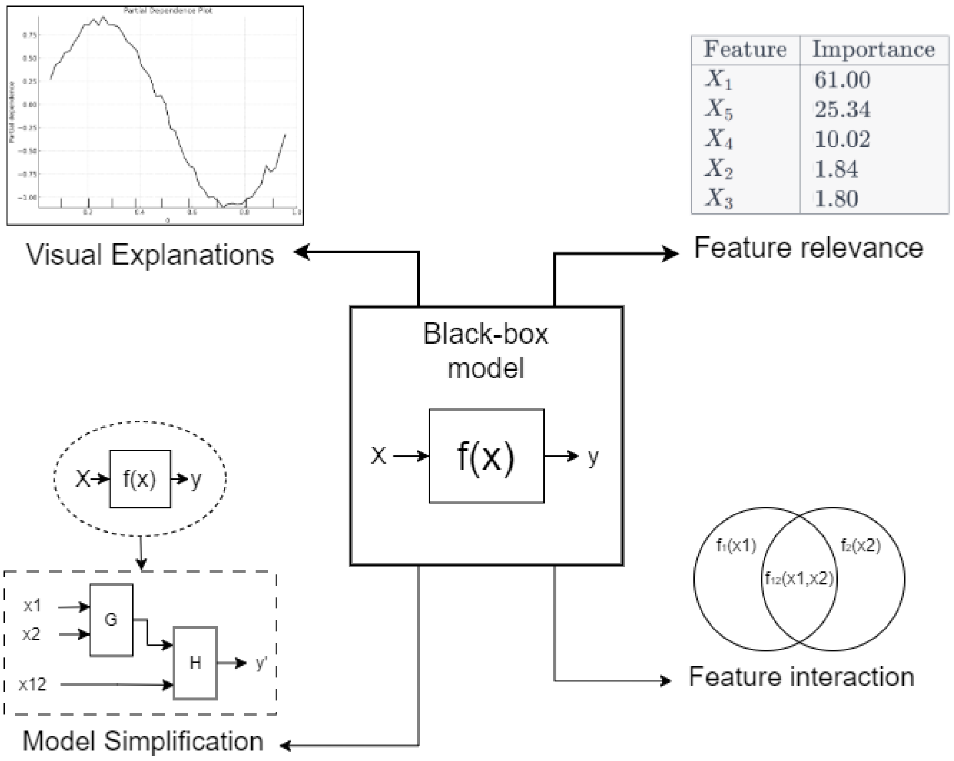
2. **Feature relevance:** These methods aim to analyze the contribution of each feature to the prediction of a specific instance or on average across all instances. This is often done by computing variable importance scores or by analyzing the type of relationship between input and output of the model (Lundberg and Lee, 2017). Feature relevance methods not only provide insight into how the model uses the input features to predict the output, but also enable the detection of potential bias in the model's decision-making process.

3. **Feature Interactions:** These methods delve into understanding how different features interact with each other to influence the model's predictions. Specifically, they explore how the joint behavior of multiple features affects the prediction, as opposed to considering each feature's effect in isolation. Such interactions are crucial in capturing complex relationships and patterns in the data, which could be missed by only examining individual feature relevance (Tsang *et al.*, 2020). This analysis often involves computing partial derivatives, interaction values, or employing decomposition techniques to dissect the model's behavior concerning interactions among features.

4. **Visual Explanations:** Visual explanations provide a way to understand the model's inner working using visual representation techniques. They offer a more intuitive and comprehensive approach to interpreting complex models by visualizing data, features, and model behavior. This category includes methods like Partial Dependence Plots (PDP) and Individual Conditional Expectation (ICE) plots, which illustrate how individual features in the model affect the predictions (Friedman, 2001).

A schematic view of each previously described post-hoc explainability techniques types can be found in figure 2.4.

Each of the aforementioned methods has its own set of advantages and disadvantages. Simplifying a complex model by generating a simpler one (or several simpler ones) allows for a detailed analysis of the model's behavior. However, this approach is typically effective only within a specific subset of the data, where the simpler model accurately represents the functioning of the complex model. Feature relevance explanations provide quantitative information about the relationship between the model's output and input variables, often across the entire dataset, enabling easier comparisons between the contributions of each variable. Nonetheless, these explanations often require a higher level of mathematical understanding to interpret the provided information. Feature interactions methods offer a nuanced understanding of how different features collectively contribute to the model's predictions by examining their synergies and interdependencies. This deeper insight is crucial for capturing complex



Feature	Importance
X_1	61.00
X_5	25.34
X_4	10.02
X_2	1.84
X_3	1.80

Figure 2.4. Schematic diagram of post-hoc explainability techniques.

relationships within the data but may also demand a more sophisticated level of analytical skill to interpret. Moreover, the analysis of feature interactions generally requires higher computational resources compared to single variable analysis, due to the increased complexity associated with examining multiple features simultaneously. Visual explanations, while intuitive, may lack quantitative details, making it challenging to compare the effects of different input variables. Ultimately, when choosing the type of explanation, it is important to consider the objectives of the explanation (such as legal requirements for quantitative explanations) as well as the characteristics of the intended recipients (where visually-oriented explanations might be more accessible for less-educated users), in order to ensure effective communication and understanding of the findings.

2.3.4. Model-specific vs Model-agnostic

Model-specific explanations are customized to particular types of models or algorithms. These methods leverage the inherent structure, properties, or characteristics of the model to elucidate its decisions. Due to their knowledge of the model's internal representations and parameters, these techniques generally offer more accurate insights into the model's behavior. Common examples of model-specific XAI methods include Garson's (Garson, 1991) or Olden's (Olden *et al.*, 2002) algorithms for calculating feature importance in Multi-Layer Perceptron (MLP) neural networks.

In contrast, model-agnostic explanations are engineered to be universally applicable to any Machine Learning model or algorithm. These methods view the model as a black box and produce explanations based on the model's input-output relationships, without information of the model's internal structure or parameters. A prominent example of a model-agnostic XAI method is the Permutation Importance algorithm, which offers a practical way to quantify the importance of features by permuting them and observing the effect on the model's performance (Breiman, 2001).

2.3.5. Local vs. Global Explanations

Global explanations strive to provide a comprehensive understanding of the behavior of a Machine Learning model across the complete input space (Saleem *et al.*, 2022), providing insights into its overall behavior rather than specific instances. This can involve understanding how different features are generally weighted in the model's decision-making process, or identifying general patterns, trends, or rules that the model follows when making predictions. However, when the model is complex or deals with a high-dimensional space, visualizing and interpreting these global interactions can become challenging.

On the other hand, local explanations focus on specific instances or predictions made by a model, revealing how the model arrived at a particular decision (Molnar, 2022). These explanations involve analyzing a localized region in the input space surrounding a specific data point. Local explanations can help users understand and validate individual decisions made by the model, which is particularly valuable in applications where accountability and traceability of predictions are crucial (Lundberg and Lee, 2017). However, one must take into account that local data distributions may exhibit behavior that differs from global trends, so the information retrieved shall not be extrapolated to other regions of the input space.

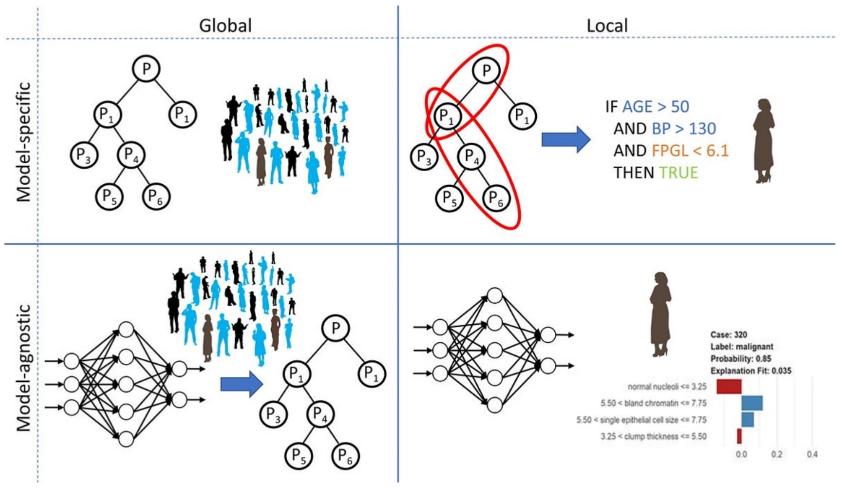


Figure 2.5. Diagram of XAI classification based on specific vs agnostic and local vs global explanation criteria. In the figure, model-specific techniques are applied to a decision tree, while model-agnostic are applied to a neural network. Global explanations are applied to a whole population, while local explanations are applied to a single person. Source: (Visani, 2020).

2.3.6. Contrastive vs. Non-contrastive Explanations

Contrastive explanations offer insights into model decisions by comparing a specific instance with one or more reference points, typically other data instances (Lundberg and Lee, 2017). The purpose of these explanations is to shed light on the underlying reasons for the differences in model predictions between the focal instance and the chosen reference points (Stepin *et al.*, 2021). For example, in image classification, a contrastive explanation may show how altering certain features of an input image would lead the model to classify it as a different object. Contrastive explanations help users understand the decision boundaries and key factors that influence the model's predictions by emphasizing the changes in outcomes.

On the contrary, non-contrastive explanations do not hinge upon comparisons between different instances. Instead, they aim to provide a holistic understanding of the model's decision-making process in relation to the specific characteristics of an instance or the model itself (Lundberg and Lee, 2017). Non-contrastive explanations can take the form of feature importance scores, attention maps, or other techniques that reveal the relative significance of input features in influencing the model's output. They help users gain a deeper understanding of how the model processes the input data, without the need for explicit comparisons.

2.4. XAI for Artificial Neural Networks

As introduced in section 2.2, Artificial Neural Networks (ANNs) are a type of Machine Learning model that mimic the structure and function of the human brain to learn patterns from data. They have gained considerable attention in recent years and has proven to be highly effective in various domains, including object detection in images (Ghasemi *et al.*, 2022), natural language processing (Devlin *et al.*, 2018), or predictive maintenance (San Roque, 1996). This success is largely due to the availability of large amounts of data and significant computational power increase in the last decade.

In a relatively brief span of time, a multitude of methods and strategies have been developed to provide explanations for non-transparent models such as ANNs (Samek *et al.*, 2021). While there are numerous studies and methods aimed at interpreting image or text-based Deep Learning models (Walia *et al.*, 2022; Pluciński, 2022); less attention has been paid to MLPs using tabular data (Sahakyan *et al.*, 2021). Therefore, the development of XAI methods that tackles this specific model and type of data is a meaningful contribution to the field of Explainable AI.

Figure 2.6 shows a categorization of post-hoc explainability techniques that can be applied to MLPs trained on tabular data, considering the distinctions between local and global approaches, as well as model-specific and model-agnostic criteria. The figure serves as an overview of the methods available to users seeking to explain an MLP model. It considers two primary factors that guide the selection of a particular method: the desire to explain a single prediction (local) or the overall behavior of the model (global), and whether the method requires specific information from the model (model-specific) or the method shall be compatible with other type of models (model-agnostic). The categorization enables users to choose an appropriate XAI technique based on their specific needs and preferences.

It is important to note that not all the methods depicted in Figure 2.6 will be discussed in subsequent sections of this thesis. Our focus will be on explaining the techniques that directly pertain to the advancements made within this research and would be later compared to the methods developed in this thesis. Before the XAI methods are explained, a common ML use case is presented to illustrate the application of these methods. At the end of the chapter, table 2.1 contains a classification of these methods based on the taxonomy criteria explained in previous sections.

2.4.1. Use Case: Prediction of flights delay

The flights dataset from the nycflights13 R library provides an ideal use case for illustrating the XAI techniques that will be explored in subsequent sections. This dataset contains information on all domestic flights departing from New York City airports (JFK, LGA, and EWR) in 2013.

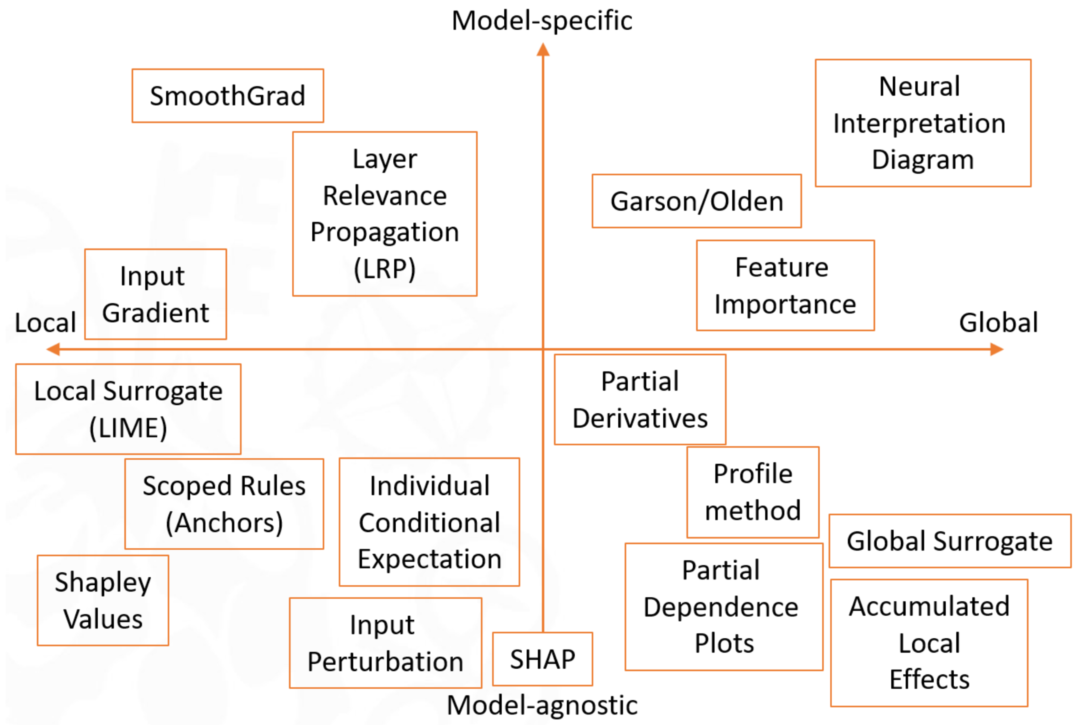


Figure 2.6. Classification of post-hoc explainability techniques for MLP based on the local vs global and model-specific vs model-agnostic criteria.

With 336,776 records and 19 variables, the dataset offers a rich source of information for analysis and interpretation. Some of the variables included in the dataset are:

- Year, month, and day of the flight
- Scheduled departure and arrival times
- Actual departure and arrival times
- Carrier code and flight number
- Origin and destination airport codes
- Air time, distance, and hour of the day
- Flight delay status (arr_delay variable)

For simplicity, only flights from United Airlines will be considered. The objective of this use case is to predict the arrival delay of the flights based on the distance of the flight ($distance$), departure and arrival time (dep_time, arr_time), departure delay (dep_delay) and time on air (air_time).

To demonstrate the various XAI techniques, we have created a Multilayer Perceptron (MLP) model using this flights dataset. Given the complex nature of flight delays and the large number of influencing factors, understanding the model's predictions is crucial for various stakeholders, such as airline operators and regulatory authorities. In the following subsections, the described XAI techniques will be applied to the MLP to gain insights into its decision-making process.

2.4.2. Local Surrogate (LIME)

LIME (Local Interpretable Model-agnostic Explanations) (Ribeiro *et al.*, 2016) is a model-agnostic technique that generates local explanations by approximating the model's behavior around a specific instance using a simpler, transparent model (e.g., linear regression or decision tree) that approximates the complex model's behavior in the local vicinity of the instance to be explained. The main steps are as follows:

1. **Sampling:** Being $x^{(j)}$ the j_{th} sample in a given dataset the instance that we want to explain, we generate a new dataset consisting of perturbed samples around $x^{(j)}$. The perturbations could involve flipping binary features or sampling from a normal distribution for continuous features.
2. **Weighting:** Each perturbed instance z_i is assigned a weight according to its proximity to the original instance $x^{(j)}$. The proximity metric can be computed using a distance function, which we denote as $\pi_{x^{(j)}}(z_i)$.
3. **Training:** A simple model g is trained on the newly generated dataset. The output values for this new dataset are obtained by using the complex model's predictions. This simple model is trained by minimizing a loss function $L(f, g, \pi_{x^{(j)}})$, that measures how unfaithful g is in approximating f in the locality defined by $\pi_{x^{(j)}}$.
4. **Interpretation:** Explanations obtained from the simple model are then used to explain the complex model, as the local behavior of both models is assumed to be similar.

Mathematically, the objective of LIME is to find and transparent model g that minimizes the following objective function:

$$\xi(x^{(j)}) = argmin_{g \in \mathcal{G}} \left(L(f, g, \pi_{x^{(j)}}) + \Omega(g) \right) \tag{2.1}$$

Where:

- f is the complex model to be explained.
- g is the simpler (transparent) model.
- $\mathcal{G}$ is the set of possible simple models.
- $\pi_x(z)$ is the proximity measure between the instance to be explained x and an instance from the generated dataset z.
- L is a measure of how unfaithfully g approximates f in the perturbed samples z_i.
- $\Omega(g)$ is a complexity measure of the model g, for example, the number of input variables.
- $\xi(x^{(j)})$ is the model that minimises the loss of the unfaithfulness of the explanation provided by g to the behavior of f for $x^{(j)}$ and the simpler model complexity $\Omega(g)$.

Advantages of LIME:

1. LIME can be applied to any black-box model, regardless of its architecture or underlying learning algorithm (Ribeiro *et al.*, 2016).
2. LIME provides explanations for individual samples, which can be more informative and actionable than global explanations.

3. LIME allows users to control the complexity of the local surrogate model, balancing the interpretability of the explanation with its fidelity to the complex model.

Disadvantages of LIME:

1. Generating perturbed instances and training surrogate models can be computationally expensive, particularly for large and complex datasets.
2. LIME's explanations can be sensitive to the choice of perturbation method and kernel function.
3. LIME's assumes that the decissions taken by the complex model can be accurately captured by a simpler model.

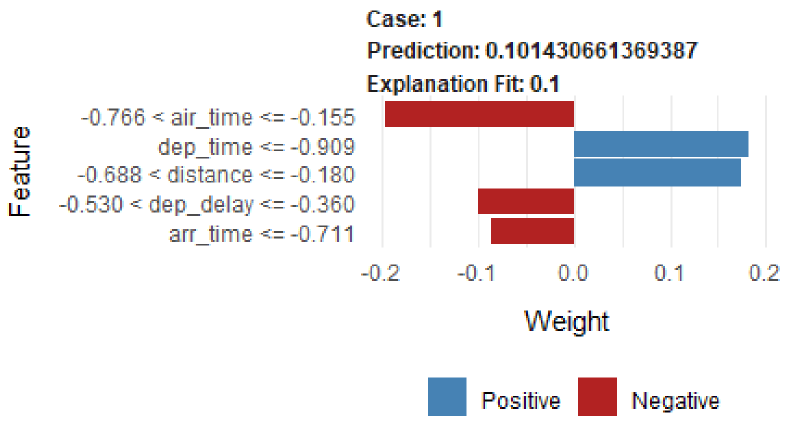


Figure 2.7. Example of plot obtained using LIME on a sample of the `flights` dataset. Attributions to each feature are assigned base on the surrogated model.

Figure 2.7 shows the local explanations of a trained decision tree model using LIME for a randomly chosen sample of the `flights` dataset. These explanations are the mean effect on the output caused by each input variable for the sample being explained. It can be seen that for the air_time value for the analyzed flight is the most influential variable, with a mean negative effect to the prediction (i.e., the air_time of the flight determines that the arrival delay would be reduced). Similar explanations can be obtained for the other input features.

2.4.3. Individual Conditional Expectation (ICE)

Individual Conditional Expectations (ICE) is a model-agnostic, post-hoc explanation technique that provides instance-level explanations by analyzing the relationship between input features and the model's predictions (Goldstein *et al.*, 2015). For each sample in the dataset, ICE plots display how the prediction changes when a single feature is varied, ceteris paribus. For a dataset with n instances, a predictive model f, and a feature x_i, the ICE plot can be formulated as follows:

Let $x^{(j)}$ represent the j^{th} instance in the dataset ($j = 1, ..., n$), and $x_{\setminus i}^{(j)}$ represent the vector of all features of the j^{th} instance except x_i. For a grid of values $x_i = x_i^{(1)}, ..., x_i^{(n)}$ for the feature x_i, the ICE plot is defined by the curve:

$$ICE(f, x^{(j)}, x_i) = \{f(x_i^{(1)}, x_{\setminus i}^{(j)}), ..., f(x_i^{(n)}, x_{\setminus i}^{(j)})\} \quad (2.2)$$

For each instance j, this creates a curve that shows how the model prediction changes when feature x_i is varied, while keeping all other feature values the same as in the original instance.

Advantages of ICE:

- Offers detailed, instance-level explanations, which can help users understand the behavior of the model for specific data points (Goldstein *et al.*, 2015).
- Can be used in conjunction with other explanation techniques, such as Partial Dependence Plots (PDPs), to obtain a comprehensive understanding of the model (Friedman, 2001).

Disadvantages of ICE:

- ICE plots can become cluttered and difficult to interpret when dealing with a large number of instances or high-dimensional feature spaces (Goldstein *et al.*, 2015).
- ICE assumes that the features are independent, which may not always be the case, potentially leading to inaccurate explanations (Friedman, 2001).
- ICE perturbates all features in the entire range, it may provide visual explanations for unfeasible scenarios.

2.4.4. Partial Dependence Plots

Partial Dependence Plots (PDP) is a model-agnostic, post-hoc visualization technique that illustrates the marginal effect of one or two input features on the predicted outcome of a Machine Learning model (Friedman, 2001).

For each value of the chosen feature, the average predictions of the model are computed. Let x_i be a feature in a dataset and f be the predictive model. $x_{\setminus i}$ represents all features excluding x_i. For a grid of values $x_i = x_i^{(1)}, ..., x_i^{(k)}$ for the feature x_i, the partial dependence function is defined by the average prediction over all instances $j = 1, ..., n$:

$$PDP(f, x_i) = \frac{1}{n}\{\sum_{j=1}^{n} f(x_i^{(1)}, x_{\setminus i}^{(j)}), ..., \sum_{j=1}^{n} f(x_i^{(n)}, x_{\setminus i}^{(j)})\} \tag{2.3}$$

This produces a curve (or surface in the case of two features) that shows the average effect of feature(s) x_i on the prediction.

These average predictions are then plotted against the corresponding feature values, providing an overview of how the model's predictions change as the feature of interest varies, while accounting for the influence of other features. PDP plots help visualize the marginal effect of a feature on the model's predictions.

Advantages of PDP:

- Provides a global view of the model's behavior, allowing users to understand the general trends and patterns in the decision-making process (Friedman, 2001).
- Visual explanations offered by PDPs are readily comprehensible due to their depiction of how output changes in response to incremental shifts in input features. This approach allows for the straightforward identification of trends and patterns, ensuring intuitive insights into the relationship between features and predictions.

Disadvantages of PDP

- PDPs assume that the features are independent, which may not always be the case, potentially leading to inaccurate explanations (Friedman, 2001).
- PDPs can be difficult to interpret when dealing with high-dimensional feature spaces or complex interactions between features (Molnar, 2022).
- PDPs do not provide quantitative information, making comparing information for several features not trivial.
- PDP perturbates all features in the entire range, it may provide visual explanations for unfeasible scenarios.

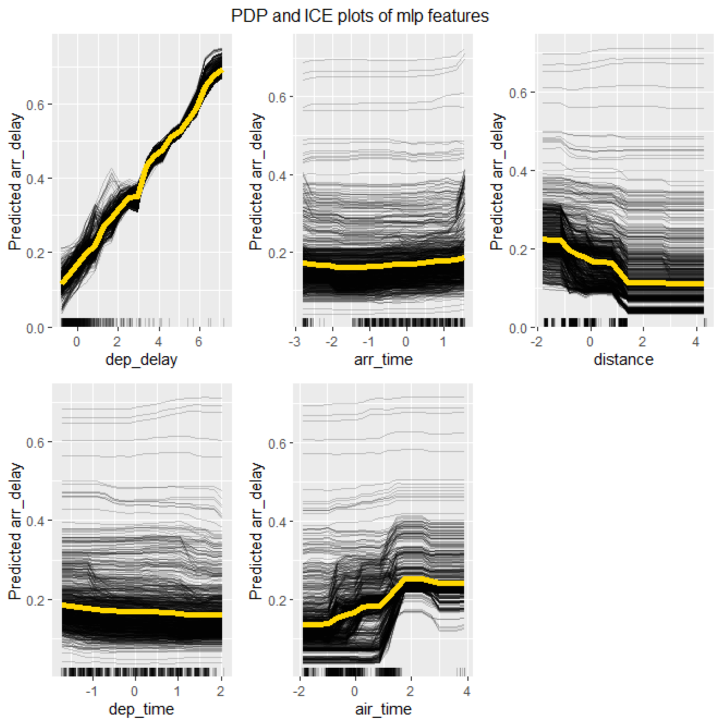


Figure 2.8. Example of plot obtained using ICE and PDP on the `flights` dataset. Effects of each feature can be analyzed based on the shape of the ICE and PDP curves.

Figure 2.8 shows the PDP and ICE curves obtained for the model trained on the `flights` dataset. As expected, the greater the departure delay (`dep_delay`), the greater the expected arrival delay. Moreover, the fact that all ICE curves have a similar shape suggests a linear relationship between the departure and the arrival delay. A non-linear behavior is observed in the `air_time` variable, where depending on the flight a greater value of `air_time` might produce a greater arrival delay or might not have any effect at all.

2.4.4.1. Partial Dependence Plots for Two variables

A natural extension of the one-variable Partial Dependence Plots (PDP) is the two-variable PDP, allowing the analysis of the combined effect of two features on the prediction (Friedman, 2001). The two-variable PDP not only shows how individual features impact the model's prediction but also how the simultaneous combination between these two features affects the outcome.

Assuming two features x_i and x_j with corresponding grids of values $x_i = x_i^{(1)}, ..., x_i^{(n)}$ and $x_j = x_j^{(1)}, ..., x_j^{(n)}$, the two-variable PDP function is computed as follows:

$$\begin{aligned} PDP(f, x_i, x_j) =& \frac{1}{n}\{(\sum_{k=1}^{n} f(x_i^{(1)}, x_j^{(1)}, x_{\setminus ij}^{(k)}), ..., \sum_{k=1}^{n} f(x_i^{(n)}, x_j^{(1)}, x_{\setminus ij}^{(k)}), ..., \\ & \sum_{k=1}^{n} f(x_i^{(1)}, x_j^{(n)}, x_{\setminus ij}^{(k)}) ..., \sum_{k=1}^{n} f(x_i^{(n)}, x_j^{(n)}, x_{\setminus ij}^{(k)}))\} \end{aligned} \tag{2.4}$$

The result is a surface or a heatmap plot, providing a comprehensive view of how the variation of both of x_i and x_j affects the model's predictions. This is particularly useful when there is a complex relationship between these two features that could not be captured by examining one-variable PDPs independently. In order to identify interaction between input variables using this plot, one must analyze how the level curves varies along the input space. Let us define the presence of an interaction between inputs x_1 and x_2 in a model $f(x_1, x_2)$ if f can be decomposed as $f = \alpha(x_1) + \beta(x_2) + \gamma(x_1, x_2)$ where $\gamma(x_1, x_2)$ represents the effect of the interaction between the two input variables. If $\gamma(x_1, x_2) = 0$, i.e., there is no interaction between the input variables, then the output of the model can be understood as a superposition of the individual effects of the input variables. Therefore, the interaction between the two input variables can be detected using this plot if, for two given points $x_1^{(1)}, x_1^{(2)}$ in one of the input variables, the magnitude $f(x_1^{(1)}, x_2) - f(x_1^{(2)}, x_2)$ is not constant for different values of x_2.

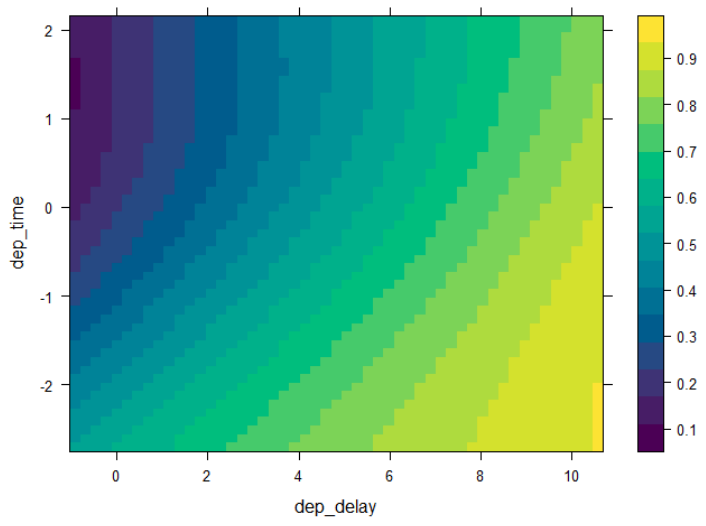


Figure 2.9. Example of a two-variable PDP plot showing the interaction of features `dep_delay` and `dep_time` on the `flights` dataset. The color intensity indicates the magnitude of the predicted arrival delay.

Figure 2.9 illustrates a two-variable PDP for `dep_delay` and `dep_time`. The color intensity reflects the magnitude of the predicted arrival delay. This 2D plot allows us to observe the combined effect of `dep_delay` and `air_time`, elucidating potential interactions between these variables not discernible from their individual PDPs. In the example, with a `dep_delay` of 0, the variation of the `dep_time` input might produce an `air_delay` from 0.1 to 0.45 (an increment of 0.35). However, with a `dep_delay` of 10, the variation of the `dep_time` input can only produce an output between 0.8 and 0.9 (an increment of 0.1). This difference in the increments indicates an interaction effect between both input variables.

However, the limitations of the one-variable PDP extend to the two-variable case. High dimensional spaces and the potential violation of the feature independence assumption can lead to misleading results. As with single variable PDPs, the insights garnered from these plots should be considered in the context of these limitations. Moreover, quantifying the degree of interaction between both input variables

2.4.5. Friedman's H-index

Friedman's H-index is a model-agnostic, post-hoc technique used to detect and quantify feature interactions in any type of predictive model (Friedman, 2001; Friedman and Popescu, 2008). The method is based on the concept of partial dependence of a function with respect to an input variable.

Retrieving the information of partial dependence as presented in the previous method, we define $\hat{f}_s$ as the partial dependence of function f with respect to a set of input variables x_s in the i^{th} sample in the dataset as:

$$\hat{f}_s(x_{is}) = \frac{1}{n}\sum_{j=1}^{n} f(x_s^{(i)}, x_{\backslash s}^{(j)}) \tag{2.5}$$

Given this definition of Partial Dependence function, the H-index of the interaction of two variables x_j and x_k is defined as:

$$H_{jk}^2 = \frac{\sum_{i=1}^{N}[\hat{f}_{jk}(x_{ij}, x_{ik}) - \hat{f}_j(x_{ij}) - \hat{f}_k(x_{ik})]^2}{\sum_{i=1}^{N}[\hat{f}_{jk}(x_{ij}, x_{ik})]^2} \tag{2.6}$$

It measures the fraction of variance of $\hat{f}(x_j, x_k)$ not captured by $\hat{f}(x_{ij}) + \hat{f}(x_{ik})$ over the data distribution. The H-statistic ranges from 0 to 1, where 0 indicates no interaction and 1 indicates a strong interaction.

Advantages of Friedman's H-index:

- Applicable to any type of model, making it a versatile method for detecting and quantifying feature interactions.
- Provides a measure of interaction strength, allowing for the identification of the most important feature interactions in a model.

Disadvantages of Friedman's H-index:

- Can be computationally expensive, especially for high-dimensional feature spaces or large datasets, as the interaction strength must be calculated for all possible pairs of features.
- Provides global interaction scores, but does not offer instance-level explanations.

Figure 2.10 shows the interactions of the input features with the `distance` variable for the trained model on the `flights` dataset. In this case, the greater interaction is detected between the distance of the flight and the time in the air, which might suggests that longer flights would have greater or lower arrival delays. Although the interaction is detected, information of the effect of the interaction on the model's prediction is not obtained.

2.4.6. Lek's profile Method

The Lek's Profile method is a model-agnostic, post-hoc explanation technique that generates instance-level explanations by perturbing the values of all features simultaneously and assessing

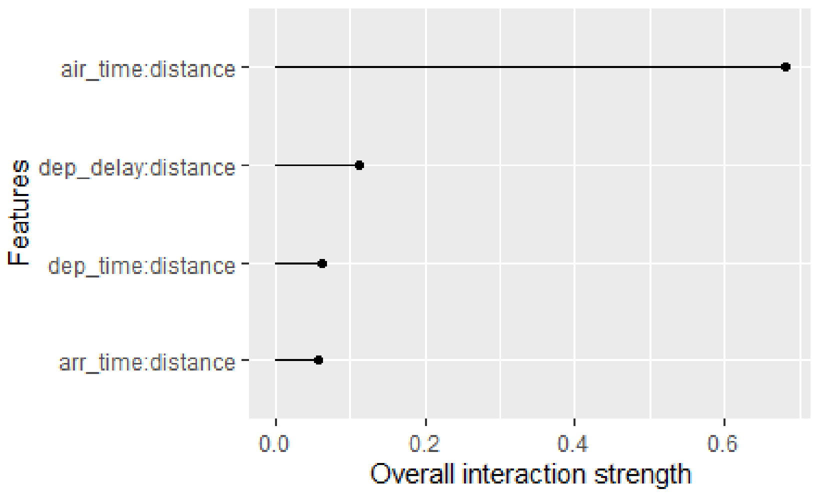


Figure 2.10. Example of plot obtained using Friedman's H-index on the `flights` dataset. Interactions of input features with `distance` input variable are shown.

the impact of these changes on the model's predictions (Gevrey *et al.*, 2003). The method creates a series of instances by modifying the values of input features according to predefined perturbation schemes, such as sampling from their distribution or applying predefined transformations, and records the corresponding model predictions. The resulting profiles visualize the model's sensitivity to changes in the input features and offer insights into the decision-making process for specific instances. A variation of this method was proposed in (Lek *et al.*, 1996), where the quartiles of each input variable are used as the instances to assess the impact on the model's prediction.

Advantages of the profile method:

- Offers a middle-point view between PDP and ICE, which can help users understand general trends in the data while capturing more variability than PDP (Gevrey *et al.*, 2003).

Disadvantages of the profile method:

- The method can be computationally expensive, especially for high-dimensional feature spaces or large datasets (Gevrey *et al.*, 2003).
- Profile plots do not provide quantitative information, making comparing information for several features a not trivial task.
- Profile perturbates all features in the entire range, it may provide visual explanations for unfeasible scenarios.

Figure 2.11 shows the Lek's profile curves for the model trained on the `flights` dataset. The information obtained is similar to the one of PDP and ICE curves. In this case, the fact of using a subset of curves make it easier to obtain information of the type of relationship between input and output variables. However, if a flight does not follow the general patterns shown in the figure, Lek's profile would not be able to retrieve that information, contrary to ICE.

2.4.7. SHAP

SHAP (SHapley Additive exPlanations) is a unified, model-agnostic, post-hoc explanation technique that provides both global and instance-level explanations for the predictions of any

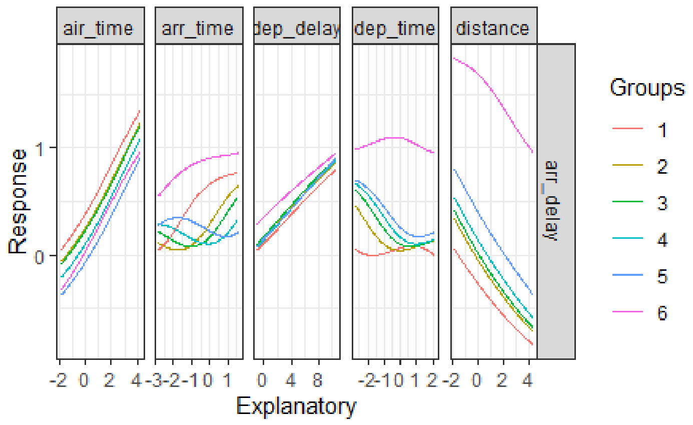


Figure 2.11. Example of plot obtained using Lek´s profile on the `flights` dataset. Effects of each feature can be analyzed based on the shape of the profile curves.

Machine Learning model (Lundberg and Lee, 2017). The method is based on the concepts from cooperative game theory, particularly the Shapley value. The Shapley value is a widely accepted method for fairly distributing gains among cooperating entities by calculating the average marginal contribution of each entity to all possible coalitions (Shapley, 1997). It quantifies the importance or impact of each feature by considering its interaction with other features in different combinations.

The calculation of Shapley values involves evaluating the model's output for every possible subset of features, comparing the predictions with and without the inclusion of a specific feature. This calculation is shown in equation 2.7.

$$\phi_i(v) = \sum_{S \subseteq p \setminus \{i\}} \frac{|S|!(|p| - |S| - 1)!}{|p|!} [v(S \cup \{i\}) - v(S)] \tag{2.7}$$

where:

- The sum ranges over all subsets S of the player set N excluding the player i.
- $|S|$ is the cardinality of the set S (i.e., the number of players in S).
- $|p|$ is the total number of players.
- The terms $|S|!(|N| - |S| - 1)!$ and $|N|!$ represent the number of orderings that place the i^{th} player in a particular position, and all possible orderings, respectively. They are used to give each player a fair share of the total payout.
- The expression in the square brackets gives the marginal contribution of player i when added to subset S.
- $v(S)$ is the characteristic function, which gives the payoff for a coalition S, where:

$$v(S) = \mathbb{E}\left[f(X) \mid X_S = x_S, X_{p\setminus S} = x^*_{p\setminus S}\right] \tag{2.8}$$

  - x is a random instance from your dataset.
  - x_S is the subset of features in x corresponding to the set S.
  - x_S is the subset of features in x corresponding to the set S.

- $x^*_{p\setminus S}$ represents the "baseline" values for the features not in S.

The Shapley value of player i is then the weighted average of these marginal contributions across all possible coalitions.

In the context of interpretability, Shapley values can be used to explain the importance of each feature for a specific prediction or output, interpreting the features as the players and the prediction as the payout. SHAP uses the Shapley values for each individual feature as an additive feature attribution method, i.e., a linear model. From the Shapley values definition provided in 2.7, SHAP gives a local explanation for a sample as:

$$\hat{f}(x^{(j)}) = E_X(\hat{f}(X)) + \sum \phi_i^{(j)} \cdot x_i^{(j)} \tag{2.9}$$

where $x^{(j)}$ is the j^{th} sample of dataset X that shall be explained, $\hat{f}(x^{(j)})$ is the prediction of the model f for the j^{th} sample, $E_X(\hat{f}(X))$ is the mean value of the predictions for all samples in dataset X, $\phi_i^{(j)}$ is the Shapley value associated to variable x_i in the j^{th} sample of dataset X and $x_i^{(j)}$ is the value of the i^{th} variable in the j^{th} sample of the dataset X. This method can be applied to any type of model and has various efficient approximations for specific model classes, such as TreeSHAP for tree-based models (Lundberg, Erion, *et al.*, 2020).

Although Shapley values are obtained for an specific sample, if the Shapley values are calculated for all the samples in a given dataset they can be used to obtain global information from the model. For example, the mean of absolute Shapley values for a given variable provides a feature importance measure of the variable.

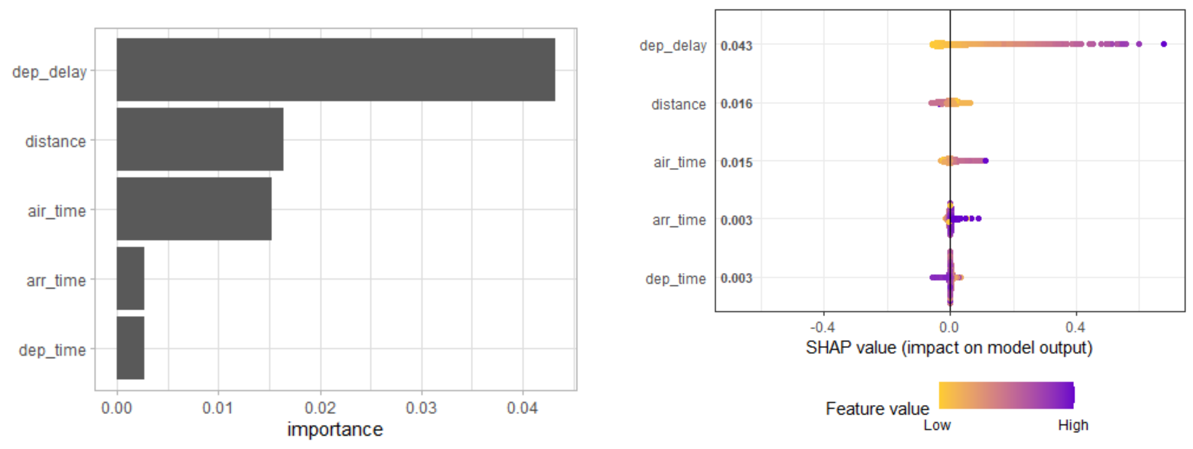
Advantages of SHAP:

- Offers both global and instance-level explanations, providing a comprehensive understanding of the model's behavior (Lundberg and Lee, 2017).
- Accounts for feature interactions and dependencies, offering more accurate and fair explanations compared to other methods that assume feature independence (Lundberg and Lee, 2017).
- Provides a theoretically grounded and consistent framework for attributing importance values to features.

Disadvantages of SHAP:

- The method can be computationally expensive, especially for high-dimensional feature spaces or large datasets (Lundberg and Lee, 2017).
- Assumes that the difference between the mean output of the model and the explained prediction is a linear function of the input values, which may not always be the case.
- SHAP values may become difficult to interpret when dealing with a large number of instances or high-dimensional feature spaces.
- Understanding the Shapley value concept and its connection to Machine Learning models may require a relatively high level of mathematical background.

Figure 2.12 shows two types of plots that can be obtained using SHAP on the model trained on the `flights` dataset. The first plot shows the input feature importance regarding their contributions to the models's predictions as the mean Shapley values for each variable in the



(a) Feature importance **(b)** Feature effect

Figure 2.12. Example of plots obtained using SHAP on the `flights` dataset. (a) Importance of each input feature (b) Effect of each input feature.

`flights` dataset. For this dataset, the departure delay is the most important feature to predict the arrival delay. The second plot shows the Shapley value of each variable in each sample of the dataset. A continuous color gradient for a variable indicates a linear relationship between input and output, while a non-continuous gradient indicates a non-linear relationship.

2.4.8. Permutation Importance

Permutation Importance is a model-agnostic, post-hoc feature importance measurement technique that quantifies the impact of each feature on the model's predictions by assessing the change in model performance when the values of a particular feature are randomly permuted (Altmann *et al.*, 2010). The method works by first computing the baseline model performance using a chosen evaluation metric. Then, for each feature, the feature values are randomly shuffled, and the model's performance is re-evaluated. The difference between the baseline performance and the permuted performance represents the importance of the feature. A larger decrease in performance indicates a more important feature. Mathematically, the Permutation Importance of a feature x_i is given by $\text{PI}(x_i) = \rho_{\text{baseline}} - \rho_{\text{permuted}i}$, where ρ is the performance metric, and the permuted performance ρ_{permuted_i} is calculated after randomly shuffling the values of the feature x_i in the dataset.

Advantages of Permutation Importance

- Applicable to any type of model, making it a versatile method for providing feature importance measurements.
- Relatively simple to implement and understand, with no need for advanced mathematical background.
- Accounts for feature interactions, as it directly measures the impact of a feature's perturbation on the model's predictions (Molnar, 2022).

Disadvantages of Permutation Importance

- Can be computationally expensive, especially for high-dimensional feature spaces or large datasets, as the model's performance must be re-evaluated for each feature (Molnar, 2022).

- Provides global feature importance scores, but does not offer instance-level explanations.
- Susceptible to noise or leakage in the data, as it evaluates the importance of a feature based on its effect on the model's performance, which may be influenced by noisy or leaking features.

Figure 2.13 shows the permutation feature importance for the model trained on the `flights` dataset. It must be noted that, although the information is similar to the one from figure 2.12 obtained using SHAP, the meaning of importance is different between the two methods. Using permutation importance, the importance metric is correlated directly to the prediction error. Using SHAP, the importance metric is correlated directly to the contributions to the model's prediction, not to the prediction error.

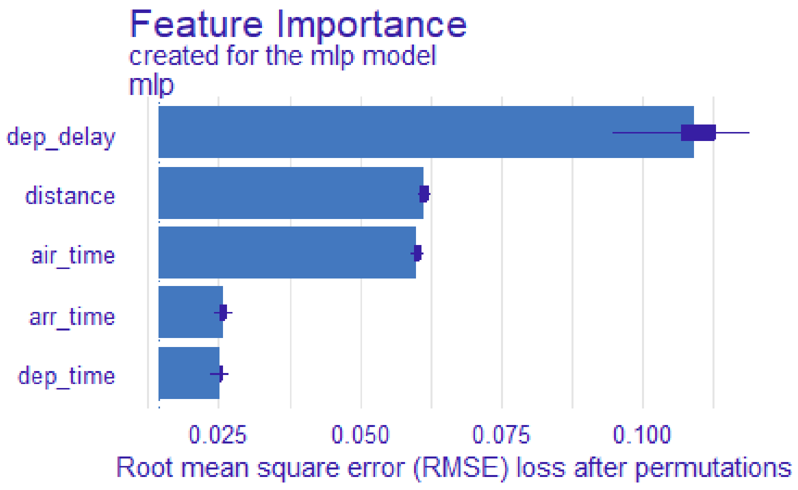


Figure 2.13. Example of plot obtained using permutation importance on the `flights` dataset.

2.4.9. Garson/Olden Importance

Garson and Olden Importance is a model-specific, post-hoc feature importance measurement technique, specifically designed for Artificial Neural Networks (ANNs) (Garson, 1991), (Olden *et al.*, 2002). Olden's method is usually considered an evolution of Garson Importance method. Both methods calculate the importance of input features by analyzing the connection weights between the input layer, hidden layers, and output layer of an ANN.

Garson's method assigns importance to a feature x_i, $i = 1, \dots, p$; based on the sum of the absolute values of the weights w_{ij}, $j = 1, \dots, m$ connecting the input node i to each of the m hidden nodes, multiplied by the absolute values of the weights w_{jk}, $k = 1, \dots, n$ connecting each hidden node j to the output nodes k. The importance of feature x_i, $GI(x_i)$, is calculated as follows:

$$GI(x_i) = \frac{\sum_{j=1}^{m} \sum_{k=1}^{n} |w_{ij}| \, |w_{jk}|}{\sum_{i=1}^{p} \sum_{j=1}^{m} \sum_{k=1}^{n} |w_{ij}| \, |w_{jk}|} \tag{2.10}$$

Olden's method, on the other hand, uses the actual values of the weights in the calculation, allowing for negative contributions:

$$OI(x_i) = \frac{\sum_{j=1}^{m} \sum_{k=1}^{n} w_{ij} w_{jk}}{\sum_{i=1}^{p} \sum_{j=1}^{m} \sum_{k=1}^{n} w_{ij} w_{jk}} \tag{2.11}$$

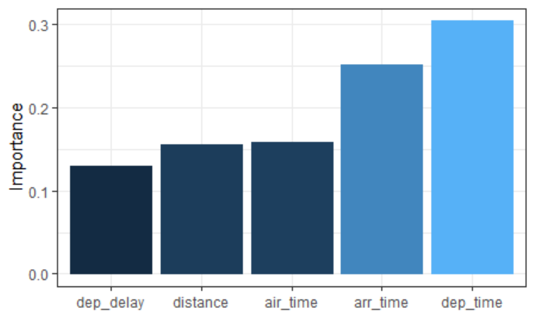
Advantages of Garson/Olden Importance:

- Relatively simple to implement and understand, with no need for advanced mathematical background.
- Can be applied to any feedforward ANN architecture, offering a valuable tool for understanding feature importance in these models.

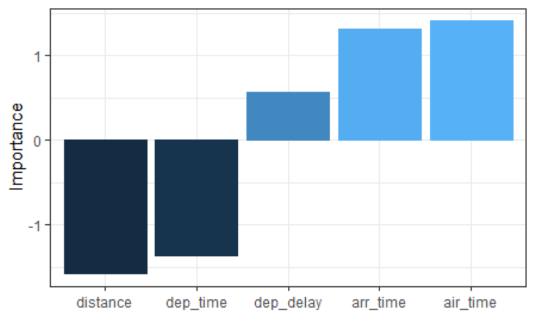
Disadvantages of Garson Importance:

- Provides global feature importance scores, but does not offer instance-level explanations.
- Highly susceptible to the model weights value. This makes them susceptible to any factor which leads to a different optimal set of weights during the model training (Beck, 2018).
- Does not take into account the value of the input variables, it only focus on the neural network architecture.

Figure 2.14 shows the garson's and olden's importance for the model trained on the `flights` dataset. The information retrieved using this method seems unintuitive, as the department delay is assigned the lowest importance. However, it must be considered that the value of the variable does not affect the importances assigned using these methods, which might influence notably the effect of the input variables on the output.



(a) Garson's importance



(b) Olden's importance

Figure 2.14. Example of plots obtained using Garson's and Olden's algorithm on the `flights` dataset. Input importance in Olden's also gives information about the effect of the feature input on the output.

2.4.10. Sensitivity Analysis based on Partial Derivatives

Y. Dimopoulos *et al.*, 1995 present the concept of sensitivity of a model as the partial derivative of the output with respect to the input:

$$\xi = \frac{\partial \hat{y}}{\partial x} \tag{2.12}$$

where $\hat{y}$ represents the output or predicted value of the model, and x represents the input or feature of interest.

This partial derivative quantifies how the predicted value changes with incremental modifications in the input feature. By evaluating these partial derivatives in all the points in the training dataset (or an analog dataset), we obtain a distribution of partial derivatives of the output with respect to each input variable. Sensitivity Analysis is a model-agnostic, post-hoc method that calculate statistics about these distributions to retrieve information about the input-output relationships of the model (Y. Dimopoulos *et al.*, 1995; Muñoz and Czernichow, 1998; I. Dimopoulos *et al.*, 1999; White and Racine, 2001; Gevrey *et al.*, 2003; Gevrey *et al.*, 2006; Zhang *et al.*, 2022; Pizarroso-Gonzalo *et al.*, 2022). In fact, Muñoz and Czernichow (1998), White and Racine (2001) and Sobol' and Kucherenko (2009) defend the mean of the squared partial derivatives of the output with regard to the input as a valid statistic to determine if an input is significant to predict the output. We provide a detailed description of this method in section 3.3.

Advantages of Sensitivity Analysis based on Partial Derivatives:

- Relatively simple to implement and understand, requiring only basic calculus knowledge.
- Can be applied to any differentiable model, offering a valuable tool for understanding feature importance in these models.
- Can provide information from local to global level depending on the dataset used to calculate partial derivatives.

Disadvantages of Sensitivity Analysis based on Partial Derivatives:

- Input variables should be normalized when using this method, as otherwise the value of the partial derivatives may depend on the scale of each variable and produce misleading results.

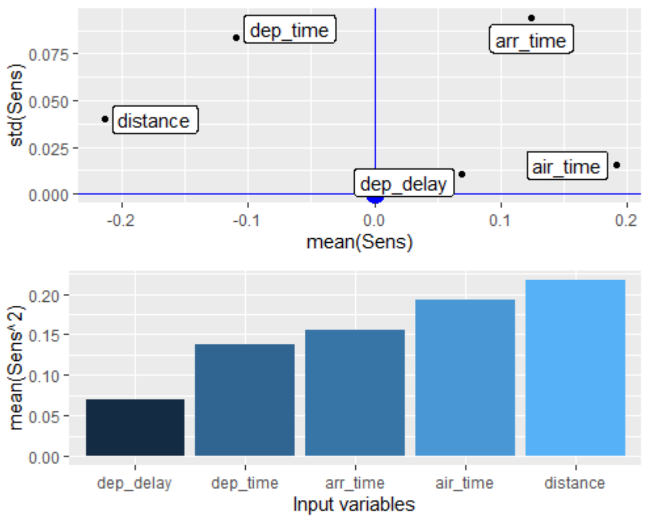


Figure 2.15. Example of plot obtained using Sensitivity Analysis based on partial derivatives on the `flights` dataset.

Figure 2.15 shows the sensitivity analysis results using one of the methods developed in this thesis. This method provides information of the type of relationship between input and output in the first plot and feature importance in the second plot. As this method is explained in following sections, further explanations are not provided here.

2.4.11. Computational Time Analysis of XAI Methods

A critical factor to consider in the application of XAI methods is the computational time required to derive the explanations. This is particularly important in scenarios where real-time or near real-time explanations are required. To assess this, an empirical study was conducted using the YearPredictionMSD dataset (Bertin-Mahieux, 2011). This dataset consists of 90 input variables and 515345 samples. The target of this dataset is the prediction of the release year of a song from audio features. In this case, we are not interested in how good the model is able to predict the output, but in how computationally intensive the XAI methods previously described are.

In order to compare the computational resources needed to perform a given XAI method, we propose to measure the computational time when varying the number of input variables and the number of samples of the analyzed dataset, and the size of the hidden layer of a single hidden layer MLP model. These variations allow us to understand the impact of model complexity, data size, and dimensionality on the computation time of the XAI methods.

A MLP model was trained with a different configuration. The configurations tested were:

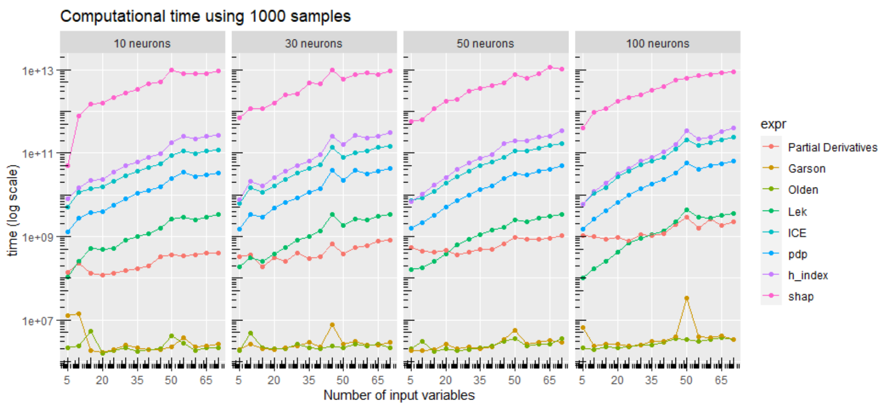
- Number of neurons in the hidden layer of the MLP model: 10, 30, 50, 100
- Number of samples of the dataset: 1000, 5000, 10000
- Number of variables of the dataset: 5, 10, 15, ..., 70

These analysis have been performed on a computer with the following specs: processor Intel(R) Core(TM) i5-6300HQ @ 2.30GHz, 32 GB of RAM memory, R version 4.3.0 (2023-04-21 ucrt), platform x86_64-w64-mingw32/x64 (64-bit) and running under Windows 10 x64 (build 19045).

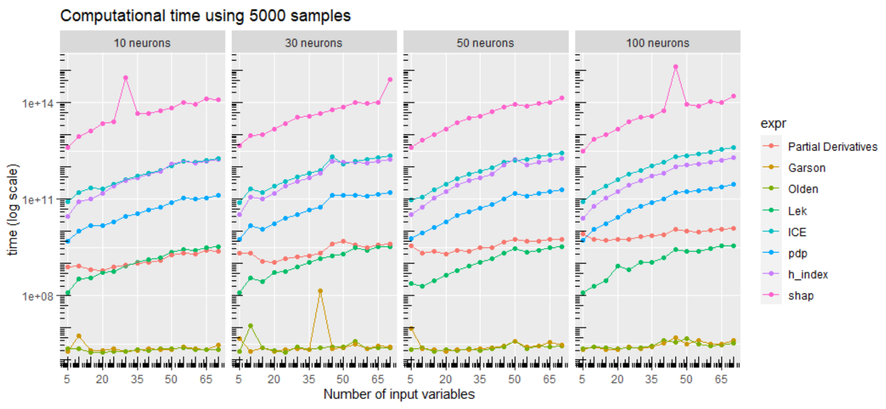
For each model, all the XAI methods discussed in this chapter were applied, and the time required for each method was logged. It should be noted that the computational time includes the time to derive the explanations, but does not include any pre-processing or post-processing time associated with the method (such as creating the plot in methods like ICE or PDP).

Figure 2.16 shows the computational time for each XAI method with the varying model and data configurations. Some conclusions can be reached from this figure:

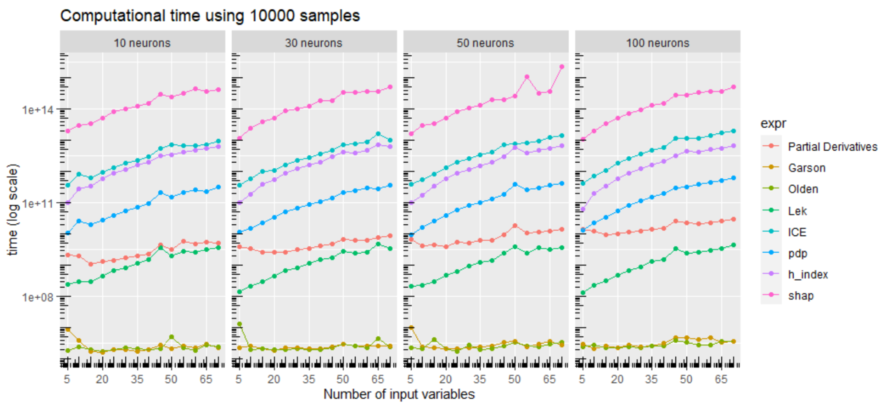
- Olden's and Garson's feature importance methods (implemented in the *olden* and *garson* functions in the **NeuralNetTools** R package (Beck, 2018)), are the most efficient techniques. They function by performing a sum of the weight matrices of the model, rendering their computational time directly proportional to the size of the neural network layers (input and hidden layers) rather than the number of samples.
- The Lek´s Profile (calculating 6 profiles using the *lekprofile* function from **NeuralNetTools** R package) and the Sensitivity Analysis based on partial derivatives methods (using the *SensAnalysisMLP* function from **NeuralSens** R package (Pizarroso-Gonzalo *et al.*, 2022)) need of the similar computational times. It must be noted that Lek's profile scales better with the number of samples, as the number of profiles remains constant for all trained models and increasing the number of predictions to compute the partial dependence of each profile does not significantly increase the computational time. However, the computational time required by Sensitivity Analysis based on partial derivatives remains approximately constant for a given amount of samples no matter the number of input variables analyzed. Lek's method is severely affected by the number of input variables because it needs to calculate 6 profiles for each new variable analyzed.
- The PDP, ICE and Friedman's H-index method rely in the same concept - partial dependence (using the *Partial* function from the **iml** package (Molnar *et al.*, 2018)).



(a) Computational time using 1000 training samples.



(b) Computational time using 5000 training samples.



(c) Computational time using 10000 training samples.

Figure 2.16. Computational time of the different sensitivity analysis methods with different number of training samples (1000, 5000 and 10000 training samples), number of input variables (from 5 to 90 input variables) and number of neurons in the hidden layer (10, 30, 50 and 100 neurons).

However, as PDP only needs to compute the mean partial dependence curve for each variable, it is noticeable faster than the other two methods. H-index requires almost twice the time than PDP to compute, which makes sense as it computes the interaction between two variables based on the mean partial dependence with respect to each variable and the mean partial dependence with respect both variables, requiring to calculate the same as the PDP method plus the each pair combination. ICE scales severely worse with respect the number of samples than the other two, as it requires to calculate a partial dependence curve for each variable and each sample.

- The slowest function is the SHAP method (using the *Shapley* function from the **iml** R package). It also is the worst scaling function with respect to the number of samples analyzed, as it estimates the shapley value for each variable and for each sample. In case that we only want to obtain local explanation from a subset of samples or variables, the time required by the algorithm shall decrease drastically, as the operations performed by SHAP depends on the cardinality of the set of variables analyzed and the samples that each Shapley value is calculate for.

Alongside these findings, it is important to note that our analysis has been executed using a model with a single output variable. XAI techniques employing matrix or tensor operations, such as Garson's, Olden's, or Partial Derivatives methods, generate explanations for all output variables through the same set of operations.

However, other methods like PDP and SHAP necessitate separate operations for each output variable, subsequently leading to a proportional increase in computational time when generating explanations for multiple outputs. This distinction is a significant consideration when dealing with models with multiple output variables and can significantly influence the selection of an appropriate XAI method.

2.5. Current Challenges

Despite the significant advancements in the field of Explainable Artificial Intelligence (XAI), several challenges persist that need to be addressed:

- **Interpretability of High-Dimensional Feature Spaces:** Techniques like ICE can become cluttered and difficult to interpret when dealing with a large number of instances or high-dimensional feature spaces.
- **Explanation of Unfeasible Scenarios:** Some techniques, like ICE or PDP, perturb all features in the entire range, which may provide explanations for unfeasible scenarios. This is usually related to the assumption of feature independence, where each feature is considered in isolation without considering potential dependencies or interactions with other features.
- **Balancing Interpretability and Computational Costs:** The trade-off between the information quality/quantity provided by an XAI method and the computational resources required to execute the algorithm is a common consideration. Certain methods, such as Garson's, exhibit fast computation times, but they tend to provide only a quantity of feature importance without the robustness offered by other techniques like SHAP. In contrast, SHAP not only provides feature importance but also offers valuable insights into the input-output relationship. However, the computational demands of SHAP make it unfeasible for its application in high-dimensional datasets.
- **Holistic Explanations for Global and Local levels:** Current XAI strategies predominantly cater to either local or global context, yet a pressing need exists for methodologies that provide coherent explanations at both levels. Imagine a criminal justice system deploying an AI model to predict recidivism rates. On a local level, comprehending why a specific individual receives a high-risk prediction is crucial for just and transparent decision-making. Simultaneously, understanding overarching patterns in the model's behavior across the entire incarcerated population provides actionable insights for policy improvements. Bridging this analytical gap not only enhances model accountability but

also empowers stakeholders at every level to make informed, ethical choices that align with individual rights and societal needs.

These challenges highlight the need for further research and development in the field of XAI to improve the interpretability and transparency of Machine Learning models. The methodologies and advancements proposed in this thesis tackle these challenges by employing partial derivatives as a tool to scrutinize the relationships between inputs and outputs based on the dataset samples. This approach ensures more precise and practical explanations for intricate models and datasets. Importantly, this method refrains from making presumptions about the relationships between variables and inherently avoids unrealistic scenarios. It facilitates the interpretability of high-dimensional feature spaces by implementing aggregation measures on the derivatives and using vectorial operations to accelerate the derivatives calculation. Furthermore, it provides explanations at an intermediate level by examining the partial derivatives at varying levels of aggregation, thereby offering a more nuanced understanding of the model's behavior.

2.6. Conclusions

In conclusion, this chapter has provided an in-depth exploration of Explainable Artificial Intelligence (XAI) techniques. The importance of these techniques in creating explanations from AI models has been underscored, emphasizing their role in making AI decisions understandable to humans. This is crucial in a world where AI is increasingly used in decision-making processes, from healthcare to finance, and users need to trust and understand the decisions made by these models.

We have provided a detailed taxonomy of XAI, which takes into account various factors such as the explanation purpose, transparency levels of models, post-hoc explainability techniques, types of data, model-agnosticism, local and global explanations, and contrastive and non-contrastive explanations. Furthermore, this chapter has presented a detailed explanation of various XAI methods, including LIME, ICE, PDP, Lek's Profile, SHAP, Permutation Importance, Garson Importance, Olden Importance, Sensitivity Analysis, and Friedman's H-Index. Each of these methods has its unique characteristics and computational resource requirements, making them suitable for different types of data and explanation needs. This taxonomy aims to guide researchers and practitioners in selecting the most suitable XAI techniques for their specific requirements and contexts.

However, it is important to acknowledge that the XAI techniques discussed in this section are not without limitations. Some of the most complete XAI methods, such as SHAP can be computationally unfeasible for high-dimensional dataset. Other commonly used XAI techniques, such as LIME, does not give quantitative importance measures, which make the comparisons between variable a non-trivial challenge. Finally, permutation methods, such as PDP, might analyze unrealistic scenarios and give misleading information to the user. Therefore, new XAI methods must been designed to overcome this limitations and increase the applicability of XAI in current ML applications.

Table 2.1. Taxonomy of described XAI methods

Criteria	**Type of Explanation**	**Data Type**	**Local/ Global**	**Contrastive/ Non-contrastive**	**Comp. Resources**
LIME	Explanation by Simplification, Feature Relevance Explanation	Tabular, Text, Image	Local	Non-contrastive	Medium
ICE	Visual Explanation	Tabular	Local	Non-contrastive	High
PDP	Visual Explanation, Feature Interactions	Tabular	Global	Non-contrastive	Medium
Lek's Profile	Visual Explanation	Tabular	Global	Non-contrastive	Medium
SHAP	Feature Relevance Explanation, Feature Interactions, Visual Explanation	Tabular, Text, Image	Both	Contrastive	High
Permutation Importance	Feature Relevance Explanation	Tabular	Global	Non-contrastive	Medium
Garson Importance	Feature Relevance Explanation	Tabular	Global	Non-contrastive	Low
Olden Importance	Feature Relevance Explanation	Tabular	Global	Non-contrastive	Low
Sensitivity Analysis	Feature Relevance Explanation, Visual Explanation	Tabular	Both	Non-contrastive	Medium
Friedman's H-Index	Feature Interactions	Tabular	Global	Non-contrastive	Medium

3

Development of XAI methods based on Partial Derivatives for Multilayer Perceptron

Success is walking from failure to failure with no loss of enthusiasm.
Winston Churchill (1874–1965)

This chapter presents the main innovations developed during the thesis related to using partial derivatives as a XAI method. It includes the rationale on how to apply XAI methods based on partial derivatives and how to interpret their results.

3.1. Introduction

The study of Explainable Artificial Intelligence (XAI) for MLP through the lens of sensitivity analysis and partial derivatives has been developed during the last decades. This section emphasizes significant contributions that have established the groundwork for our present research.

The seminal work by Zurada *et al.* (1994) introduced the concept of sensitivity as partial derivatives of the output with respect to the inputs evaluated in the samples of the dataset. The authors innovatively utilized the ratio of error divided by mean sensitivity to detect the most important variables for a model. Their study also presented the first partial derivative formulas for Multilayer Perceptron (MLP) networks with one hidden layer and one output variable.

Building on this foundation, Muñoz and Czernichow (1998) applied sensitivity analysis based on partial derivatives for variable selection in feedforward and recurrent neural network models. Their approach, which employs the mean of squared sensitivities as an indicator of feature importance, marked a significant step forward in the field.

An innovative proposal was presented in White and Racine (2001), who devised a statistical methodology based on the bootstrapped distribution of the mean of squared sensitivities. This method allowed for detecting significant input variables in MLP models.

Expanding the scope of partial derivatives, Gevrey *et al.* (2006) utilized the mean of the absolute value of second partial derivatives to identify interactions between pairs of variables. Their work, using 10 habitat characteristics to predict the density of brown trout spawning redds, highlighted the practical implications of these theoretical advancements with the analysis of the contribution profiles of each pair of variables. From the contribution profile patterns, it was seen that the predicted density of redds closely corresponded to ecological reality. Moreover, the contributions of the variables, which were not significantly differentiated with the first partial derivatives, were revealed with the second partial derivatives.

Building on prior research, Zhang *et al.* (2022) employ the mean of squared second partial derivatives as a technique for detecting interaction importance. They validate the effectiveness of this method by applying it to several synthetic datasets with known pairwise input interactions. Subsequently, they develop a new model, ParaACE, which utilizes the interaction knowledge gained to train a transparent surrogate model. This surrogate model demonstrates comparable performance across the tested datasets.

Our research uniquely contributes to the advancement of XAI methods by not only contributing theoretically on the significant previous works in this field but also implementing and presenting these advancements in a readily accessible manner. One important contribution of our research has been the development of two software packages: the Python package `neuralsens` and the corresponding R library `NeuralSens`. These packages embody the research findings and methodologies detailed in this section and provide a practical interface for applying these concepts in real-world scenarios.

We believe that our development and popularization of `neuralsens` and `NeuralSens` have served to bridge the gap between complex theoretical advancements and their practical implementations. This assertion is substantiated by the over $100,000$ downloads that the packages have received at the time of writing, indicating a robust and growing interest within the research community for accessible XAI tools. This remarkable number also reflects our contribution in fulfilling the demand of researchers, practitioners, and enthusiasts for tools that promote transparency and interpretability in machine learning models.

In this chapter, we begin by investigating the partial derivatives of activation functions, followed by an in-depth exploration of the first, second, and third partial derivatives of Multilayer Perceptrons (MLP) with an arbitrary number of hidden layer and output variables. Subsequently, we present three novel algorithms that harness the power of these partial derivatives to provide meaningful insights and enhanced interpretability of ML models. By carefully examining these developments, we aim to push the boundaries of XAI techniques and offer a comprehensive toolkit for practitioners and researchers alike to improve their understanding of complex neural networks.

3.2. Calculation of the Partial Derivatives of the Multi-Layer Perceptron (MLP) Model

In this section, we systematically explore the computation of partial derivatives in the context of MLPs, providing a solid foundation for understanding the inner workings of our XAI techniques. We begin by presenting the MLP model architecture and the nomenclature that will be used in the rest of the chapter. Then, we continue breaking down the process of calculating partial derivatives for each layer within the MLP, taking into consideration various activation functions and their respective properties. Subsequently, we delve into the first, second, and third-order partial derivatives of the output with respect to the input in a MLP model. By dissecting these calculations step by step, we aim to offer a comprehensive understanding of the MLP model's sensitivity to changes in input variables.

3.2.1. The Multi-Layer Perceptron (MLP)

A MLP is a fully-connected feed-forward Artificial Neural Network (ANN) that has one-way connections from the neurons of one layer to all neurons of the subsequent layer. Each time the output of one unit travels along one connection to another unit, it is multiplied by the weight of the connection. At each unit the weighted inputs are summed and a constant, or bias, is added. Once all the input terms of each unit are summed, an activation function is applied to the result. Figure 3.1 shows the scheme of a neuron in a MLP model and represent graphically

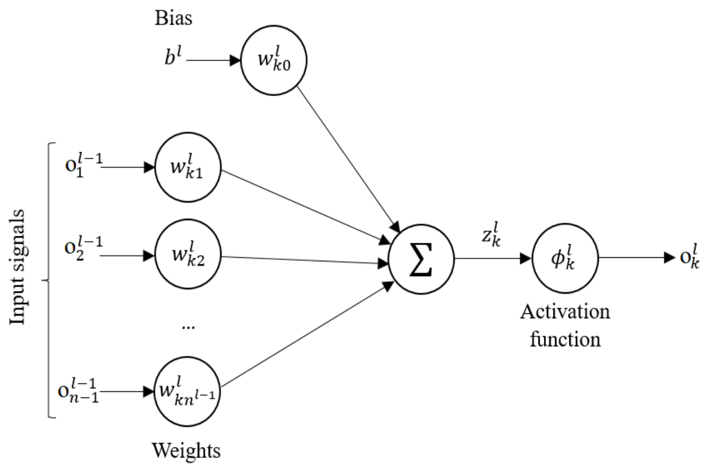


Figure 3.1. Scheme of the k^{th} neuron in the l^{th} layer of a MLP model. ϕ^l_k represent the activation function of the neuron, b^l represent the bias of the l^{th} layer, o^{l-1}_j represent the output of the j^{th} neuron in the previous layer and w^l_{jk} represent the weight of the connection between the neuron and the j^{th} neuron of the previous layer.

the operations in Equation 3.1.

For each neuron, the output o^l_k of the k^{th} neuron in the l^{th} layer can be calculated by:

$$o^l_k = \phi^l_k\left(z^l_k\right) = \phi^l_k\left(\sum_{j=1}^{n^{l-1}} w^l_{kj} \cdot o^{l-1}_j + w^l_{k0} \cdot b^l\right) \tag{3.1}$$

where z^l_k refers to the weighted sum of the neuron inputs, n^{l-1} refers to the number of neurons in the $(l-1)^{th}$ layer, w^l_{kj} refers to the weight of the connection between the j^{th} neuron in the $(l-1)^{th}$ layer and the k^{th} neuron in the l^{th} layer, ϕ^l_k refers to the activation function of the k^{th} neuron in l^{th} layer, b^l refers to the bias in the l^{th} layer and $\cdot$ refers to the scalar product operation. For the input layer thus holds $l = 1$, $y^{1-1}_j = x_j$, $w^1_{kj} = 1$ and $b^1 = 0$.

Figure 3.2 can be treated as a general MLP model. A MLP can have L layers, and each layer l $(1 \leqslant l \leqslant L)$ has n^l $(n^l \geqslant 1)$ neurons. n^1 stands for the input layer and n^L for the output layer. As a common convention between ML practitioners, a neural network must have more than two hidden layers to be called *deep*, hence the name *Deep* Learning for this type of models. For each layer $l \geq 1$ the input dimension is equal to the output dimension of layer $(l-1)$. For a neuron i $(1 \leqslant i \leqslant n^l)$ in layer l, its input vector, weight vector and output are

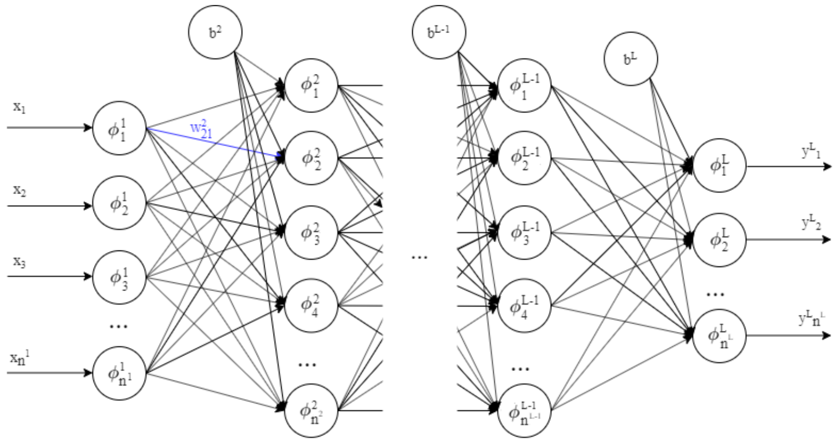


Figure 3.2. General multilayer perceptron structure with L layers. ϕ^i_j represent the activation function of the j^{th} neuron in the i^{th} layer, b^i represent the bias of the i^{th} layer, x_k represent the input variables and y_k represent the output variables.

$\mathbf{yb}^{l-1} = \left(b^l, y^{l-1}_1, \cdots, y^{l-1}_{n^{l-1}}\right)$, $\mathbf{w}^l_i = \left(w^l_{i0}, w^l_{i1}, \cdots, w^l_{in^{l-1}}\right)^\top$ and $y^l_i = \phi^l_i\left(z^l_i\right) = \phi^l_i\left(\mathbf{yb}^{l-1} \cdot \mathbf{w}^l_i\right)$ respectively, where $\phi^l_i : \mathbb{R} \to \mathbb{R}$ refers to the neuron activation function and $\cdot$ refers to the matrix multiplication operator. For each layer l, its input vector is $\mathbf{yb}^{l-1}$, its weight matrix is $\mathbf{W}^l = \left[\mathbf{w}^l_1 \cdots \mathbf{w}^l_{n^l}\right]$ and its output vector is $\mathbf{y}^l = \left(y^l_i, \cdots, y^l_{n^l}\right) = \Phi^l\left(\mathbf{z}^l\right) = \Phi^l\left(\mathbf{yb}^{l-1} \cdot \mathbf{W}^l\right)$, where $\Phi^l : \mathbb{R}^{n^l} \to \mathbb{R}^{n^l}$ is a vector-valued function defined as $\Phi^l(\mathbf{z}) = (\phi^l_1(z_1), \cdots, \phi^l_{n^l}(z_{n^l}))$.

Activation functions of neural networks can be basically divided into 2 types: Linear Activation Functions and Non-linear Activation Functions. Non-linear Activation Functions are the most common because neurons with these activation functions presents superior learning capabilities and facilitate a complex modelling of the input-output relationship. The choice of activation function depends on the specific problem and architecture being used. More information about these activation functions can be found in section 3.2.2.

Weights in the neural structure determine how the information flows from the input layer to the output layer. Identifying the optimal weights that minimize the prediction error is called training the neural network. The process of finding the optimal weights can be seen as an optimization problem, where the metric to minimize is given by a loss function. This loss function shall provide a measure of how similar the predictions of the model are to the desired values for the output (Bishop and Nasrabadi, 2006). A common optimization technique to diminish the loss value is Gradient Descent (Ruder, 2016). The goal of this algorithm is to find the local minimum of a given objective function. This is achieved by iteratively moving closer to the minimum value by taking small steps in the direction indicated by the gradient of the function. The gradient is calculated by determining the partial derivative of the cost function with respect to the neural network's weights. This is done using an algorithm called backward propagation of errors (or backpropagation, for short), which uses the chain rule of calculus to calculate the gradient backward through the layers of a neural network. The weights are then updated simultaneously following equation 3.2 where W^* represents the updated weight values, W represents the current weight values, α is the learning rate which determines how big the steps the algorithm takes into the direction indicated by the gradient, and L represents a loss function:

$$W^* = W - \alpha \frac{\partial L}{\partial W} \tag{3.2}$$

The learning rate is a critical hyperparameter in the training process of neural networks, and finding the optimal value can significantly impact the performance of the model. As depicted in figure 3.3, a small learning rate may result in slow convergence, while a large learning rate may cause the loss function to fluctuate or diverge. Therefore, selecting an appropriate learning rate is crucial for achieving better accuracy and faster convergence in neural network training (Bengio, 2012).

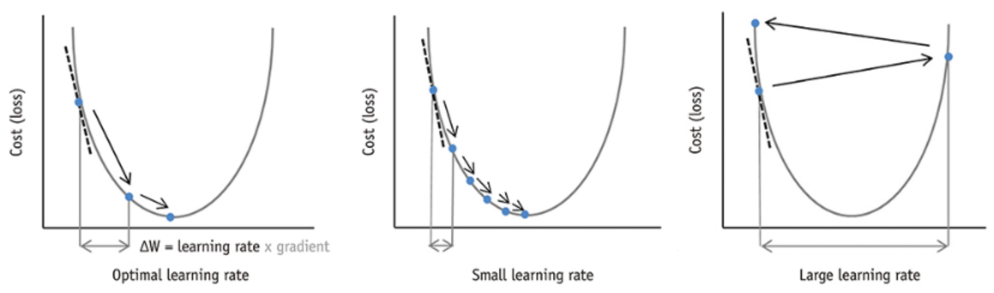


Figure 3.3. Evolution of loss metric during the training of a neural network with different values of learning rate α.

While Gradient Descent is a popular optimization technique for neural networks, there exist many other optimization algorithms that can be used to train a neural network. One such algorithm is Stochastic Gradient Descent (SGD), which is a variant of Gradient Descent that updates the weights after a batch of samples, rather than after computing the gradient for all samples. Convergence analysis reveals that SGD often converges faster than the base Gradient Descent algorithm due to its ability to make frequent weight updates based on batches, effectively navigating through complex loss landscapes. However, SGD may exhibit more oscillations and slower convergence in the final stages due to the inherent noise introduced by using a subset of the data for each update. Another Gradient Descent variant is Adam (Adaptive Moment Estimation), which combines the benefits of both SGD and momentum to achieve faster convergence. RMSprop is another optimizer that adapts the learning rate based on the gradients of the past few iterations. Other optimizers, such as Adagrad, Adadelta, and Nadam, each have their own unique approach to optimizing the weights of a neural network. The choice of optimizer can have a significant impact on the training of a neural network, as different optimizers have their own strengths and weaknesses, and may perform better or worse depending on the specific problem being tackled. For more information about optimizers, we refer the reader to the state-of-the-art reviews in (Sun, 2019; Heidari *et al.*, 2020; Kastrati and Biba, 2021).

Apart from the previous Gradient Descent variants, there are other optimizers that use higher-order derivatives of the loss function. One of these optimization methods is BFGS (Broyden-Fletcher-Goldfarb-Shanno). BFGS belongs to the family of quasi-Newton methods and is known for its efficient convergence properties. Unlike the other gradient-based methods, BFGS approximates the inverse Hessian matrix using gradient information and updates the weights iteratively based on this approximation. This approach allows BFGS to effectively navigate the optimization landscape and converge towards the optimal solution. BFGS has been widely utilized in various domains, including neural network training (K. Liu *et al.*, 2013; Maeda *et al.*, 2014)), and it has demonstrated favorable performance for both small and large-scale optimization problems.

One potential issue that can arise during the training of a neural network is overfitting, where the model performs well on the training data but poorly on new, unseen data. This can occur when the model becomes too complex and starts to fit to noise in the data rather than the underlying patterns. To address this issue, a regularization technique known as weight decay can be employed. Weight decay adds a penalty term to the loss function, which discourages the weights from taking on large values (Goodfellow *et al.*, 2016). This penalty term is usually proportional to the square of the magnitude of the weights, and its coefficient is controlled by a hyperparameter called the regularization strength. By adding this penalty term to the loss function, the model is incentivized to use smaller weights, which in turn reduces the complexity of the model and helps prevent overfitting.

Having now comprehensively explained the structure and function of the Multilayer Perceptron (MLP) architecture, following sections describe how to calculate the partial derivatives of the output variable with respect to the inputs that are used by the methodologies developed in this thesis. The calculus of these partial derivatives is fundamentally tied to the nature of activation functions employed within the MLP. Consequently, the following section will delineate the calculation of partial derivatives of various activation functions.

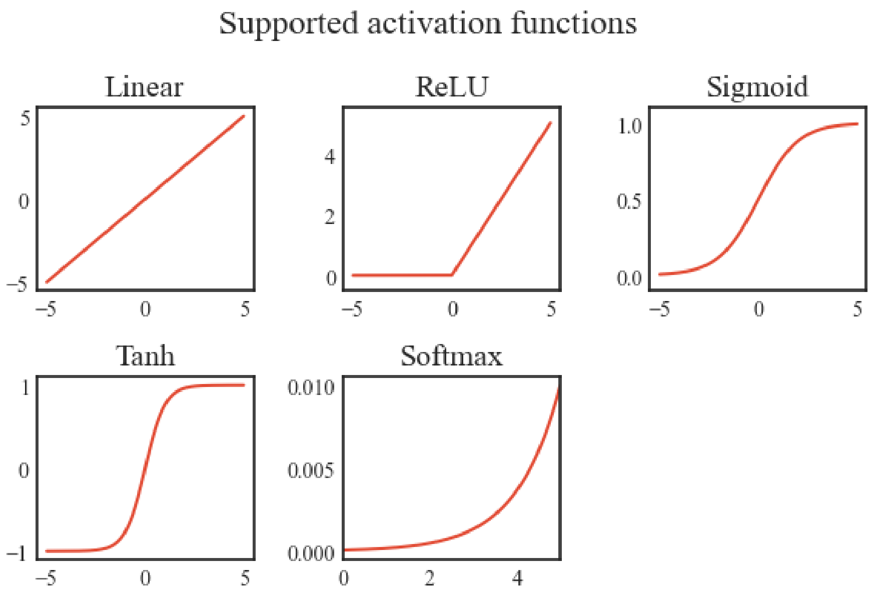


Figure 3.4. Activation functions supported in `neuralsens` package.

3.2.2. Partial Derivatives of Activation functions

In this section, we provide the expressions to calculate the first, second, and third partial derivatives of the output with respect to the inputs of the most common activation functions used in artificial neural networks. All this derivatives are later used to calculate the first, second and third partial derivatives of an MLP. The activation functions supported by our package are the Sigmoid, Linear, Hyperbolic Tangent (tanh), Rectified Linear Unit (ReLU), and Softmax functions. table 3.1 collects the expression and derivatives of the activation functions supported by the `neuralsens` package, and figure 3.4 shows the output of each of these function for different values of the input variable.

Table 3.1. Summary of the expressions, first, second, and third partial derivatives of common activation functions supported in the `neuralsens` package.

Activation Function	Expression	First Derivative
sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$
Linear	$I(x) = x$	$I'(x) = 1$
Tanh	$tanh(x) = th(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$th'(x) = 1 - th^2(x)$
ReLU	$ReLU(x) = \max(0, x)$	$ReLU'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$
Softmax	$Softmax(x)_i = Sf(x)_i = \frac{e^{x_i}}{\sum e^{x_j}}$	$\frac{\partial Sf(x_i)}{\partial x_j} = \begin{cases} Sf(x)_i \cdot (1 - Sf(x)_i) & \text{if } i = j \\ -Sf(x)_i \cdot Sf(x)_j & \text{if } i \neq j \end{cases}$

Activation Function	Second Derivative	Third Derivative
sigmoid	$\sigma''(x) = \sigma'(x) \cdot (1 - 2\sigma(x))$	$\sigma'''(x) = \sigma''(x) \cdot (1 - 3\sigma(x)) + \sigma(x) \cdot (\sigma(x) - 1)^2$
Linear	$I''(x) = 0$	$I'''(x) = 0$
Tanh	$th''(x) = -2 \cdot th(x) \cdot th'(x)$	$\sigma'''(x) = 2 \cdot (th^2(x) - 1) \cdot th'(x) + th'(x) + 4 \cdot th(x) \cdot th''(x)$
ReLU	$ReLU''(x) = 0$	$ReLU'''(x) = 0$
Softmax	Equation 3.3 with $n = 2$	Equation 3.3 with $n = 3$

For the softmax function, second and third partial derivatives are calculated using the following formula for the n^{th} partial derivative with $n = 2$ and $n = 3$ respectively:

$$\frac{\partial^n Sf(x)_i}{\partial x_1 \partial x_2 ... \partial x_n} = \sum_{\substack{S \subseteq \{1,2,...,n\} \\ |S|=n}} \prod_{t=1}^{n} (\delta_{it} - Sf(x)_t) \prod_{s \in S} Sf(x)_s \tag{3.3}$$

where S is a subset of indices of size n, δ_{it} is the Kronecker delta, and $Sf(x)_s$ denotes the softmax function evaluated at x_s.

When using the `neuralsens` package, all neurons in the same layer are supposed to use the same activation function. However, custom activation functions can be used if the derivatives of the activation function are passed to the methods of the `neuralsens` package. These derivatives shall return arrays of the appropriate size for further operations. Being $\Phi^l : \mathbb{R}^{n^l} \to \mathbb{R}^{n^l}$ the activation function of the l^{th} layer where n^l is the number of neurons in the l^{th} layer of the neural network as defined in section 3.2.1, we expect the output of the first, second and third partial derivative function evaluated in the $\mathbf{z}^l$ input of the l^{th} layer to be an array defined as:

1. First partial derivatives:

$$\frac{\nabla \mathbf{\Phi}^l}{\nabla \mathbf{z}^l} \left(\mathbf{z}^l\right)_{[n^l \times n^l]} = \left[\frac{\nabla \phi_1^l}{\nabla \mathbf{z}^l} \left(\mathbf{z}^l\right), \ldots, \frac{\nabla \phi_{n^l}^l}{\nabla \mathbf{z}^l} \left(\mathbf{z}^l\right) \right]_{[n^l \times n^l]}$$

2. Second partial derivatives:

$$\frac{\nabla^2 \mathbf{\Phi}^l}{\nabla^2 \mathbf{z}^l}\left(\mathbf{z}^l\right)_{[n^l \times n^l \times n^l]} = \left[\frac{\nabla^2 \phi_1^l}{\nabla^2 \mathbf{z}^l}\left(\mathbf{z}^l\right), \ldots, \frac{\nabla^2 \phi_{n^l}^l}{\nabla^2 \mathbf{z}^l}\left(\mathbf{z}^l\right)\right]_{[n^l \times n^l \times n^l]}$$

3. Third partial derivatives:

$$\frac{\nabla^3 \mathbf{\Phi}^l}{\nabla^3 \mathbf{z}^l}\left(\mathbf{z}^l\right)_{[n^l \times n^l \times n^l \times n^l]} = \left[\frac{\nabla^3 \phi_1^l}{\nabla^3 \mathbf{z}^l}\left(\mathbf{z}^l\right), \ldots, \frac{\nabla^3 \phi_{n^l}^l}{\nabla^3 \mathbf{z}^l}\left(\mathbf{z}^l\right)\right]_{[n^l \times n^l \times n^l \times n^l]}$$

As long as the user-defined function return a suitable array, these functions can be handled by the internal functions of the package and use the methods described in sections 3.3, 3.4 and 3.5. In the subsequent sections, the derivatives of the activation functions previously described are used to compute the partial derivatives between different layers of a MLP model.

3.2.3. First Partial Derivatives

First partial derivatives of the output variables with respect to the input variables, also called sensitivities, provide valuable insights into the sensitivity of a model's output to changes in its input variables. More information about how to exploit the information from first partial derivatives is found in sections 3.3 and 3.4.

For a MLP model with L layers, first partial derivatives are defined as:

$$s_{i,k}\big|_{\mathbf{x}_p} = \frac{\partial o_k^L}{\partial x_i}\left(\mathbf{x}_p\right) \tag{3.4}$$

where $\mathbf{x}_p$ refers to the p^{th} sample of the dataset used to perform the sensitivity analysis and $s_{i,k}\big|_{\mathbf{x}_p}$ refers to the sensitivity of the output of the k^{th} neuron in the output layer with regard to the input of the i^{th} neuron in the input layer evaluated in $\mathbf{x}_p$. We calculate these sensitivities applying the chain rule to the partial derivatives of the inner layers (derivatives of Equation 3.1 for each neuron in the hidden layers).

The partial derivatives of the inner layers are defined following the next equations:

- Derivative of z_k^l (weighted linear input combination of the k^{th} neuron in the l^{th} layer) with regard to o_i^{l-1} (output of the i^{th} neuron in the $(l-1)^{th}$ layer). This partial derivative corresponds to the weight of the connection between the k^{th} neuron in the l^{th} layer and the i^{th} neuron in the $(l-1)^{th}$ layer:

$$\frac{\partial z_k^l}{\partial o_i^{l-1}} = w_{ki}^l \tag{3.5}$$

- Derivative of o_k^l (output of the the k^{th} neuron in the l^{th} layer) with regard to z_i^l (input of the i^{th} neuron in the l^{th} layer):

$$\left.\frac{\partial o_k^l}{\partial z_i^l}\right|_{z_i^l} = \frac{\partial \phi_k^l}{\partial z_i^l}\left(z_i^l\right) \tag{3.6}$$

where $\frac{\partial \phi_k^l}{\partial z_i^l}$ refers to the partial derivative of the activation function of the k^{th} neuron in the l^{th} layer with regard to the input of the k^{th} neuron in the l^{th} layer evaluated for the input z_i^l of the i^{th} neuron in the l^{th} layer.

Equations 3.5 and 3.6 have been implemented in the package in matrix form to reduce computational time following the next equations:

$$\mathbf{W}^{*l}_{[n^{l-1}\times n^l]} = \frac{\partial \mathbf{z}^l_{[1\times n^l]}}{\partial \mathbf{o}^{l-1}_{[1\times n^{l-1}]}} = \begin{bmatrix} \frac{\partial z_1^l}{\partial o_1^{l-1}} & \frac{\partial z_2^l}{\partial o_1^{l-1}} & \cdots & \frac{\partial z_{n^l}^l}{\partial o_1^{l-1}} \\ \frac{\partial z_1^l}{\partial o_2^{l-1}} & \frac{\partial z_2^l}{\partial o_2^{l-1}} & \cdots & \frac{\partial z_{n^l}^l}{\partial o_2^{l-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_1^l}{\partial o_{n^{l-1}}^{l-1}} & \frac{\partial z_2^l}{\partial o_{n^{l-1}}^{l-1}} & \cdots & \frac{\partial z_{n^l}^l}{\partial o_{n^{l-1}}^{l-1}} \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{21}^l & \cdots & w_{n^l1}^l \\ w_{12}^l & w_{22}^l & \cdots & w_{n^l2}^l \\ \vdots & \vdots & \ddots & \vdots \\ w_{1n^{l-1}}^l & w_{2n^{l-1}}^l & \cdots & w_{n^ln^{l-1}}^l \end{bmatrix}$$

$$\mathbf{J}^l_{l_{[n^l\times n^l]}} = \frac{\partial \mathbf{o}^l_{[1\times n^l]}}{\partial \mathbf{z}^l_{[1\times n^l]}} = \begin{bmatrix} \frac{\partial o_1^l}{\partial z_1^l} & \frac{\partial o_2^l}{\partial z_1^l} & \cdots & \frac{\partial o_{n^l}^l}{\partial z_1^l} \\ \frac{\partial o_1^l}{\partial z_2^l} & \frac{\partial o_2^l}{\partial z_2^l} & \cdots & \frac{\partial o_{n^l}^l}{\partial z_2^l} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial o_1^l}{\partial z_{n^l}^l} & \frac{\partial o_2^l}{\partial z_{n^l}^l} & \cdots & \frac{\partial o_{n^l}^l}{\partial z_{n^l}^l} \end{bmatrix} = \begin{bmatrix} \frac{\partial \phi_1^l}{\partial z_1^l}\left(z_1^l\right) & \frac{\partial \phi_2^l}{\partial z_1^l}\left(z_1^l\right) & \cdots & \frac{\partial \phi_{n^l}^l}{\partial z_1^l}\left(z_1^l\right) \\ \frac{\partial \phi_1^l}{\partial z_2^l}\left(z_2^l\right) & \frac{\partial \phi_2^l}{\partial z_2^l}\left(z_2^l\right) & \cdots & \frac{\partial \phi_{n^l}^l}{\partial z_2^l}\left(z_2^l\right) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \phi_1^l}{\partial z_{n^l}^l}\left(z_{n^l}^l\right) & \frac{\partial \phi_2^l}{\partial z_{n^l}^l}\left(z_{n^l}^l\right) & \cdots & \frac{\partial \phi_{n^l}^l}{\partial z_{n^l}^l}\left(z_{n^l}^l\right) \end{bmatrix}$$

where $\mathbf{W}^{*l}$ is the reduced weight matrix of the l^{th} layer and $\mathbf{J}^l_l$ is the Jacobian matrix of the outputs in the l^{th} layer with regard to the inputs in the l^{th} layer.

Following the chain rule, the Jacobian matrix of the outputs in the l^{th} layer with regard to the inputs in the p^{th} layer can be calculated by:

$$\mathbf{J}^l_{p_{[n^p\times n^l]}} = \mathbf{J}^{l-1}_{p_{[n^p\times n^{l-1}]}} \otimes \mathbf{W}^{*l}_{[n^{l-1}\times n^l]} \otimes \mathbf{J}^l_{l_{[n^l\times n^l]}} \tag{3.7}$$

where $\otimes$ refers to the tensor product operation, $1 \leqslant p \leqslant (l-1)$ and $2 \leqslant l \leqslant L$.

Using equation 3.7 with $l = L$ and $p = 1$, the partial derivatives of the outputs with regard to the inputs of the MLP are obtained.

3.2.4. Second Partial Derivatives

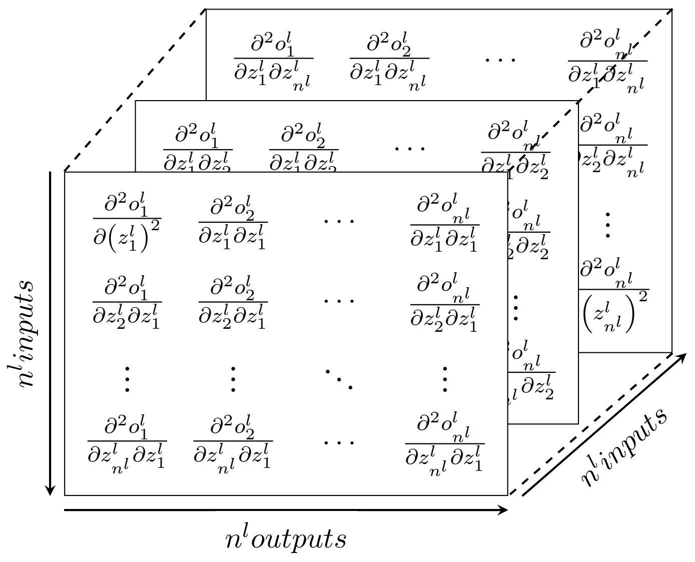
The first order sensitivities give information only about the relationship between one output and one input. In order to obtain information about the relationship between output and the interaction of two input variables (or the curvature of the relationship between output and one input variable), the partial derivatives method could be extended calculating the second partial derivatives:

$$\begin{gathered} s_{ij,k}|_{\mathbf{x}_n} = \frac{\partial^2 o_k^L}{\partial x_i \partial x_j}(\mathbf{x}_n) = \frac{\partial}{\partial x_j}\left(\frac{\partial o_k^L}{\partial x_i}\right)(\mathbf{x}_n) = \\ \frac{\partial}{\partial x_j}\left(\sum_{p=1}^{n^L}\left(\frac{\partial o_k^L}{\partial z_p^L}\cdot\frac{\partial z_p^L}{\partial x_i}\right)\right)(\mathbf{x}_n) \end{gathered} \tag{3.8}$$

Using the product rule of the derivatives:

$$\begin{aligned}\frac{\partial}{\partial x_j}\left(\sum_{p=1}^{n^L}\left(\frac{\partial o_k^L}{\partial z_p^L}\cdot\frac{\partial z_p^L}{\partial x_i}\right)\right)(\mathbf{x}_n) = \\ \sum_{p=1}^{n^L}\left(\frac{\partial^2 o_k^L}{\partial\left(z_p^L\right)^2}\cdot\frac{\partial z_p^L}{\partial x_i}\cdot\frac{\partial z_p^L}{\partial x_j}+\frac{\partial o_k^L}{\partial z_p^L}\cdot\frac{\partial^2 z_p^L}{\partial x_i \partial x_j}\right)(\mathbf{x}_n)\end{aligned} \tag{3.9}$$

These second derivatives can also be computed using the chain rule with the second derivatives of the inner layers, using the Hessian $(\mathbf{H})$ array, the Jacobian $(\mathbf{J})$ and weight $(\mathbf{W})$ matrixes. The Hessian array for a given layer $\mathbf{H}_l^l = \frac{\partial^2 \mathbf{o}^l_{[1\times n^l]}}{\partial \mathbf{z}^l_{[1\times n^l]}\partial \mathbf{z}^l_{[1\times n^l]}}$ has the following structure:



$\frac{\partial^2 o_1^l}{\partial (z_1^l)^2}$ $\frac{\partial^2 o_2^l}{\partial z_1^l \partial z_1^l}$ $\cdots$ $\frac{\partial^2 o_{n^l}^l}{\partial z_1^l \partial z_1^l}$
$\frac{\partial^2 o_1^l}{\partial z_2^l \partial z_1^l}$ $\frac{\partial^2 o_2^l}{\partial z_2^l \partial z_1^l}$ $\cdots$ $\frac{\partial^2 o_{n^l}^l}{\partial z_2^l \partial z_1^l}$
$\frac{\partial^2 o_1^l}{\partial z_{n^l}^l \partial z_1^l}$ $\frac{\partial^2 o_2^l}{\partial z_{n^l}^l \partial z_1^l}$ $\cdots$ $\frac{\partial^2 o_{n^l}^l}{\partial z_{n^l}^l \partial z_1^l}$
$\frac{\partial^2 o_1^l}{\partial z_1^l \partial z_{n^l}^l}$ $\frac{\partial^2 o_2^l}{\partial z_1^l \partial z_{n^l}^l}$ $\cdots$ $\frac{\partial^2 o_{n^l}^l}{\partial z_1^l \partial z_{n^l}^l}$
$\frac{\partial^2 o_{n^l}^l}{\partial (z_{n^l}^l)^2}$
$n^l inputs$
$n^l outputs$
$n^l inputs$

Applying the chain rule, the Hessian array of the outputs in the l^{th} layer with regard to the inputs in the p^{th} layer can be calculated using:

$$\begin{aligned}\mathbf{H}^l_{p_{[n^p\times n^p\times n^l]}} = \left(\mathbf{J}^{l-1}_{p_{[n^p\times n^{l-1}]}} \otimes \mathbf{W}^{*l}_{[n^{l-1}\times n^l]}\right) \otimes_i \left(\mathbf{J}^{l-1}_{p_{[n^p\times n^{l-1}]}} \otimes \mathbf{W}^{*l}_{[n^{l-1}\times n^l]}\right) \otimes_j \mathbf{H}^l_{l_{[n^l\times n^l\times n^l]}} + \\ \mathbf{H}^{l-1}_{p_{[n^p\times n^p\times n^{l-1}]}} \otimes_k \mathbf{W}^{*l}_{[n^{l-1}\times n^l]} \otimes_k \mathbf{J}^l_{l_{[n^l\times n^l]}}\end{aligned} \tag{3.10}$$

where $\otimes_a$ is the tensor multiplication operator applied to a n-dimensional array along the a axis, $1 \leqslant p \leqslant (l-1)$ and $2 \leqslant l \leqslant L$. Using equation 3.10 with $l = L$ and $p = 1$ the second partial derivatives of the output with regard to the inputs of the MLP are obtained.

3.2.5. Third derivatives

As stated in the previous chapter, the hessian array $(\mathbf{H})$ can be used to detect interaction between a pair of input variables. Further developments to detect interactions need the third partial derivatives to be efficiently computed (see section 3.5). These third derivatives are defined as:

$$s_{ijm,k}|_{\mathbf{x}_n} = \frac{\partial^3 o_k^L}{\partial x_i \partial x_j \partial x_m}(\mathbf{x}_n) = \left(\frac{\partial}{\partial x_m}\left(\frac{\partial}{\partial x_j}\left(\frac{\partial o_k^L}{\partial x_i}\right)\right)\right)(\mathbf{x}_n) \tag{3.11}$$

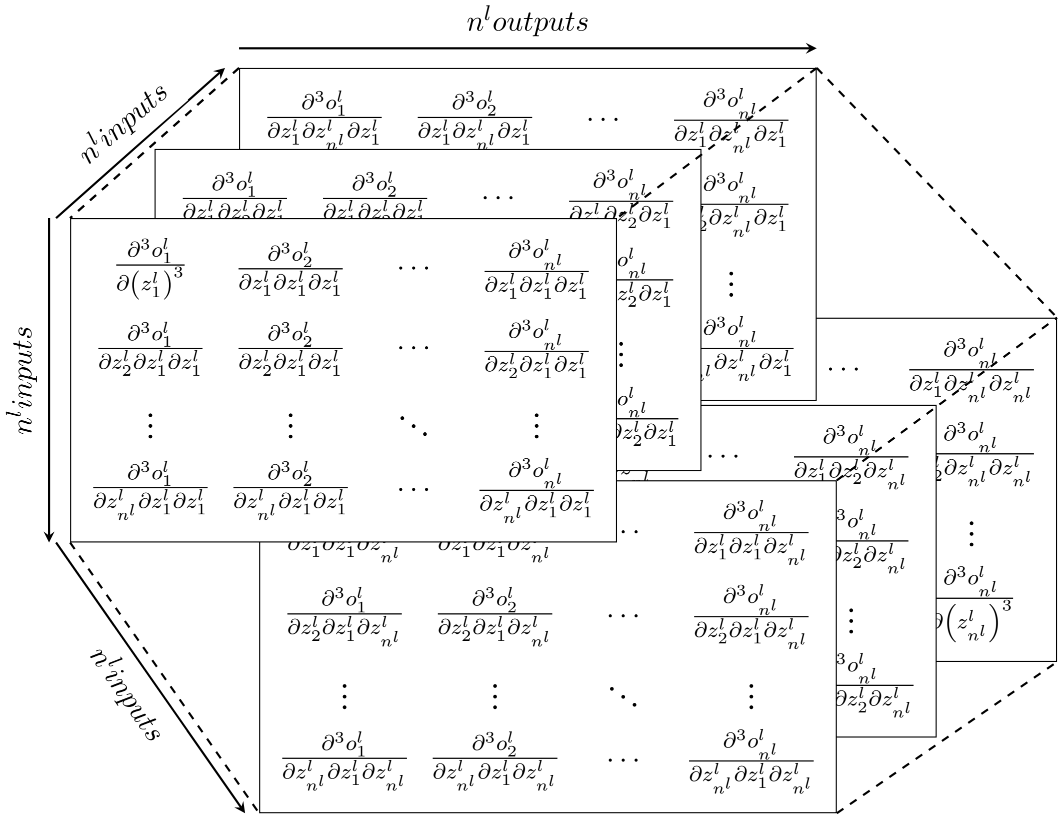
Using again chain rule and the product rule of the derivatives, the third partial derivatives can be obtained:

$$\frac{\partial^3 o_k^l}{\partial x_i \partial x_j \partial x_m}(\mathbf{x}_n) = \frac{\partial}{\partial x_m}\left(\frac{\partial^2 o_k^L}{\partial x_i \partial x_j}\right)(\mathbf{x}_n) = \frac{\partial}{\partial x_m}\left(\sum_{p=1}^{n^L}\left(\frac{\partial o_k^L}{\partial z_p^L}\cdot\frac{\partial^2 z_p^L}{\partial x_i \partial x_j}\right)\right)(\mathbf{x}_n) =$$
$$\sum_{p'=1}^{n^{l-1}}\left(\frac{\partial^3 o_k^l}{\partial\left(z_k^l\right)^3}\cdot\frac{\partial z_k^l}{\partial x_i}\cdot\frac{\partial z_k^l}{\partial x_j}\cdot\frac{\partial z_k^l}{\partial x_m}+\frac{\partial o_k^l}{\partial z_k^l}\cdot\frac{\partial^3 z_k^l}{\partial x_i \partial x_j \partial x_m}+\right.$$
$$\left.\frac{\partial^2 o_k^l}{\partial\left(z_k^l\right)^2}\cdot\frac{\partial z_k^l}{\partial x_i}\cdot\frac{\partial^2 z_k^l}{\partial x_j \partial x_m}+\frac{\partial^2 o_k^l}{\partial\left(z_k^l\right)^2}\cdot\frac{\partial z_k^l}{\partial x_j}\cdot\frac{\partial^2 z_k^l}{\partial x_i \partial x_m}+\frac{\partial^2 o_k^l}{\partial\left(z_k^l\right)^2}\cdot\frac{\partial z_k^l}{\partial x_m}\cdot\frac{\partial^2 z_k^l}{\partial x_i \partial x_j}\right) \tag{3.12}$$

where:

$$\frac{\partial^3 z_p^l}{\partial x_i \partial x_j \partial x_m} = \sum_{p'=1}^{n^{l-1}}\left(\frac{\partial z_p^l}{\partial o_{p'}^{l-1}}\cdot\frac{\partial^3 o_{p'}^{l-1}}{\partial x_i \partial x_j \partial x_m}\right) \tag{3.13}$$

The third derivatives can be computed using the chain rule with the third derivatives of the inner layers, using the Jerkian ($\mathbf{K}$) and Hessian ($\mathbf{H}$) arrays, and the Jacobian ($\mathbf{J}$) and weight ($\mathbf{W}$) matrixes. We define the Jerkian array for a given layer as $\mathbf{K}_l^l = \frac{\partial^3 o_{[1\times n^l]}^l}{\partial z_{[1\times n^l]}^l \partial z_{[1\times n^l]}^l \partial z_{[1\times n^l]}^l}$, which has the following structure:


$n^l outputs$
$n^l inputs$
$n^l inputs$
$n^l inputs$
$\frac{\partial^3 o_1^l}{\partial(z_1^l)^3}$
$\frac{\partial^3 o_2^l}{\partial z_1^l \partial z_1^l \partial z_1^l}$
$\frac{\partial^3 o_{n^l}^l}{\partial z_1^l \partial z_1^l \partial z_1^l}$
$\frac{\partial^3 o_1^l}{\partial z_2^l \partial z_1^l \partial z_1^l}$
$\frac{\partial^3 o_{n^l}^l}{\partial z_{n^l}^l \partial z_1^l \partial z_1^l}$
$\frac{\partial^3 o_1^l}{\partial z_{n^l}^l \partial z_1^l \partial z_{n^l}^l}$
$\frac{\partial^3 o_{n^l}^l}{\partial z_{n^l}^l \partial z_1^l \partial z_{n^l}^l}$
$\frac{\partial^3 o_{n^l}^l}{\partial(z_{n^l}^l)^3}$

Applying the chain rule, the Jerkian array of the outputs in the l^{th} layer with regard to the inputs in the p^{th} layer can be calculated using:

$$\begin{aligned}
\mathbf{K}^{l}_{p_{[n^p\times n^p\times n^p\times n^l]}} = & \left(\mathbf{J}^{l-1}_{p_{[n^p\times n^{l-1}]}} \otimes \mathbf{W}^{l}_{[n^{l-1}\times n^l]}\right) \otimes_i \left(\mathbf{J}^{l-1}_{p_{[n^p\times n^{l-1}]}} \otimes \mathbf{W}^{l}_{[n^{l-1}\times n^l]}\right) \otimes_j \\
& \left(\mathbf{J}^{l-1}_{p_{[n^p\times n^{l-1}]}} \otimes \mathbf{W}^{l}_{[n^{l-1}\times n^l]}\right) \otimes_m \mathbf{K}^{l}_{l_{[n^l\times n^l\times n^l\times n^l]}} + \\
& \left(\mathbf{J}^{l-1}_{p_{[n^p\times n^{l-1}]}} \otimes \mathbf{W}^{l}_{[n^{l-1}\times n^l]}\right) \otimes_i \left(\mathbf{H}^{l-1}_{p_{[n^p\times n^p\times n^{l-1}]}} \otimes_k \mathbf{W}^{l}_{[n^{l-1}\times n^l]}\right) \otimes_{jm} \mathbf{H}^{l}_{l_{[n^l\times n^l\times n^l]}} + \\
& \left(\mathbf{J}^{l-1}_{p_{[n^p\times n^{l-1}]}} \otimes \mathbf{W}^{l}_{[n^{l-1}\times n^l]}\right) \otimes_j \left(\mathbf{H}^{l-1}_{p_{[n^p\times n^p\times n^{l-1}]}} \otimes_k \mathbf{W}^{l}_{[n^{l-1}\times n^l]}\right) \otimes_{im} \mathbf{H}^{l}_{l_{[n^l\times n^l\times n^l]}} + \\
& \left(\mathbf{J}^{l-1}_{p_{[n^p\times n^{l-1}]}} \otimes \mathbf{W}^{l}_{[n^{l-1}\times n^l]}\right) \otimes_m \left(\mathbf{H}^{l-1}_{p_{[n^p\times n^p\times n^{l-1}]}} \otimes_k \mathbf{W}^{l}_{[n^{l-1}\times n^l]}\right) \otimes_{ij} \mathbf{H}^{l}_{l_{[n^l\times n^l\times n^l]}} + \\
& \mathbf{K}^{l-1}_{p_{[n^p\times n^p\times n^p\times n^{l-1}]}} \otimes_k \mathbf{W}^{l}_{[n^{l-1}\times n^l]} \otimes_k \mathbf{J}^{l}_{l_{[n^l\times n^l]}}
\end{aligned} \tag{3.14}$$

where $\otimes_{ab}$ is the tensor multiplication operation along the a and b axis of a n-dimensional array$1 \leqslant p \leqslant (l-1)$ and $2 \leqslant l \leqslant L$. Using equation 3.14 with $l = L$ and $p = 1$ the third partial derivatives of the output with red to the inputs of the MLP are obtained.

3.2.6. Comparison with automatic differentiation

In the course of examining the analytical computation of first, second, and third partial derivatives of Multi-Layer Perceptron (MLP) models, a pertinent discussion emerges regarding the feasibility and efficiency of employing automatic differentiation engines, such as Pytorch's `autograd` (Paszke *et al.*, 2019), for this task. At a glance, automatic differentiation presents a straightforward avenue for derivative computation. However, a closer inspection reveals significant drawbacks, particularly when dealing with higher-order derivatives.

To furnish a clear comparison and substantiate the rationale behind opting for analytical calculations over automatic differentiation, an empirical assessment was conducted. This assessment entailed the computation of the first, second and third partial derivatives of an MLP model, varying both the number of neurons in the hidden layer and the number of samples to be analyzed. The objective was to gauge the computational times and thereby, the efficiency of both methods under different configurations.

The configurations tested included:

- Number of inputs: The number of inputs to the model was varied from 5 to 35 in increments of 5 inputs.
- Number of neurons in hidden layer: The configurations tested had 10, 30, and 50 neurons in the hidden layer.
- Number of samples: The partial derivatives were calculated for 1000, 5000, and 10000 samples.

Figure 3.5 shows the computational time required to calculate the first, second and third partial derivatives using `autograd` and the analytical methods developed in our research using the `neuralsens` package. For the sake of clarity, it only shows the computational time of the first, second and third partial derivatives for the configurations tested with 10000 samples. The rest of the tested configurations can be found in Annex A.

Results unequivocally showcase the superior efficiency of analytical calculations, especially as the complexity of the model escalated. Automatic differentiation, while adept at handling

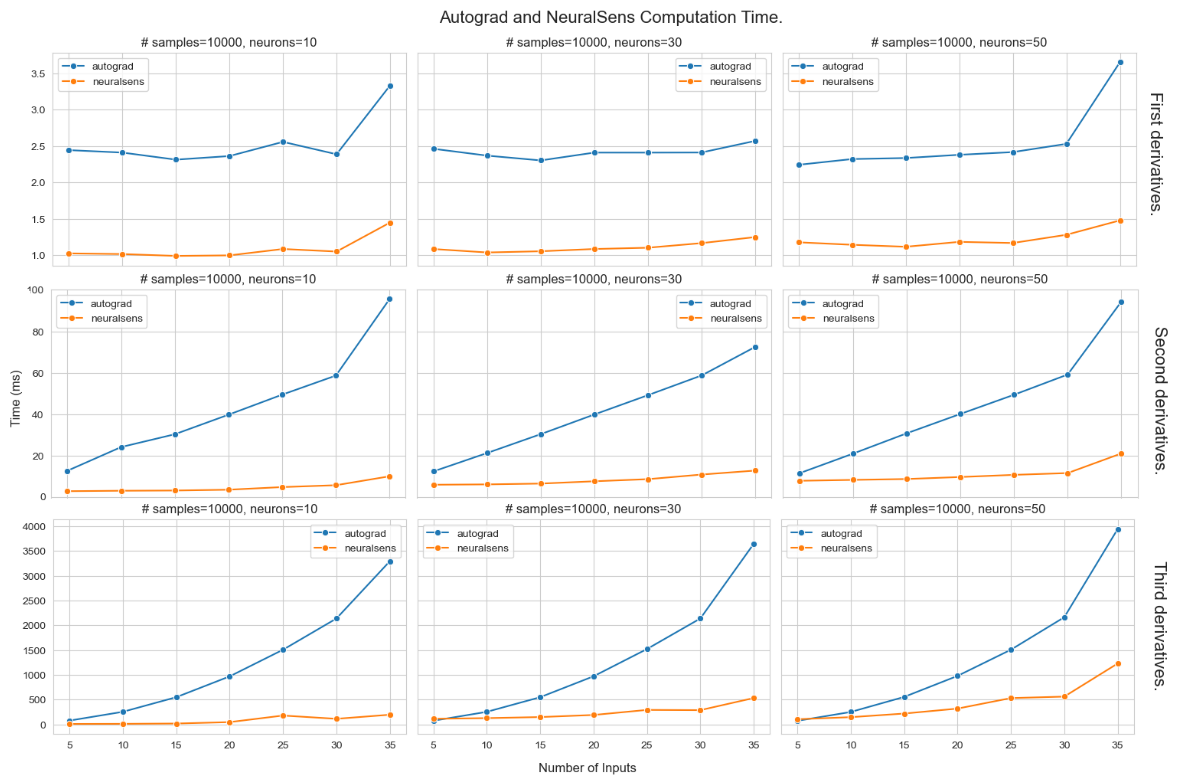


Figure 3.5. Comparative analysis of computation times for the calculation of first, second, and third partial derivatives utilizing analytical calculations (`neuralsens`) and automatic differentiation (`autograd`) amidst varying model complexities, with a constant sample size of 10000.

lower order derivatives, manifested a noticeable slowdown when confronted with higher-order derivatives. This sluggish performance not only augments the computational times but can potentially impede real-time analysis and timely decision-making, which is often crucial in time-sensitive applications. Such delays may deter users from undertaking the analysis due to time constraints, thereby impacting the overall utility and application of the model in practical scenarios.

3.3. Sensitivity Analysis

First partial derivatives provide valuable insights into the sensitivity of a model's output to changes in its input variables. By examining the sensitivities of MLPs, we aim to uncover deeper insights into the structure and function of these models, which will enable us to enhance interpretability and ultimately improve the performance of these networks in real-world tasks.

Once the first order sensitivity has been obtained for each variable and observation, different measures can be calculated to analyze the model's functioning. We propose the following sensitivity measures to summarize the information obtained by evaluating the sensitivity of the outputs for all the input samples $\mathcal{X} = \{\mathbf{x}_j\}_{j=1}^{N}$ of a given dataset:

- *Mean sensitivity* of the output of the k^{th} neuron in the output layer with regard to the i^{th} input variable:

$$S_{i,k}^{avg} = \frac{\sum_{n=1}^{N} s_{i,k}\big|_{\mathbf{x}_n}}{N} \tag{3.15}$$

  where N is the number of samples in the dataset.

- *Sensitivity standard deviation* of the output of the k^{th} neuron in the output layer with regard to the i^{th} input variable:

$$S_{i,k}^{sd} = \sigma\left(s_{i,k}\big|_{\mathbf{x}_n}\right); n \in 1, \ldots, N \tag{3.16}$$

  where N is the number of samples in the dataset and σ refers to the standard deviation function.

- *Mean squared sensitivity* of the output of the k^{th} neuron in the output layer with regard to the i^{th} input variable Yeh and Cheng, 2010; Zurada *et al.*, 1994; White and Racine, 2001:

$$S_{i,k}^{sq} = \sqrt{\frac{\sum_{n=1}^{N}\left(s_{i,k}\big|_{\mathbf{x}_n}\right)^2}{N}} \tag{3.17}$$

  where N is the number of samples in the dataset.

In case there are more than one output neuron, such as in a multi-class classification problem, these measures can be generalized to obtain sensitivity measures of the whole model as follows:

- *Mean sensitivity* with regard to the i^{th} input variable:

$$S_i^{avg} = \frac{\sum_{k=1}^{n^L} S_{i,k}^{avg}}{n^L} \tag{3.18}$$

- *Sensitivity standard deviation* with regard to the i^{th} input variable:

$$S_i^{sd} = \sqrt{\frac{\sum_{k=1}^{n^L}\left(\left(S_{i,k}^{sd}\right)^2 + \left(S_{i,k}^{avg} - S_i^{avg}\right)^2\right)}{n^L}} \tag{3.19}$$

- *Mean squared sensitivity* with regard to the i^{th} input variable (Yeh and Cheng, 2010):

$$S_i^{sq} = \left(\frac{\sum_{k=1}^{n^L}\sqrt{S_{i,k}^{sq}}}{n^L}\right)^2 \tag{3.20}$$

3.3.1. Interpretation of Sensitivity Measures

In order to provide a better explanation of how the sensitivity measures can be applied, we must review the relationship between linear regression and its derivatives. Let $f(x)$ be a linear regression model with $\hat{\beta}$ the estimated coefficient for the continuous input variable x which follows the next expression:

$$f(x) = \hat{y} = \hat{\beta}_1 \cdot x + \beta_0 \tag{3.21}$$

with input variable x. Derivative of the output with regard to the input is defined as:

$$f'(x) = \frac{\partial \hat{y}}{\partial x} = \hat{\beta}_1 \tag{3.22}$$

which is constant for all points in the input space. This derivative value indicates how much the predicted output changes for each unit change in the input variable x. In other words, the partial derivative provides a constant measure that characterizes the linear relationship between the input x and the output prediction $\hat{y}$. For instance, if $f'(x) = 2.5$, it implies that for every one-unit increase in the input x, the predicted output $\hat{y}$ will increase by 2.5 units.

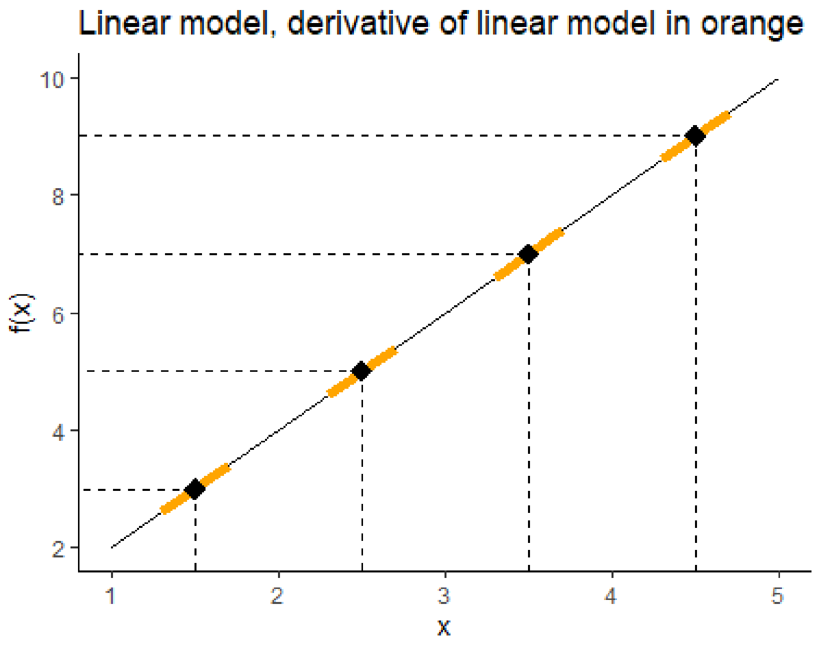


Figure 3.6. Linear model with equation $f(x) = \hat{\beta} \cdot x$, derivative of linear model in orange. Derivative is constant for all points in the input space.

Figure 3.6 represents a linear model with $\hat{\beta} = 2$ and its derivative with 4 different points of the input space. If we apply the measures proposed in section 3.3 to the distribution of the derivatives of the linear model obtained by evaluating the derivative in all the points of the input dataset, we shall obtain:

$$S_{1,1}^{avg} = \hat{\beta} = 2; \qquad S_{1,1}^{sd} = 0 \tag{3.23}$$

Consequently, we shall expect that, for any MLP model with a linear relationship between an output and an input variable, the proposed measures would have values in the form of:

$$S_{1,1}^{avg} = k; \quad k \in \mathbb{R} \qquad S_{1,1}^{sd} \approx 0 \tag{3.24}$$

Following with the previous example, let $f(\mathbf{x})$ be a non-linear regression model which follows the next expression:

$$f(\mathbf{x}) = y = \beta_1 \cdot x_1 + \beta_2 \cdot (x_2)^2 + \beta_0; \quad k_1, k_2 \in \mathbb{R} \tag{3.25}$$

with 3 continuous input variables x_1, x_2 and x_3, and k_1 and k_2 estimated coefficients during the training process of the model. From the expression, we can conclude that x_1 has a linear relationship with the output, x_2 has a non-linear relationship with the output and x_3 has no relationship with the output. Derivatives of the non-linear regressor output variable with regard to its input variables are defined as:

$$\frac{\partial y}{\partial x_1} = \beta_1; \qquad \frac{\partial y}{\partial x_2} = 2 \cdot \beta_2 \cdot x_2; \qquad \frac{\partial y}{\partial x_3} = 0 \tag{3.26}$$

for all points in the input space.

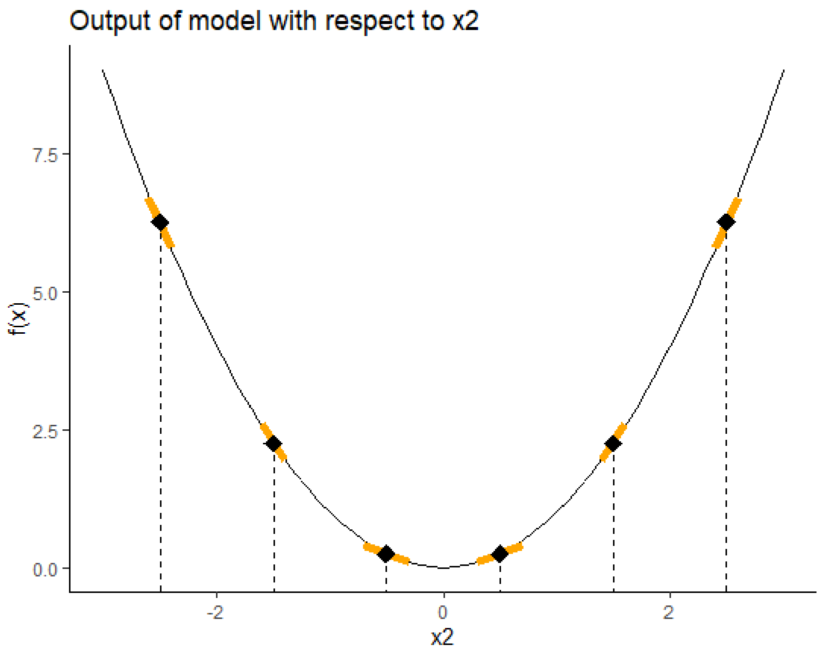


Figure 3.7. Non-linear model with equation $f(x) = x^2$, derivative of non-linear model in orange. Derivative depends on the value of x.

Figure 3.7 shows the output of a model with $k_2 = 1$ for different values of x_2 for a constant value of x_1. As can be seen in the figure, derivative with regard to this variable is not constant along the input space. The consequences in the sensitivity measures we proposed can be summarized as:

$$S_{2,1}^{sd} >> 0 \tag{3.27}$$

With regard to x_1, the output of the model would be similar to figure 3.6 for a fixed value of X_2. Therefore, sensitivity measures with respect to x_1 would be similar to the ones in equation 3.24. The only remaining relationship to analyze is with respect to x_3. Equation 3.25 states that there is no relationship between the output and x_3, so a change in x_3 shall not produce any effect on the output. Hence, derivatives of the output with regard to x_3 are 0 for the entire input space, so the sensitivity measures for x_3 are:

$$S_{2,1}^{avg} \approx 0; \qquad S_{2,1}^{sd} \approx 0 \tag{3.28}$$

To summarize, based on the S^{avg} and S^{sd} for an input variable we can conclude:

- If both $S_{i,k}^{avg} \approx 0$ and $S_{i,k}^{sd} \approx 0$, it indicates that the output is not related to the input, because for all the input space the sensitivity of the output with regard to that input is approximately zero.

- If $S_{i,k}^{avg} \neq 0$ and $S_{i,k}^{sd} \approx 0$, it indicates that the output has a linear relationship with the input, because for all the input space the sensitivity of the output with regard to that input is approximately constant.
- If $S_{i,k}^{avg} \neq 0$, regardless of the value of $S_{i,k}^{avg}$, it indicates that the output has a non-linear relationship with the input, because the relation between the output and the input vary depending on the value of the input.

$S_{i,k}^{sq}$ retrieves a different kind of information from the relationship between input and output. For a given input variable, it may be considered significant if its sensitivities $s_{i,k}\big|_{\mathbf{x}_j}$ are significantly different from zero, whether they are positive or negative. In other words, a variable is considered to be significant when changes in the input variable produce significant changes in the output variable of the model. White and Racine, 2001 conclude that the statistic $\left(S_{i,k}^{sq}\right)^2$ is a valid indicator to identify if a variable is irrelevant following this criteria. Moreover, $S_{i,k}^{sq}$ is a measure of the changes in the output due to local changes in the input. Thus, $S_{i,k}^{sq}$ can be defined as a measure of the importance of the input variables from a perturbation analysis point of view, in the sense that small changes in that input will produce larger changes in the output.

3.3.2. Synthetic example

In order to show how sensitivity analysis is performed using the `NeuralSens` R library, we use a simulated dataset to train an MLP model of class `nn` (from the `RSNNS` library). The dataset consists of a `data.frame` with 2000 rows of observations and four columns for three input variables (X_1, X_2, X_3) and one output variable (Y). The input variables are random observations of a normal distribution with zero mean and standard deviation equal to 1.

The output Y is created following Equation 3.29 based on X_1 and X_2:

$$Y = (X_1)^2 - 0.5 \cdot X_2 + 0.1 \cdot \varepsilon \tag{3.29}$$

where ε is random noise generated using a normal distribution with zero mean and standard deviation equal to 1. X_3 is given to the model for training and a proper fitted model would find no relation between X_3 and Y.

In order to test the functionality of the sensitivity analysis designed, a MLP model with 10 neurons in its hidden layer is trained using the previously described dataset. The sensitivity analysis is performed on the model in the samples of the entire dataset. Upon completion, user might retrieve information about the sensitivity measures proposed in 3.3:

```
Sensitivity analysis of 3-10-1 MLP network.
Sensitivity measures of each output:
$Y
           mean        std meanSensSQ
X1 -0.005406908 1.94524276 1.94476390
X2 -0.485564931 0.06734504 0.49021056
X3 -0.003200699 0.02971083 0.02987535
```

The results display sensitivity analysis of a 3-10-1 MLP network, revealing mean and standard deviation values for each input variable (X_1, X_2, X_3). These values denote:

- X_1 with mean sensitivity ($S_{i,k}^{avg}$) approximately 0 and standard deviation ($S_{i,k}^{sd}$) roughly 2. This implies it has a non-linear effect on the output variable.
- X_2 with $S_{i,k}^{avg} \approx 0.5$ and $S_{i,k}^{sd} \approx 0$. This suggests it has a linear effect on the output variable.
- X_3 with both $S_{i,k}^{avg}$ and $S_{i,k}^{sd}$ near 0. This signifies it has no influence on the output variable.

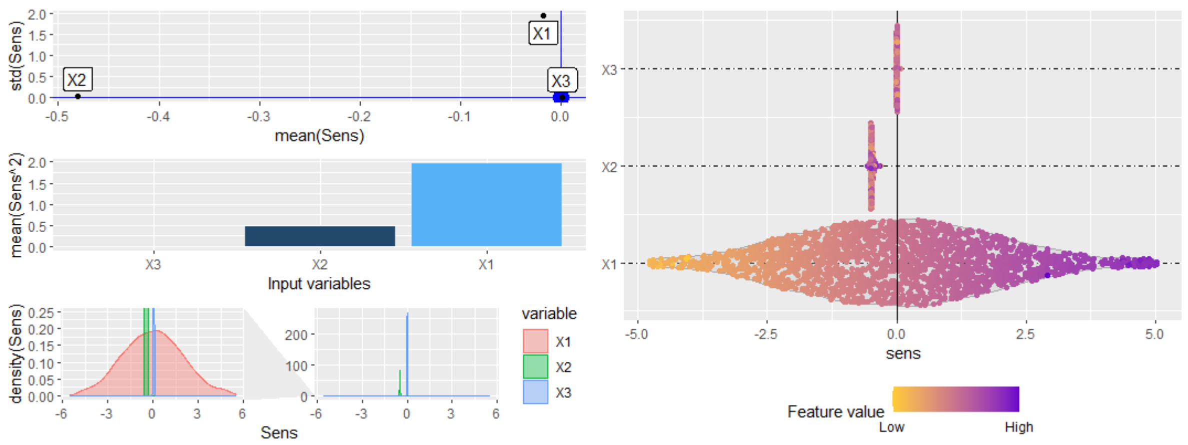
User might also be interested in the raw value of the partial derivatives. `NeuralSens` also provides methods to retrieve the partial derivatives of the output with respect to the input variables:

```
Sensitivity analysis of 3-10-1 MLP network.

2000 samples

Sensitivities of each output (only 2 first samples):
$Y
               X1         X2           X3
[1,]  2.08642384 -0.4462707 -0.044063158
[2,] -0.34976334 -0.3547381  0.014188363
```

While the ability to present sensitivities and sensitivity metrics in numeric format is valuable for extracting information from a model, it becomes increasingly challenging to interpret this data when the model have more than five input variables. To solve this, `NeuralSens` provide plotting functionalities to show the sensitivity metrics in a user-friendly manner:



(a) SensitivityPlots() **(b) SensFeaturePlot()**

Figure 3.8. Example from the **SensitivityPlots()** and **SensFeaturePlot()** function. From figure 3.8(a), first plot shows the relation between the mean and standard deviation of the sensitivities, the second plot shows the square of the sensitivities and the third and fourth plots show the distribution of the sensitivities. Figure 3.8(b) shows the value of the partial derivatives (in the x-axis) with respect to each input variables in each sample of the dataset, colored by the value of each input in that sample.

Figure 3.8(a) shows an example of the plots we propose:

1. Scatter and label plot representing the relationship between $S_{i,k}^{avg}$ (x-axis) and $S_{i,k}^{sd}$ (y-axis). Input features with non-linear relationships with the output would be placed far from the horizontal blue line which represents a $S_{i,k}^{sd}$ value of 0. Input features with linear or non-existent relationships with the outputs would be placed near the horizontal blue line,

with the former being placed far from the vertical blue line (which represents a $S_{i,k}^{avg}$ of 0) and the latter being placed near the vertical blue line.

2. Bar plot that shows $S_{i,k}^{sq}$ for each input variable, related to the feature importance of each input.
3. Density plot that shows the distribution of output sensitivities with regard to each input (Muñoz and Czernichow, 1998):
   - The narrow distribution of sensitivity values for X_2 (corresponding to a constant sensitivity) indicates a linear relationship between this input and the output of the neural net.
   - The wide distribution of sensitivity values for X_1 (corresponding to a variable sensitivity) indicates a non-linear relationship between this input and the output.

   When the height of at least one of the distributions is greater than 10 times the height of the smallest distribution, then an extra plot is created using the **facet_zoom()** function of the `ggforce` package (Pedersen, 2019). These plots provides a better representation of the sensitivity distributions.

In this case, the first plot of figure 3.8(a) shows that Y has a negative linear relationship with X_2 ($S_{i,k}^{sd} \approx 0$ and $S_{i,k}^{avg} < 0$), no relationship with X_3 ($S_{i,k}^{sd} \approx 0$ and $S_{i,k}^{avg} \approx 0$) and a non-linear relationship with X_1 ($S_{i,k}^{sd} \neq 0$). The second plot shows that X_3 barely affects the response variable, being X_1 and X_2 the inputs with most effect on the output.

Figure 3.8(b) displays the value of the first partial derivative of the output with respect to each input variable, with the color indicating the value of the input for every sample in the dataset. This figure presents similar information to the first plot of figure 3.8(a). It reveals a non-linear relationship with respect to `X1` (where the partial derivative range varies from -5 to 5 depending on the value of `X1`), a linear relationship with respect to `X2` (where the partial derivative remains constant with a non-zero value for every sample of `X2`), and no relationship with `X3` (where the partial derivative remains constant and equal to 0 for every sample of `X3`). However, this plot provides an opportunity to analyze how the partial derivatives are influenced by the value of the input feature. In this case, the plot of `X1` exhibits a quadratic-like relationship between the output and `X1`, where lower values of `X1` correspond to negative values of the partial derivative and higher values of `X1` correspond to positive values of the partial derivative.

Analogous objects and plots can be created using the `neuralsens` python package. A synthetic example in python can be found in the repository associated with the thesis in https://github.com/JaiPizGon/NeuralSens.

3.4. α-curves as a XAI method

Measures from sensitivity analysis are useful to retrieve information from a ML model. The main advantage of this method is that it provides feature importance measures together with information about the relationship between the output and the input, requiring less computational resources compared to other techniques.

However, the feature importance measures of $S_{i,k}^{sq}$ give few information about the sensitivity distribution along the input space. An input variable with low sensitivities in most of the input space but with high sensitivity in certain samples would be assigned a low average importance, misleading the user. The $\alpha-$curves method is an evolution of sensitivity analysis, providing a metric interpretation of the partial derivatives distribution. This provides extra information by not only giving the same feature importance measures, but also provides information about how the sensitivity with respect to a variable is distributed in the input space. In this section, a definition of $\alpha-$curves as XAI method is presented. This definition rises from a more complete explanation presented in Pizarroso *et al.*, 2023, where it is demonstrated how the application of metric theory on the sensitivities of a model derives to the $\alpha-$curves method presented in this section.

3.4.1. Definition of α-curves

A natural setup for a metric analysis on tabular data is to choose the involved metrics to be L^p norms. Recall that for each $p \in [1, \infty)$, we define the L^p norm as:

$$\|(x_1, \ldots, x_M)\|_p = \left(\sum_{i=1}^{M} |x_i|^p \right)^{1/p}$$

where x is the vector of values to which L^p norm is applied.

Taking the limit when $p \to \infty$, we also have:

$$\|(x_1, \ldots, x_M)\|_\infty = \max\{|x_i|\}$$

Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a ML regression model with n input variables and 1 scalar output variable, let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ be a dataset with $\mathbf{x}_i \in \mathbb{R}^n$ and let $s_{j,1}\big|_{\mathbf{x}_n}$ denote the derivative of the output of f with regard to variable X_j in the n^{th} sample of $\mathcal{X}$ dataset as defined in equation 3.4. This motivates the following definition presented in Pizarroso *et al.*, 2023. Let us define the *α-mean sensitivity of f with respect to variable x_j on the dataset $\mathcal{X}$* as

$$\mathrm{ms}^{\alpha}_{\mathcal{X},j}(f) := M_\alpha\{s_{j,1}\big|_{\mathbf{x}_1}, \ldots, s_{j,1}\big|_{\mathbf{x}_N}\} = N^{-1/\alpha} \cdot \|(s_{j,1}\big|_{\mathbf{x}_1}, \ldots, s_{j,1}\big|_{\mathbf{x}_N})\|_\alpha$$

Then, define the sensitivity α-curve as the map

$$\begin{array}{rcl} \mathrm{ms}_{\mathcal{X},j}(f) : [1,\infty] & \longrightarrow & [0,\infty) \\ \alpha & \mapsto & \mathrm{ms}^{\alpha}_{\mathcal{X},j}(f) \end{array} .$$

On the other hand, observe that the Generalized Mean Inequality implies that for each $0 \leq \alpha < \beta \leq \infty$ we have

$$M_\alpha \left\{ \left| \frac{\partial f}{\partial X_j}(\mathbf{x}_i) \right| \right\} \leq M_\beta \left\{ \left| \frac{\partial f}{\partial X_j}(\mathbf{x}_i) \right| \right\}$$

and we know that

$$M_\infty \left\{ \left| \frac{\partial f}{\partial X_j}(\mathbf{x}_i) \right| \right\} = \lim_{\alpha \to \infty} M_\alpha \left\{ \left| \frac{\partial f}{\partial X_j}(\mathbf{x}_i) \right| \right\}$$

so we conclude that $\mathrm{ms}_{\mathcal{X},j}(f)$ is an increasing bounded curve whose limit when $\alpha \to \infty$ is $\mathrm{ms}^{\infty}_{\mathcal{X},j}(f)$. A representation of this curve, together with the asymptotic value $\mathrm{ms}^{\infty}_{\mathcal{X},j}(f)$, yields an interesting visualization of the whole sensitivity analysis. We will call this representation the *sensitivity α-curve associated to f with respect to variable X_j over the dataset $\mathcal{X}$*.

These α-curves are designed to study scalar regression data models by comparing the values of the curves for different values of α. Thus, in order to analyze data with this method, the first step is to build a representative data model f trained over the data. The α-sensitivity analysis is always meant to study properties of the chosen model f and not necessarily on the data which generated it. In order to have proper comparisons between variables when analyzing the model, it is recommended that the variables of the dataset are normalized or have comparable magnitudes before constructing the model f and performing the α-sensitivity analysis.

To facilitate the comparison of the $\alpha-$curves values, Pizarroso *et al.*, 2023 proposed the α-curve plot. An α-curve plot is a 2-d plot in which we draw together, for each variable X_j, the variation of the α-sensitivity of the model, $\mathrm{ms}^{\alpha}_{\mathcal{X},j}(f)$, when α varies. By the Generalized Mean Inequality, we know that each α-curve is increasing and bounded. As the curve can sometimes increase very slowly as α increases, and the limit value $\alpha = \infty$ is interesting for our analysis, we will draw the curve up to a certain limit (Pizarroso *et al.*, 2023 found that $\alpha \in [1, 16]$ is enough in most experiments) and then add to the plot the asymptotic value $\mathrm{ms}^{\infty}_{\mathcal{X},j}(f)$ for each variable X_j. An example of this plot can be found in figure 3.10.

3.4.2. Comparison of variables for a fixed α

For each value of α, the set of values $\mathrm{ms}^{\alpha}_{\mathcal{X},j}(f)$ provides a measure of the sensitivity of the model f with regard to each variable X_j. Thus, it can be used to compare which input variables are more relevant for the output in the model on different scales.

It must be noted that, for $\alpha = 1$ and $\alpha = 2$, analogous metrics for S^{avg} and S^{sq} as defined in section 3.3 are obtained. In the literature, these metrics have been used as sensitivity metrics White and Racine, 2001; Muñoz and Czernichow, 1998; Pizarroso-Gonzalo *et al.*, 2022 and utilized, for instance, for variable pruning Yeh and Cheng, 2010; Zeng *et al.*, 2018.

Each vertical cut α to the α-curve plot can be used to compare the variables and draw quantitative and qualitative conclusions about their relative relevance to the model. Any value of α could, theoretically, be used for the sensitivity comparison task independently. Nevertheless, analyzing the whole picture across all different α allows a deeper understanding of the behavior of the model. Due to the properties of α-means, as α increases, the average value $\mathrm{ms}^{\alpha}_{\mathcal{X},j}(f)$ takes more into account the existence of regions in the dataset where the sensitivity with regard to the variable is higher than average ("exceptionally sensitive" or "localized high sensitivity" behavior). Lower values of α focus instead on the "average behavior" of the function with regard to the variable.

The analysis of high values of α may be crucial for certain tasks like variable pruning. It is possible that a variable has almost no impact on a regression problem if one looks at a generic point in the phase space, but that there exists a mode change in the model making the variable very relevant for the analysis when the inputs move inside a certain critical region (think, for instance, in the case where there exist "activation" variables or states, which enable a different variable to influence the result but otherwise disable it). A general pruning analysis with $\alpha = 1$ or $\alpha = 2$ could "discard" the variable as irrelevant for the model, whereas it might be the most relevant variable for high α metrics.

The limit values $\mathrm{ms}^{\infty}_{\mathcal{X},j}(f)$ included in the plot help identify the extreme cases. They measure the maximum sensitivity of f with regard to the input variable X_j that can be found at any point in the dataset.

3.4.3. Analysis of the variation of an α-curve

Due to the aforementioned properties of the α-means, studying the variation of the α-sensitivity when α changes can give valuable information on the dynamics of the variables of the model f. Let us study some examples.

3.4.3.1. Linearity analysis

By the Generalized Mean Inequality, the α-curve of variable X_j is constant if and only if f is of the form

$$f(X_1, \dots, X_n) = g(X_1, \dots, X_{j-1}, X_{j+1}, \dots, X_n) + CX_j$$

for some function g depending only on the rest of the variables.

By extension, the closer an α-curve is to be flat, the closer the dependence of f with respect to X_j is to a linear dependence. For instance, when an α-curve starts almost flat and then begins to increase at some alpha, this can mean that the derivative $\frac{\partial f}{\partial X_j}$ has a low variation through most of the input space of the dataset, but that there are one or more regions of the input space where it varies notably, either due to its own non-linear behavior (like an activation function, or function where the derivative increases close to a point, like a $\mathcal{C}^2$-approximation of a square root), or due to an interaction with other variables.

3.4.3.2. Irrelevant variables

As a particular case of the previous analysis, f does not depend on a variable X_j if and only if the α-curve is constantly zero. The closer a curve is to 0, the less important the variable is for the model.

If a curve starts flat and close to 0 but increases afterwards, this indicates that the output of the model has, in general, a low dependence on the variable, but that there exists a region in the phase space in which the variable is indeed relevant for the model.

These properties can be used to improve the specificity of variable pruning methodologies. If a variable presents a low value of $\mathrm{ms}^{\infty}_{\mathcal{X},j}(f)$ (and, thus, the whole α-curve is low) then it is not important for the model and it can be safely removed. On the contrary, a variable presenting higher values of the curve for some α (and thus, a higher $\mathrm{ms}^{\infty}_{\mathcal{X},j}(f)$) should not be pruned without a deeper analysis.

3.4.3.3. Detection of local regions with high sensitivity

As outlined before, it is possible for a variable to have low sensitivity for low α but high sensitivity in higher α. This makes comparing its α-sensitivity with the α-sensitivity of other variables depend heavily on α. When this happens and a variable is not sensitive for low α but it becomes highly sensitive for high α, two things can happen.

- The variable shows a non-linear behavior on X_j which makes the partial derivative $\frac{\partial f}{\partial X_j}$ increase only on certain values of X_j.
- There exists an interaction between the variable and a combination of other variables which makes the derivative become high in a certain region of the phase space.

The higher the variation of the α-curve and the earlier this variation appears, the stronger and more generalized the interaction or non-linear effect is across the dataset. If the α-curve starts flat and then there is a sudden increase, it is more probable that the interaction or non-linear input effect on the output is relevant only in certain bounded areas of the dataset.

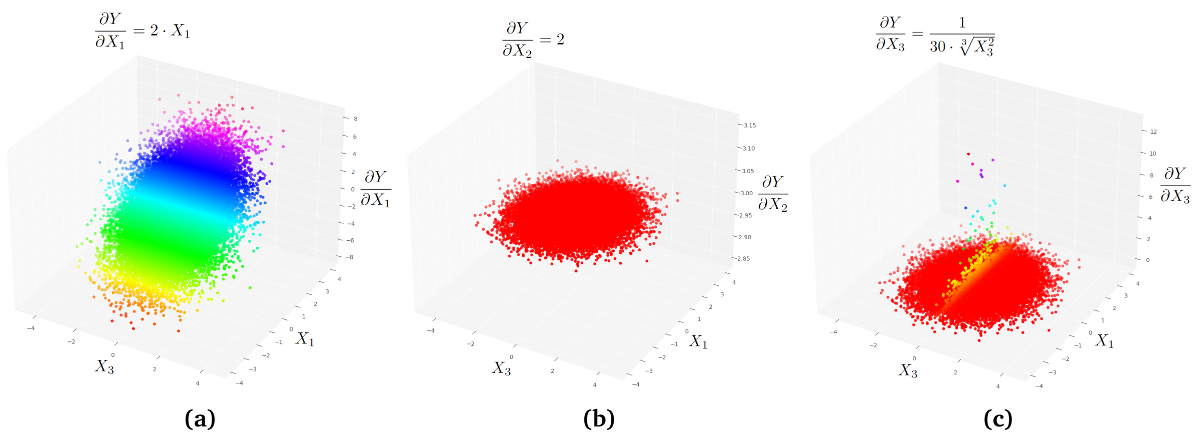


Figure 3.9. 3-D plots of partial derivatives of output Y with respect to inputs X_1, X_2 and X_3 ((a), (b) and (c) respectively) for square root synthetic dataset. X-axis follows X_1 and y-axis follows X_3 in the three plots. X_2 is not used as plot axis due to the irrelevance of this variable on the derivative plots.

3.4.4. Synthetic example

A synthetic dataset with known derivatives is used to illustrate the usefulness of the α-curves to retrieve information about how the model uses the input variables to predict the output variable. The dataset is composed by 8 input variables $[X_1, \ldots, X_8]$ and one output variable $Y \in \mathbb{R}$ created as a function of the input variables, i.e., $Y = f(\mathbf{X})$. Input variables $\mathbf{X}$ consist in 50000 samples drawn from a normal distribution with $\mu = 0$ and $\sigma = 1$.

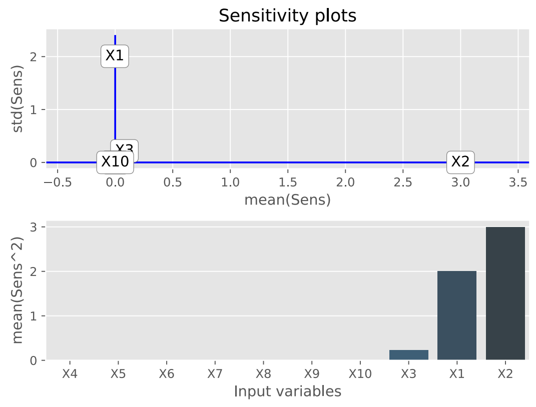
In this case, the output follows the next expression:

$$Y = (X_1)^2 + 2 \cdot X_2 + \frac{1}{10} \cdot \sqrt[3]{X_3} \tag{3.30}$$

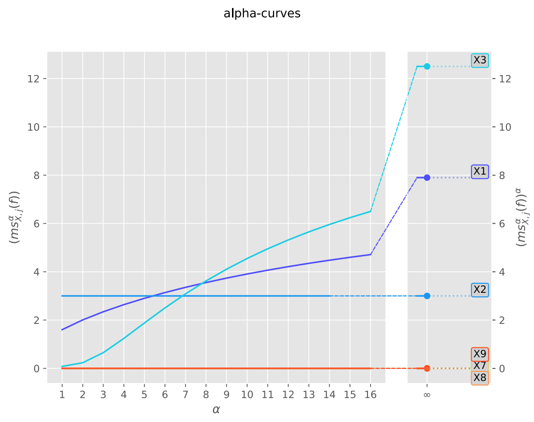
From figure 3.9, we can conclude that X_2 has a linear relationship with Y as $\frac{\partial Y}{\partial X_2}$ is constant and different from zero for all samples. X_1 and X_3 have a non-linear relationship with Y, as $\frac{\partial Y}{\partial X_1}$ and $\frac{\partial Y}{\partial X_3}$ are not constant for all samples. Furthermore, figure 3.9(c) shows that $\frac{\partial Y}{\partial X_3} = 0$ for most samples, except for the samples where X_3 is close to 0. In these samples, sensitivities with respect of X_3 are far higher than with respect to the other input variables, so changes of X_3 in this region of the input space shall provoke large changes on Y. This can be understood as a local importance of X_3, and it shall be detected by XAI methods.

Results of XAI analysis performed on equation 3.30 are presented in figure 3.10. Figure 3.10(a) shows the sensitivity plots as introduced in Pizarroso-Gonzalo *et al.*, 2022. First plot shows two sensitivity metrics: mean (x-axis) and standard deviation (y-axis). Second plot of figure 3.10(a) shows the mean squared sensitivity for each of the input variables, which could be used as a variable importance metric. According to these metrics, the following information can be retrieved from figure 3.10(a):

- X_2 variable has a linear relationship with the output.
- X_1 variable has a non-linear relationship with the output.
- X_3 is almost irrelevant to predict the output, with much lower importance than X_1 and X_2, but greater than $X_4 - X_8$.
- The remaining variables have no relationship with the output.



(a) Sensitivity plots of cubic root synthetic dataset.



(b) α-curves of cubic root synthetic dataset.

Figure 3.10. XAI techniques plots of cubic root synthetic dataset.

Using the $\alpha-$curves methodology described in section 3.4.3, previously obtained information can be obtained from figure 3.10(b). However, it also shows that, apart from the non-linearities presented in X_1 and X_3, there are regions where output Y is far more sensitive to X_3 than to X_2. In fact, peak sensitivities in some samples are detected, as can be seen by the rapid increase of $\mathrm{ms}^{\alpha}_{\mathcal{X},3}(f)$ for $\alpha > 4$. This information could not be retrieved using the sensitivity analysis method, which assigned little importance to X_3, due to the aggregation techniques used to calculate importance of the input variables. Therefore, although sensitivity analysis suggests to remove X_3 from the model, it should be kept as input variable to model the relationship between output and input correctly.

3.5. Interaction Detection

A limitation of the $\alpha-$curves method presented in the previous section is that it is difficult to distinguish between the increase in sensitivity produced by an interaction between variables and a non-linear input effect (which can be thought of as a self-interaction of the variable). A viable solution to this problem is to explicitly detect the interactions between variables with a complementary method.

In the context of machine learning, interactions between variables play a crucial role in understanding the behavior of complex models. An interaction occurs when the effect of one variable on the output depends on the value of another variable.

Mathematically, we can say that a function $f(\mathbf{x})$ has an interaction between variables x_1 and x_2 if it can not be decomposed as:

$$f(\mathbf{x}) = f_1(x_1, \mathbf{x}_{\setminus(1,2)}) + f_2(x_2, \mathbf{x}_{\setminus(1,2)}) \tag{3.31}$$

where f_1 and f_2 are functions of x_1 and x_2 respectively. The term $\mathbf{x}_{\setminus(1,2)}$ represents all variables in $\mathbf{x}$ excluding x_1 and x_2. If there exist a non-negligible interaction between x_1 and x_2, the decomposition of f might be performed following:

$$f(\mathbf{x}) = f_1(x_1, \mathbf{x}_{\setminus(1,2)}) + f_2(x_2, \mathbf{x}_{\setminus(1,2)}) + f_{12}(\mathbf{x}) \tag{3.32}$$

where f_{12} is a function that captures the interaction between x_1 and x_2.

Detecting and understanding these interactions is important as they can significantly influence the output of a machine learning model. Ignoring these interactions can lead to oversimplified models that fail to accurately capture the underlying data patterns.

As an initial approach, one could adopt the approach outlined in Gevrey *et al.* (2006) and use second partial derivatives to detect interactions between a pair of variables. To detect interactions between variables x_i and x_j in a machine learning model, we can compute the second-order partial derivative $\frac{\partial^2 f}{\partial x_i \partial x_j}$, where $f(x_1, x_2, \ldots, x_n)$ represents the output of the model with input variables $x_1, x_2, \ldots, x_n$. Then, using the mean of these second partial derivatives

$$S_{ij,k}^{avg} = \frac{\sum_{n=1}^{N} s_{ij,k}\big|_{\mathbf{x}_n}}{N} \tag{3.33}$$

we could obtain information about pairwise input interactions in the model as:

1. If $S_{ij,k}^{avg}$ is positive (i.e., $\frac{\partial^2 f}{\partial x_i \partial x_j} > 0$), this indicates that the variables have a synergistic interaction. In other words, when the values of both x_i and x_j increase, their combined effect on the output is greater than the sum of their individual effects.

2. If $S_{ij,k}^{avg}$ is negative (i.e., $\frac{\partial^2 f}{\partial x_i \partial x_j} < 0$), this indicates that the variables have an antagonistic interaction. That is, when the values of both x_i and x_j increase, their combined effect on the output is less than the sum of their individual effects.

3. If $S_{ij,k}^{avg}$ is close to zero (i.e., $\frac{\partial^2 f}{\partial x_i \partial x_j} \approx 0$), this suggests that there is little or no interaction between the two variables, and their effects on the output are independent of each other.

By calculating these second-order partial derivatives for all possible pairs of input variables, we can identify the presence and type of interactions between them. This approach can be complemented with the one adopted in Zhang *et al.* (2022), where the mean of squared second partial derivatives:

$$S_{ij,k}^{sq} = \sqrt{\frac{\sum_{n=1}^{N} \left(s_{ij,k}\big|_{\mathbf{x}_n}\right)^2}{N}} \tag{3.34}$$

is used as an interaction importance measure. Using these two measures we obtain an interaction analysis methodology similar to the sensitivity analysis presented in section 3.3.

3.5.1. Cancelling Interactions through Transformations

While interactions between variables can be identified through second partial derivatives, the detection of interactions following this method presents a notable problem due to the nature of data-related tasks. When detecting interactions between variables in a machine learning model, it is important to consider the transformation that input or output space might suffer due to the modelling process performed by the user.

Let us consider a model with output $y = f(x_1, x_2)$ and input variables x_1, x_2. This model can be represented as:

$$y = f(x_1, x_2) = \alpha(x_1) + \beta(x_2) + f_I(x_1, x_2) \tag{3.35}$$

with α, β and f_I being functions modelling the effect of x_1, x_2 and the interaction between x_1 and x_2 respectively. This nomenclature is the established and accepted in existing literature (Friedman and Popescu, 2008; Caruana *et al.*, 2015) to determine if a model presents an interaction between input variables. Two situations are found considering this nomenclature:

- No interaction: model f can be expressed without the term f_I and only the individual effects of x_1 and x_2 are needed to model the output. f is explained as a superposition between the effects of x_1 and x_2.
- Interaction: model f can not be expressed without the term f_I, where each input variable changes how f responds to variations in the other.

Let us consider a model which follows the next expression:

$$y = f(x_1, x_2) = x_1 \cdot x_2 \tag{3.36}$$

Following the previous classification, this model present an interaction between the two input variables x_1 and x_2 as the model f can not be expressed as a superposition of x_1 and x_2 individually.

Nevertheless, if we apply a change of variables:

$$\tilde{x}_1 = \log(x_1) \quad \tilde{x}_2 = \log(x_2) \quad \tilde{y} = \log(y)$$

Then we can model the transformed output $\tilde{y}$ using the transformed inputs $\tilde{x}_1$ and $\tilde{x}_2$ as:

$$\tilde{y} = \log(y) = \log(x_1 \cdot x_2) = \log(x_1) + \log(x_2) = \tilde{x}_1 + \tilde{x}_2$$

By taking the logarithm of both the input and output spaces, we can transform the multiplicative interaction into an additive superposition of the individual effects of the input variables. Now, the effects of x_i and x_j are additive in the transformed space and interaction has vanished.

This effect is specially relevant considering that is a common practice to apply transformations to the input variables (preprocess) previous to the training of the ML model. This transformation, as the previous seen logarithm, can cause interactions to appear or disappear without being detected by second partial derivatives, providing the user with misleading information. Consequently, a more interesting information that might be retrieved from the model would be the interaction effects between variables that do not vanish when a change of variables in the input and/or output space is applied.

At this point, for a model $y = f(x_1, x_2)$, three types of pairwise input interactions can be defined:

1. No interaction: model f can be understood as a superposition of the effects of x_1 and x_2 following the expression: $f(x_1, x_2) = \alpha(x_1) + \beta(x_2)$ for some α and β. Example: $y = f(x_1, x_2) = x^2 + \cos(y^3)$.
2. Explicit interactions: interaction of variables can be detached if there exist a change of variable $\gamma : \mathbb{R} \mapsto \mathbb{R}$ such that f can be expressed as: $f(x_1, x_2) = \gamma(\alpha(x_1) + \beta(x_2))$.

Model f can be understood as a superposition between the effects of x_1 and x_2 after an appropriate change of variables. Example: $y = f(x_1, x_2) = x_1 \cdot x_2$ with $\gamma = \exp()$ and $\alpha = \beta = \log()$.

3. Implicit interactions: interaction of variables can not be detached by any change of variables $\gamma : \mathbb{R} \mapsto \mathbb{R}$ and f follows the expression: $f(x_1, x_2) = \alpha(x_1) + \beta(x_2) + f_I(x_1, x_2)$.

As explicit interactions depends on the characteristics of the input and output space where the model is evaluated, and existing literature already detects this type of interactions (see Gevrey *et al.*, 2006); the information that is interesting to retrieve should be about existing implicit interactions in the model.

3.5.2. Implicit Interaction Detection using Interaction Invariant

The interaction invariant described in this section is specifically designed to detect implicit interactions between a pair of variables by detecting the variation of the curvature of the isovalue curves of a model´s output space. Alfaya *et al.* (2023) demonstrates mathematically that this change in the curvature can only be produced by an implicit interaction between input variables, and, therefore, this invariant is impervious to transformations of the input or output space, i.e., non-linear changes of the input and/or output variables. A deep explanation of how this invariant measures the variation of the curvature is out of the scope of this document. We refer the reader to Alfaya *et al.* (2023), where the mathematical derivation of the invariant expression is documented.

Let $f(x_1, x_2)$ be a regression model such that $\frac{\partial f}{\partial x_1}$ and $\frac{\partial f}{\partial x_2}$ are nowhere zero for the input space analyzed, and let be $I_L(f)$ be the local interaction function of $f(x_1, x_2)$. $I_L(f)$ defined as:

$$I_L(f) = \frac{f_{x_1x_1x_2} f_{x_1} f_{x_2}^2 - f_{x_1x_2} f_{x_1x_1} f_{x_2}^2 - f_{x_1x_2x_2} f_{x_1}^2 f_{x_2} + f_{x_1x_2} f_{x_2x_2} f_{x_1}^2}{f_{x_1}^2 f_{x_2}^2} \tag{3.37}$$

where $f_x = \frac{\partial f}{\partial x}$, $f_{x_1x_2} = \frac{\partial^2 f}{\partial x_1 \partial x_2}$ and $f_{x_1x_2x_3} = \frac{\partial^3 f}{\partial x_1 \partial x_2 \partial x_3}$. This local interaction function gives an interaction measure for an specific point of the input space which is invariant to single-variable linear transformations, i.e., to transformations of the form $\alpha(x) = a \cdot x + b$. A more general "total amount of interaction" is obtained through the integration of $|I_L|$ in the input region of interest.

Let $\Omega = [a, b] \times [c, d]$ be any measurable set and let $f : \Omega \longrightarrow \mathbb{R}$ be a regression model such that f_x and f_y vanish nowhere on Ω. Then define the measure $I(f)$ as follows. For each measurable $\Omega_0 \subseteq \Omega$, take

$$I(f)(\Omega_0) := \int_{\Omega_0} |I_L(f)| dx_1 dx_2 \tag{3.38}$$

We call $I(f)$ the *invariant interaction measure of f in* Ω_0 and we call $\mathcal{I}(f) := I(f)(\Omega)$ the *total interaction invariant of f*. These measures have two important properties:

- $I(f)$ and $\mathcal{I}(f)$ are preserved by any nonlinear change of variables
- $\mathcal{I}(f) = 0$ if and only if $f(x_1, x_2) = \gamma(\alpha(x_1) + \beta(x_2))$

The previous local and global interaction measures are built on the assumption that the function f has nowhere vanishing partial derivatives. The local interaction function $I_L(f)$ can clearly become singular at points where either f_{x_1} or f_{x_2} vanish if the numerator does not vanish simultaneously. The invariance computations are not affected, so the invariant interaction measure $I(f)$ and the total interaction invariant $\mathcal{I}(f)$ are all still invariant with respect to the action of a linear transformation, but the interaction measure $I(f)$ may not be a finite measure anymore in this case and $\mathcal{I}(f)$ might become infinite.

As vanishing partial derivatives for ML models are common, it is of utmost importance to provide a "regularized" version $\tilde{I}(f)$ of the invariant measure $I(f)$ which approximates $I(f)$ away from the zeros of the partial derivatives f_{x_1} and f_{x_2} and is still a finite measure even for functions whose partial derivatives vanish occasionally in Ω. This regularized invariant must meet the following conditions:

- $\tilde{I}(f)$ is a finite measure for all functions f, independently on the existence of zeroes of f_x or f_y in Ω.
- $\tilde{I}(f)(\Omega_0) \approx I(f)(\Omega_0)$ when Ω_0 does not contain zeroes of f_{x_1} or f_{x_2}. More precisely, the relative error between $\tilde{I}(f)$ and $I(f)$ should be bounded.
- $\tilde{I}(g \cdot f)(g \cdot \Omega_0) = \tilde{I}(f)(\Omega_0)$ for any linear transformation g.
- $\tilde{I}(f) = 0$ if and only if $I(f) = 0$.

As the singularity of $I(f)$ comes from the existence of zeroes of the denominator $f_x^2 f_y^2$ of the expression of $I_L(f)$, a possible way to regularize I_L would be to add an extra always positive term $r(f)$ to the denominator, yielding

$$\tilde{I}_L(f) = \frac{f_{x_1x_1x_2} f_{x_1} f_{x_2}^2 - f_{x_1x_2} f_{x_1x_1} f_{x_2}^2 - f_{x_1x_2x_2} f_{x_1}^2 f_{x_2} + f_{x_1x_2} f_{x_2x_2} f_{x_1}^2}{f_{x_1}^2 f_{x_2}^2 + r(f)} \tag{3.39}$$

$$\tilde{I}(f)(\Omega_0) = \int_{\Omega_0} \tilde{I}_L(f) dx_1 dx_2 \tag{3.40}$$

In Alfaya *et al.* (2023), $r(f)$ is defined as:

$$r(f) = w \cdot e^{-\frac{f_{x_1}^2 f_{x_2}^2}{w}} \tag{3.41}$$

with

$$w = \frac{w_0}{|\Omega|} \int_{\Omega} f_{x_1}^2 f_{x_2}^2 dx_1 dx_2 \tag{3.42}$$

where w_0 is an strictly positive regularization parameter. If w_0 is set to 0, the regularization factor $r(f)$ falls to 0 and the unregularizated formula for the invariant arise. It must be noted that the greater w_0 is, the greater the difference between $\tilde{I}(f)$ and $I(f)$. For most applications, Alfaya *et al.*, 2023 suggests that $w_0 \approx 0.01$ is sufficient to avoid vanishing derivatives when calculating the interaction invariant, providing an analysis of the effect of w_0 on the information retrieved from the model.

3.5.3. Numerical Implementation of Interaction Invariant

The numerical implementation of the interaction invariant is a significant advancement in this thesis, primarily due to its practical applicability in the analysis of neural network models. An initial approach could be to symbolically calculate the interaction invariant from the model's expression, which is often unfeasible for MLP models with more than 2 neurons in the hidden layer due to the computational requirements of symbolically calculating the partial derivatives.

This limitation necessitates an alternative method to efficiently compute the interaction invariant for complex MLP models.

The proposed solution leverages tensorial calculations to compute the Jacobian, Hessian, and Jerkian arrays of the MLP model presented in previous sections. To analyze the interaction between inputs x_1 and x_2 present in a MLP model f, the numerical calculation of the interaction invariant is performed as follows:

1. Create a customizable rectangular grid along the axis determined by x_1 and x_2. The cells of this grid are delimited by the minimum and maximum value of x_1 and x_2, creating equidistant points in the axis determined by the number of points in each axis N_1 and N_2.

2. Calculate the Jacobian (J_0^L), Hessian (H_0^L) and Jerkian (K_0^L) of the output of f with respect to x_1 and x_2 in each vertex of each rectangle in the grid. These arrays have $[N_1 \times N_2]$ rows, corresponding to the number of vertex in the grid.

3. If $\omega_0 \neq 0$, calculate the regularization of the interaction invariant $r(f)$. As the partial derivatives are calculated in a discrete set of points in the input space, equation 3.42 is used performing trapezoidal integration on the values of $f_{x_1}^2 f_{x_2}^2$ in all vertex calculated in the previous step, with $|\Omega|$ being the total area of the grid. If $\omega_0 = 0$, $r(f) = 0$ for all the cells in the grid.

4. The local interaction function $\tilde{I}_L(f)$ is calculated in each vertex following equation 3.39. This operation is performed using standard tensorial operations by selecting the corresponding vectors of the Jacobian, Hessian and Jerkian arrays. For example, f_{x_1,x_1,x_2} is the column vector stored in Jerkian K_0^L in the index $(1...N_1 \times N_2, 1, 2, 2)$.

5. For each cell of the grid, calculate the interaction invariant $\tilde{I}(f)(\Omega_0)$ as defined in equation 3.40 using trapezoidal integration, with Ω_0 being the analyzed cell. This result in a $[N_1 \times N_2]$ matrix with the value of $\tilde{I}(f)(\Omega_0)$ for each cell.

A depiction of this process is illustrated in figure 3.11.

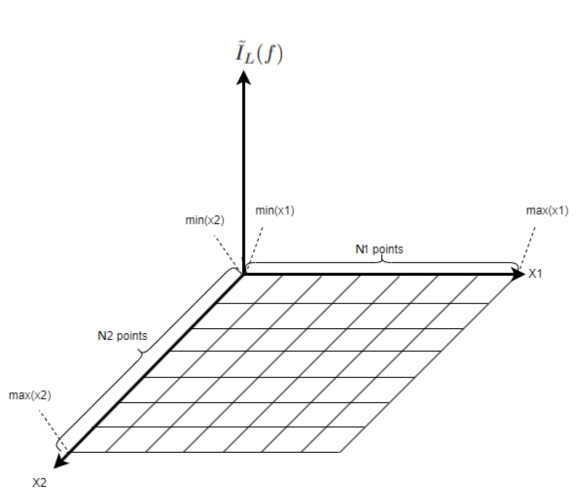
The fact that the invariant values $\tilde{I}(f)$ are calculated for each cell allows for several interpretations of the interaction between the two input variables:

1. via `sum`: summing the values of the invariant for all the cells considered gives a measure of the total interaction between the two input variables in the analyzed input space.

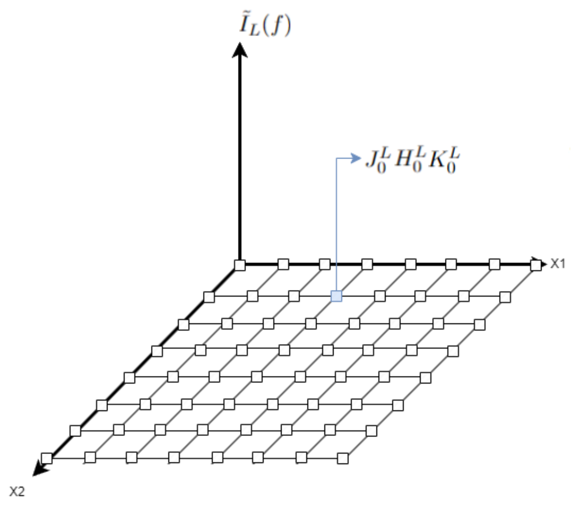
2. via `heatmap plot`: the heatmap represents the value of the invariant along the cells of the user-defined grid. Color of the heatmap depends on the invariant values, allowing to detect regions with different level of interaction.

3. via `3-D scatter plot`: similar to the heatmap, the cells analyzed are represented by points in a 3-D space, where the `x` and `y` axis represents the user-defined grid and the `z` axis represents the prediction of the model. Color of the points is given by the value of the invariant in each cell. This representation allows not only to detect regions with different level of interaction, but also the behavior of the model predictions in each region.

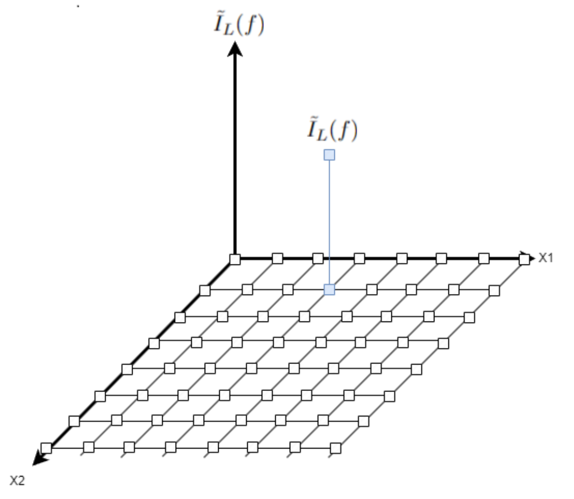
An example of the three interpretations is shown in the following section.



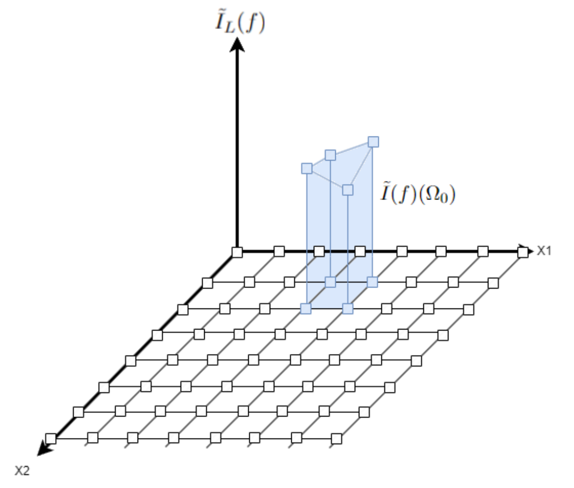
(a) Create grid of x_1 and x_2 with user-defined parameters N_1 and N_2



(b) Calculate the Jacobian (J_0^L), Hessian (H_0^L) and Jerkian (K_0^L) of f in each vertex of the grid



(c) Calculate the local interaction function $\tilde{I}_L(f)$ in each vertex of the grid



(d) Calculate the interaction invariant $\tilde{I}(f)$ in each cell of the grid

Figure 3.11. Representation of the numerical approach to calculate the interaction invariant $\tilde{I}(f)$ between input variables x_1 and x_2 for a MLP model f.

3.5.4. Synthetic example

Two synthetic datasets with known derivatives are used to illustrate the usefulness of the interaction invariant to detect implicit interactions between variables. The datasets are composed by 2 input variables $[X_1, X_2]$ and one output variable $Y \in \mathbb{R}$ created as a function of the input variables, i.e., $Y = f(\mathbf{X})$. Input variables $\mathbf{X}$ are 50000 samples drawn from a uniform distribution $U(-1, 1)$.

In these cases, the output follows the next expressions:

$$(1) Y = (X_1 + 2) \cdot (X_2 + 2) + 0.1 \cdot \varepsilon \tag{3.43}$$

$$(2) Y = (X_1 + 2) \cdot (X_2 + 2) + (X_1)^2 + 0.1 \cdot \varepsilon \tag{3.44}$$

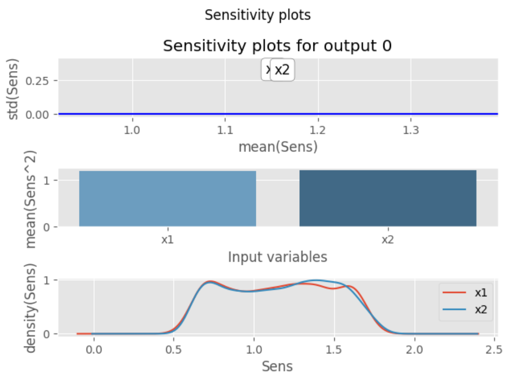
Although both expressions are similar and seem to have an interaction between X_1 and X_2, the interaction present in 3.43 is an explicit interaction which can be detached by applying a logarithm transformation:

$$\log(Y) = \log((X_1 + 2) \cdot (X_2 + 2)) = \log(X_1 + 2) + \log(X_2 + 2) \tag{3.45}$$

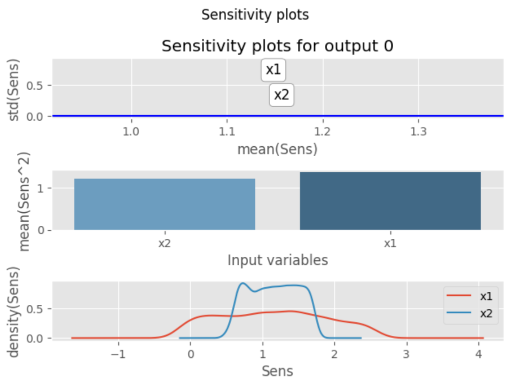
while the interaction present in 3.44 can not be detached by any transformation as far as the authors are concerned.

A MLP model is trained for each dataset. All methods described in this chapter (Sensitivity analysis, $\alpha-$curves and interaction invariant) are applied to analyze the two models. The results of each analysis are gathered below:

- Sensitivity analysis: Using sensitivity analysis with the first partial derivatives (figure 3.12), for dataset (1) both variables are equally important with similar effect on the output. For dataset (2), X_1 have a greater non-linear relationship with the output, as expected due to the $(X_1)^2$ term. Obtaining the sensitivity metrics for the second partial derivatives (figure 3.13), interaction between X_1 and X_2 is detected in both datasets.

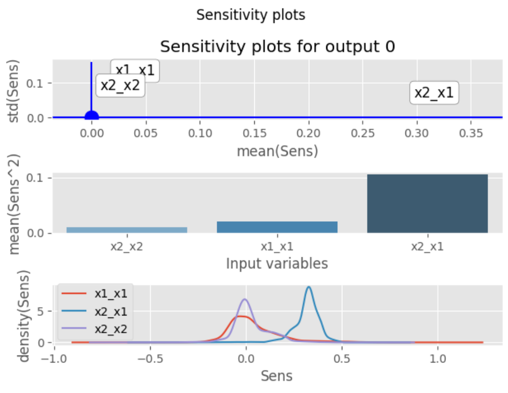


(a) Sensitivity Analysis (1)

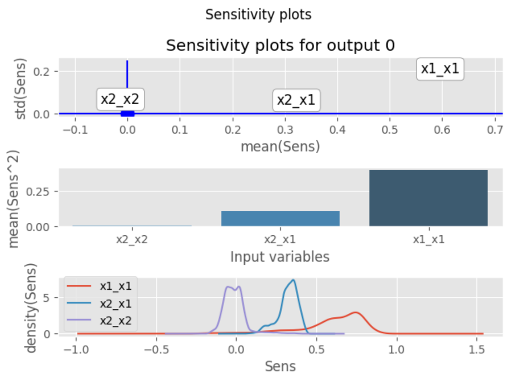


(b) Sensitivity Analysis (2)

Figure 3.12. Sensitivity analysis using the first partial derivatives for both synthetic datasets.



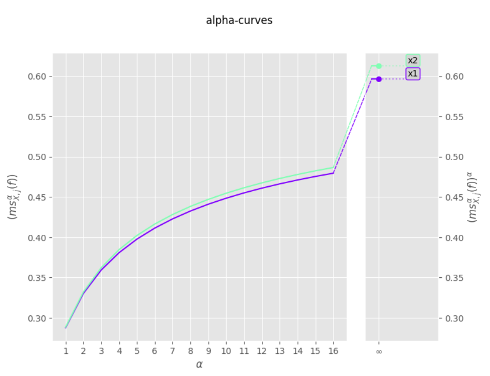
(a) Sensitivity Analysis (1)



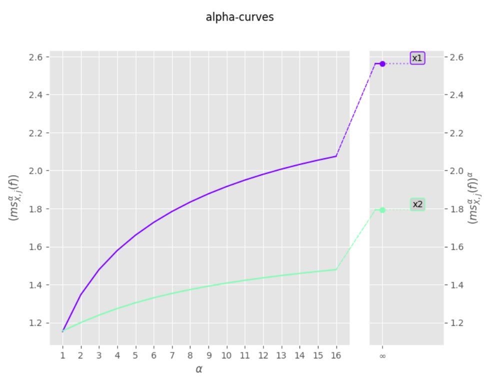
(b) Sensitivity Analysis (2)

Figure 3.13. Sensitivity analysis using the second partial derivatives for both synthetic datasets.

- $\alpha-$curves: Following the $\alpha-$curves methodology, both X_1 and X_2 variables have almost identical sensitivity distributions for dataset (1). For dataset (2), X_1 have a greater non-



(a) $\alpha-$curves analysis (1)



(b) $\alpha-$curves analysis (2)

Figure 3.14. α-curves analysis for both synthetic datasets.

linear relationship with the output variable, same information as the extracted using sensitivity analysis.

- Interaction Invariant: Figure 3.15 shows the interaction invariant plot with regularization parameter $w_0 = 0.01$ for both datasets with a grid composed by 100 intervals for each input variable. For dataset (1), no interaction is detected in any cell of the analyzed grid, which suggests that a change of variables can be applied to y, x_1 or x_2 such that the output can be expressed as a superposition of the effects of x_1 and x_2 (see equation 3.45).

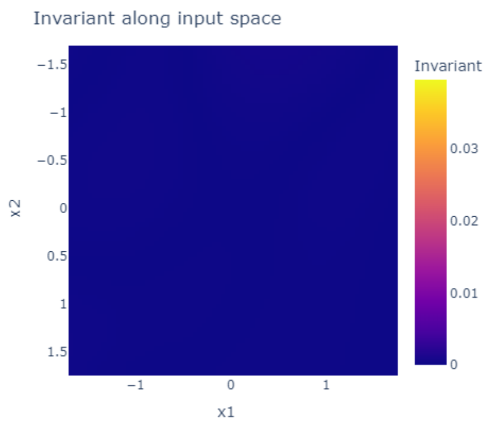
  For dataset (2), the points with a highest value of interaction invariant are on the diagonal from $(-1.5, -0.5)$ to $(-1, -1.5)$. This suggest an implicit interaction from x_1 and x_2. Outside of this region, the interaction of x_1 and x_2 can be detached using a change of variables.

  Comparing the numeric sum of the invariant, notable differences emerge. For dataset (1), the sum of the interaction detected in all the cells of the grid yielded a value of 1,03. Dataset (2) exhibits a significantly higher interaction measure of 16,02. This substantial difference indicates a much stronger level of interaction among the variables in dataset (2) compared to dataset (1). These numeric results provide quantitative evidence supporting the analysis of figure 3.15 made above, emphasizing the distinct levels of variable interaction present in the two datasets.

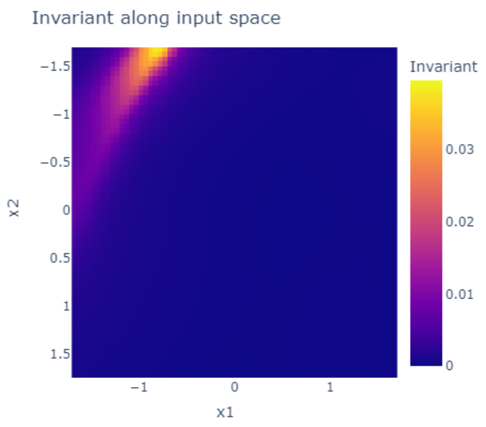
3.5.5. Extending Interaction Invariant to Higher Dimensional Datasets

A key limitation of the interaction invariant, as previously discussed, is its necessity for a model with only two input variables. This requirement is a significant drawback for data-related problems, given that real-world datasets often encompass more than two variables. To address this issue, Alfaya *et al.* (2023) propose three methods to extend the interaction invariant to higher dimensional datasets: the full-mesh, the sparse, and the full-sparse method.

As explained in previous sections, the main idea of the interaction invariant is to analyze the interaction between two variables in a grid defined by the user for the two input variables whose interaction we are interested in. The full-mesh method is the natural extension of this idea but, instead of creating a 2-d grid with size $[N_1 \times N_2]$, a n-d grid with size $[N_1 \times N_2 ... \times N_n]$ shall be created, where n is the number of input variables of the model. Then, the Jacobian, Hessian and Jerkian arrays and the local interaction function $\tilde{I}_L(f)$ are calculated for each vertex of the



(a) Interaction Invariant analysis (1)



(b) Interaction Invariant analysis (2)

Figure 3.15. Interaction Invariant analysis for both synthetic datasets.

grid as described in section 3.5.2. At this point, any pairwise interaction can be analyzed with a suitable projection of the local interaction function $\tilde{I}_L(f)$ on the ij plane defined by the x_i, x_j variable; obtaining the value of $\tilde{I}_L(f)$ in the ij grid. With this projection, the $\tilde{I}_L(f)$ can then be integrated in the ij grid to calculate the interaction invariant $\tilde{I}_L(f)$. The projection proposed in Alfaya *et al.* (2023) is to take the maximum value of $\tilde{I}_L(f)$ for each cell of the ij grid.

The full-mesh method has the distinct advantage of being able to analyze the pairwise interaction between all input variables using the same N-d grid by projecting the $\tilde{I}_L(f)$ onto every plane determined by each pair of input variables. However, this method has a disadvantage that may render it computationally unfeasible for high dimensional datasets: the number of grid points for which partial derivatives must be calculated increases exponentially with the number of input variables analyzed. To mitigate this, the sparse method proposes a lighter approximation of the interaction invariant. Instead of creating an n-d grid, a 2-d grid of size $[N_1 \times N_2]$ is created for the variables x_i and x_j, as in the original method for 2 variables. Then, to account for the effect of the other input variables,a total of N_3 points are sampled from the remaining variables, creating a $[N_1 \times N_2 \times N_3]$ grid. This grid is then used as in the full-mesh method, yielding an approximation of the interaction invariant $\tilde{I}(f)$ for the variables of interest x_i and x_j.

The full-mesh and sparse methods offer interaction measures for models with more than 2 input variables, where the computation of the model's partial derivatives is feasible for a large number of points. However, if the calculation of the partial derivatives is computationally intensive, the full-sparse method provides an even more efficient alternative than the sparse method. In this method, only a 2-d grid is required for the input variables x_i and x_j, along with a dataset where the model's partial derivatives with respect to the input variables can be calculated. After obtaining the Jacobian, Hessian, Jerkian, and $\tilde{I}_L(f)$ of the model at the dataset points, each dataset point is assigned to the Ω_0 cell of the ij grid containing that point. With all the points assigned, the maximum of the calculated $\tilde{I}_L(f)$ in each cell Ω_0 is taken as the local interaction function of Ω_0. The $\tilde{I}_L(f)$ value is then assumed to be constant for the entire cell, and the interaction invariant $\tilde{I}(f)$ is calculated as the value of $\tilde{I}_L(f)$ in Ω_0 multiplied by the cell's area.

While these methods for extending the interaction invariant to higher dimensional datasets have been designed, they are yet to be implemented and tested on real-world datasets. The next step in this research will be to develop software implementations of these methods and evaluate their performance and utility in practice. This will involve testing the methods on various types of datasets, including those with different numbers of variables, different types of interactions, and different sizes. The results of these tests will provide valuable information about the strengths and limitations of these methods, and will guide future improvements and refinements.

3.6. Conclusions

The research herein comprehensively explores the facets of Explainable Artificial Intelligence (XAI) applications that are focused on discerning and elucidating the Multi-Layer Perceptrons (MLPs). Central to this exploration is the calculation of the first, second, and third partial derivatives of the MLP. These calculations underpin three novel methods—Sensitivity Analysis, $\alpha-$curves, and the application of the Interaction Invariant—each offering distinct insights into MLP operation and effectiveness.

Sensitivity Analysis utilizes these partial derivatives to estimate how input variables influence MLP output. Through this process, it is possible to analyse the model's sensitivity relative to each input variable, thereby identifying the variables with the most significant impact. This identification paves the way for model refinement and feature selection by highlighting the most influential input variables. However, it is crucial to remember that the choice of input points can significantly influence the interpretation. The analysis is also susceptible to overlooking non-linear local effects due to the aggregation measures employed.

Meanwhile, the $\alpha-$curves method provides a thorough understanding of the sensitivity distribution across the input space, enabling a more in-depth analysis of model behaviour concerning its input variables. Unlike traditional sensitivity measures, the $\alpha-$curves method offers a more comprehensive view of the sensitivity distribution, considering not only feature importance but also the sensitivity distribution with respect to input variables. Additionally, the $\alpha-$curves method allows for the study of the model's sensitivity behaviour at different detail levels by varying the α value. This flexibility facilitates the detection of both average and localized high sensitivity regions. Furthermore, the $\alpha-$curves method aids in the identification of variables that may be insignificant for most of the input space but exhibit high sensitivity in particular regions. This capability is beneficial for more accurate variable pruning decisions. However, these benefits come at the cost of added complexity, particularly for non-experts. Additionally, the $\alpha-$curves method faces challenges in distinguishing between the effects of interactions between variables and the non-linear effects of a single variable.

Finally, the Interaction Invariant method calculates an interaction measure at each point in a user-defined 2-d-grid, enabling the detection of input interactions. The Interaction Invariant offers a clear measure of interaction between input variables, which might be visualized using a heatmap or a 3-D scatter plot. This understanding enables the identification of regions with strong or weak interactions, leading to a deeper insight of the inner process of the model. Also, the interactions detected by this method remain unaffected by transformations applied to the input or output space. However, this method's limitations include its focus on pairs of input variables, failing to capture higher-order interactions, and that is restricted to ML models with only two input variables. While the full-mesh and sparse method solve the latter, they have yet to be implemented and their functionality assessed.

Collectively, the three methods developed in this chapter make a significant contribution to our understanding and interpretation of ML models. While each method has its strengths and limitations, their combined use can offer a comprehensive understanding of any given model. By deploying these methods, researchers and practitioners can gain valuable insights into their models, identify potential issues, and make informed decisions for refining and enhancing their models.

4 Application of proposed XAI methods: Use cases

No man becomes rich without himself enriching others.
Andrew Carnegie (1835–1919)

This chapter showcases the application of the proposed XAI methods in three use cases of different domains. By employing the developed methods in this thesis and various XAI methods, comprehensive interpretations of neural network model behavior are obtained. Our methods outperform traditional techniques, offering insights into variable influences, feature interactions, and local-level dependencies.

4.1. Introduction

In this chapter, we delve into the practical applications of the novel Explainable AI (XAI) methods based on partial derivatives that have been developed throughout this thesis. These novel methods are demonstrated within three varied use cases, each illustrating the adaptability and versatility of our approach within different domains and for different prediction tasks.

Further, the application cases include the application of various other methods described in the *State-of-the-Art* chapter (section 2.4). This juxtaposition not only serves to highlight the efficacy of the methods we have developed, but also provides a comparative platform, enabling a nuanced understanding of their respective strengths and potential areas of improvement. It also underscores the compatibility of our methods with existing techniques.

The three application cases are as follows:

1. **Case 1: Predicting Nitric Oxides Concentration in Boston** - Using the Boston dataset (Ripley *et al.*, 2011), this case involves predicting the nitric oxides concentration based on various housing data points. The partial derivative based method is applied to gain insights into the factors influencing air quality.

2. **Case 2: Predicting Parkinson Disease Progression** - This case involves the prediction of the evolution of parkinson disease in patients, with a quantitative measure of disease progression one year after the baseline Efron *et al.*, 2004. The predictive model is developed using physiological variables of each patient.

3. **Case 3: Predicting Remaining Useful Life (RUL) of Turbofan Engines** Saxena *et al.*, 2008 - In this case, we apply our methods to predict the Remaining Useful Life (RUL) of turbofan engines using sensor information from the CMAPSS dataset.

Each case is analysed in a dedicated section, where the problem is introduced, the solution approach is outlined, and the results are discussed. These diverse cases not only affirm the wide applicability of the methods developed in this thesis but also exemplify their potential in gaining insights into various fields. The chapter concludes with a synthesis of the insights derived from these applications and their implications for the future of XAI methods based on partial derivatives.

4.2. Case 1: Predicting Nitric Oxides Concentration in Boston

The first application case involves the Boston Housing dataset from the `MASS` package Ripley *et al.*, 2011, a well-known dataset within the machine learning community. This dataset contains 506 instances, each representing different subsections of the Boston residential area. In this study, we focus on five attributes to predict the nitric oxides concentration:

- `zn`: This represents the proportion of residential land zoned for lots over 25,000 sq.ft.
- `rad`: This is an index of accessibility to radial highways.
- `lstat`: This feature indicates the percentage of lower status of the population.
- `indus`: proportion of non-retail business acres per town.
- `age`: proportion of owner-occupied units built prior to 1940.

The task at hand is to predict the nitric oxides concentration (`nox`), a measure of air quality, based on the other housing and demographic attributes. We trained a Multilayer Perceptron (MLP) model to solve this regression task, making it an ideal problem for the application of our developed XAI methods.

Explainable AI (XAI) plays a critical role in this context. Understanding the influence of these housing and demographic attributes on air quality can offer valuable insights for policy-making and urban planning. It allows us to decipher how changes in zoning laws, improvements in public transportation, or socio-economic shifts could impact the air quality in a region. The methods developed in this thesis aim to offer these insights in an interpretable and accessible manner, paving the way for more informed decision-making processes. To understand the influence of selected features (`zn`, `rad`, `lstat`, `indus`, `age`) on the predicted nitric oxides concentration, we applied several interpretability methods, each providing a different perspective and level of insights.

The first method applied was **sensitivity analysis based on partial derivatives** as described in Section 3.3. We used this method to compute the gradients of the MLP model's output with respect to its inputs, which provided an understanding of how small changes in each feature can influence the prediction. By analyzing the distribution of these partial derivatives, we can understand the relationships between the input features and the output.

The second and third methods employed were **Individual Conditional Expectation (ICE)** and **Partial Dependence Plots (PDP)**. ICE and PDP, as outlined in their respective sections 2.4.3 and 2.4.4, gave us both a local (instance-level) and a global (overall model behavior) view of the model's function. They allowed us to visualize how changes in a single feature, while keeping the other features constant, can affect the predicted nitric oxides concentration.

The fourth method applied was the **Lek's Profile** method. As detailed in its respective section 2.4.6, this method creates instance profiles by modifying all feature values simultaneously and visualizes the model's sensitivity to these changes. The application of Lek's Profile offered further instance-level insights into the MLP's decision-making process.

Finally, we applied the **Garson's Importance** and **Olden's Importance** methods. These methods, outlined in section 2.4.9, are specifically designed for Artificial Neural Networks and calculate feature importance by analyzing the connection weights in the network. They provided a global measure of the importance of each feature in the MLP's predictions.

Each of these methods individually offered unique perspectives into the MLP model's behavior and the influence of the input features on the predicted nitric oxides concentration. The results from each method were compared and analyzed to draw a robust and comprehensive understanding of the model's decision-making process.

The data preparation involved standardizing the selected features to have a mean of 0 and a standard deviation of 1. Standardization ensures that each feature contributes to the model's predictions proportionally, preventing features with larger scales from dominating the others and produce misleading analysis.

To create our training and testing datasets, we randomly selected 70% of the dataset instances for training and left the remaining 30% for testing. The **training set** is used to train the model, i.e., it is the data on which the model learns to make predictions. The **test set**, on the other hand, is used to evaluate the model's performance on unseen data. This split allows us to check whether our model has learned the underlying patterns and can generalize well to new, unseen data.

We trained a Multilayer Perceptron (MLP) model with a single hidden layer consisting of 5 neurons and sigmoid activation function. The model's performance was evaluated using two metrics: **Root Mean Square Error (RMSE)** and the **coefficient of determination (R^2 or R-Squared)**.

- RMSE is a commonly used metric for regression models and it provides a measure of the prediction error. Lower RMSE values indicate better model performance.
- R^2 statistic provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.

Metric	Train set	Test set
RMSE	0.39	0.41
R^2	0.85	0.83

Table 4.1. Evaluation metrics of MLP model with 5 neurons to predict the nitric oxides concentration in the `Boston` train and test dataset.

Table 4.1 shows the evaluation metrics for the trained MLP model. The high R^2 values indicate that our model explains a large proportion of the variance in the nitric oxides concentration, and the similar values for training and test set of the evaluation metric indicate that the model can generalize well on new data.

Figure 4.1 shows the results of several XAI methods applied to the MLP model trained on the `Boston` dataset. According to **Garson's Importance** (figure 4.1(c)), the most influential variable is `indus`, followed by `zn`, `rad`, `age`, and `lstat`. This indicates that the proportion of non-retail business acres per town and the proportion of residential land zoned for lots over 25,000 sq.ft. are the most influential in predicting nitric oxide concentrations. This makes sense since areas with higher industrial activity (`indus`) would logically have higher nitric oxide emissions and areas with larger residential lots (`zn`) are likely to have less emissions. The less significant impact of `lstat` can also be explained by it being a more indirect socioeconomic factor.

Olden's Importance suggests similar results (figure 4.1(d)), but `zn` appears to have a negative impact on the nitric oxide concentration. **Olden's Importance** also highlights the importance of `rad`, `age`, and `lstat`, and the direction of the relationships matches the logical expectations. The counterintuitive result is the `indus` variable having the least feature importance, as a neighbourhood with a higher proportion of non-retail business shall be related to higher nox emissions.

The **Sensitivity Analysis based on Partial Derivatives** (figure 4.1(a)) reveals that `indus` and `zn` have a highly non-linear relationship with the target variable, `age` and `rad` exhibit a non-linear relationship, while `lstat` demonstrates an almost-null relationship. This is consistent with our expectations, given that the impact of industrial activity (`indus`) and large residential zones (`zn`) on nitric oxide emissions is likely to be non-linear and complex, while the influence of `lstat` is probably less direct and significant. The `age` variable, assigned the third greatest feature importance, might be related to the amount of nitric emissions due to older houses having less efficient heating systems, which could potentially lead to higher nitric oxide emissions in areas with older housing stock.

The **Feature Plot of Partial Derivatives** (figure 4.1(b)), in addition to corroborating the information derived from figure 4.1(a), provides further insights into the relationship between the output and the inputs. For the `indus` and `rad` variables, it reveals two distinct effects on the output depending on the value of the input variable.

- For `indus`, the plot indicates that a high value of `indus` corresponds to a more negative effect on the output. This reflects the observation that the value of residential houses tends to decrease when non-retail businesses are present in the area. For lower values of `indus`, the effect on the output is ambiguous (it could be either positive or negative) and may depend on other variables.

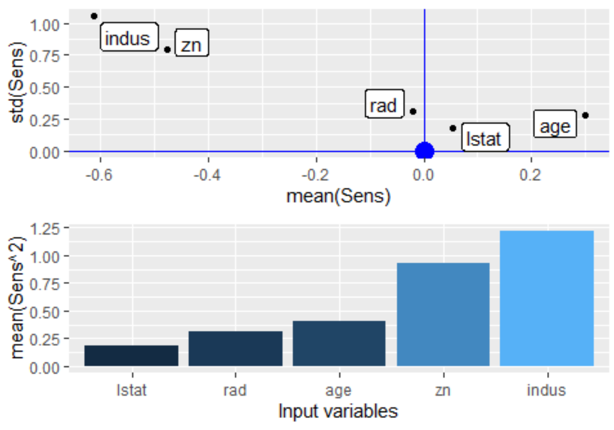
- For `rad`, the plot shows a similar relationship, where higher values of `rad` are associated with lower output values and vice versa. However, this relationship appears to be quadratic, as most negative values of the partial derivatives correspond to higher values of `rad`, and all positive values of partial derivatives correspond to lower `rad` values. This suggests that both very high and very low accessibility to radial highways impact on the concentration of NOx.

Regarding the `zn` variable, it exhibits a non-linear relationship with the output. Specifically, higher values of `zn` appear to have little to no effect on the output, as evidenced by the near-zero values of the partial derivatives. Conversely, lower values of `zn` have a pronounced non-linear impact on the output. This suggests that the proportion of residential land zoned for larger lots can significantly influence the output, particularly when this proportion is low.

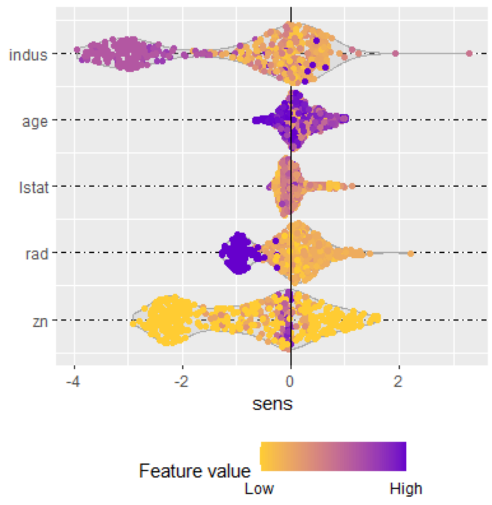
For the `age` and `lstat` variables, there appears to be a minimal effect on the output for most samples. However, it's worth noting that higher values of `age` generally have a positive impact on the NOx concentration, probably due to the lower efficient heating instalations. Conversely, lower values of `age` generally have a positive impact, indicating that newer houses tend to produce less exhausting gases. As for the `lstat` variable, its effect on the output appears to be negligible for most samples, with a few exceptions showing a positive effect on the output. This suggests that the `lstat` variable may not be a significant predictor of NOX concentration in most cases, but there could be specific contexts or neighborhoods where it plays a more substantial role.

Lek's Profile indicates a similar understanding (figure 4.1(e)), suggesting a more linear relationship for `age`, `lstat`, and `zn` variables and a non-linear relationship for `rad` and `indus`. The PDP and ICE methods (figure 4.1(f)) also provide results that align with our expectations. Both methods suggest similar relationships to those observed in the Lek's Profile, strengthening the evidence for our understanding of how these features influence nitric oxide concentrations.

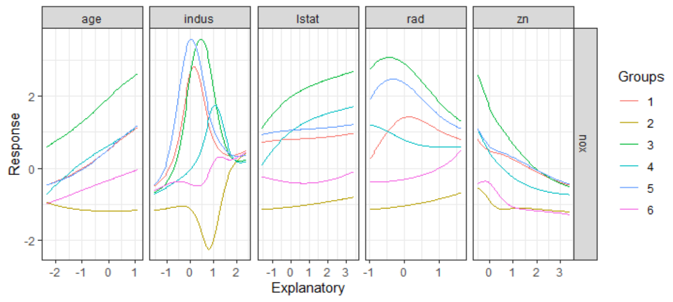
Overall, these results suggest that the model relies most heavily on the `indus` and `zn` features to make its predictions, while `lstat` appears to contribute least. It must be noted that, although the information retrieved from the model is similar along the different XAI methods, the sensitivity analysis developed in this thesis is the only method which provides feature importance and input-output relationship information at the same time. Moreover, the information retrieved using this method is coherent with what could be expected based on the description of the use case variables.



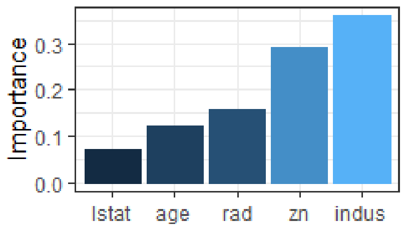
(a) Sensitivity Analysis based on Partial Derivatives



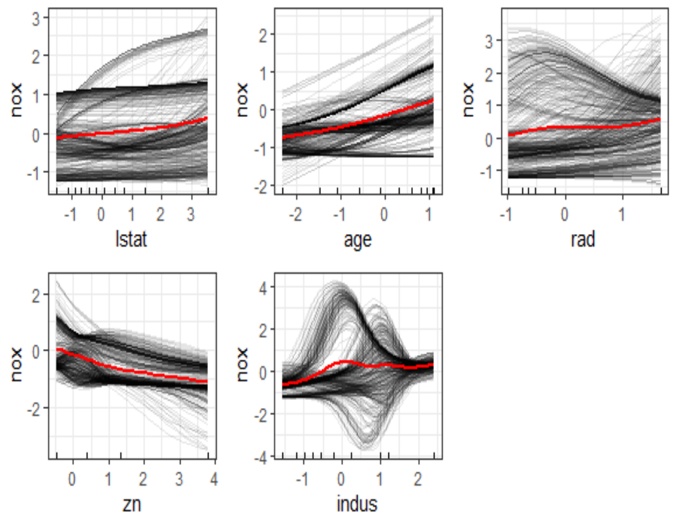
(b) Feature Plot of Partial Derivatives

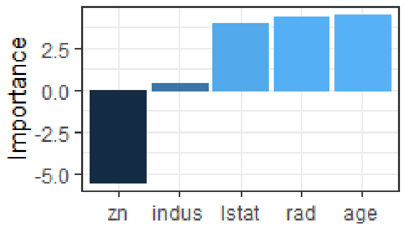


(e) Lek's profile



(c) Garson's Feature Importances





(d) Olden's Feature Importances

(f) Partial Dependence Plots and Individual Conditional Expectation curves

Figure 4.1. Visual representation of results obtained from the different XAI methods: 4.1(a) sensitivity analysis plots based on partial derivatives of the neural network model using the `NeuralSens` package, 4.1(b) feature plot of partial derivatives using the `NeuralSens` package, 4.1(c) Garson's importances of the input features, 4.1(d) Olden's importances of the input features, 4.1(e) Lek's profile method from the `NeuralNetTools` package, 4.1(f) PDP (red) and ICE (black) methods from the `pdp` package.

4.3. Case 2: Predicting Parkinson Disease Progression

The second use case involves predicting the progression of Parkinson's Disease in patients using a dataset from UCI that comprises different patient details, including various vocal fundamental frequency measurements and medical information (Tsanas and Little, 2009). The objective of this dataset, known as the Parkinson's Disease Voice Dataset, is to explore the potential of vocal biomarkers as indicators of Parkinson's Disease progression.

Parkinson's Disease is a neurodegenerative disorder that affects movement and speech, among other functions. One of the notable symptoms of Parkinson's Disease is a change in speech characteristics, which can manifest as tremors in the voice, slurred speech, or other vocal abnormalities. By analyzing the vocal measurements and other pertinent medical information contained in the dataset, it is aimed to build a predictive model that can estimate the progression of Parkinson's Disease in patients, thereby providing a non-invasive, cost-effective, and potentially early indicator of disease progression.

The dataset contains a total of 5875 instances, with 22 baseline variables.:

- **Jitter(%):** The percentage of jitter, which represents the short-term variability of the fundamental frequency.
- **Jitter(Abs):** Absolute jitter, representing the absolute differences in consecutive periods, measured in seconds.
- **Jitter:RAP:** Relative Amplitude Perturbation, a measure of the variability in the amplitude of vocal fold vibration.
- **Jitter:PPQ5:** Five-point Period Perturbation Quotient, a measure of the variability in pitch period size over five pitch periods.
- **Jitter:DDP:** Dimensionless Drift Parameter, a composite measure derived from RAP.
- **Shimmer:** A measure of the amplitude variability of the vocal fold vibration.
- **Shimmer(dB):** The logarithmic measure of shimmer, expressed in decibels.
- **Shimmer:APQ3:** Three-point Amplitude Perturbation Quotient, a measure of the variability in amplitude over three pitch periods.
- **Shimmer:APQ5:** Five-point Amplitude Perturbation Quotient, a measure of the variability in amplitude over five pitch periods.
- **Shimmer:APQ11:** Eleven-point Amplitude Perturbation Quotient, a measure of the variability in amplitude over eleven pitch periods.
- **Shimmer:DDA:** A composite measure derived from APQ measures.
- **NHR:** Noise-to-Harmonics Ratio, a measure of the ratio of noise to tonal components in the voice signal.
- **HNR:** Harmonics-to-Noise Ratio, a measure of the ratio of tonal components to noise components in the voice signal.

- **RPDE:** Recurrence Period Density Entropy, a non-linear measure that quantifies the predictability and complexity of a signal.
- **DFA:** Detrended Fluctuation Analysis, a method for determining the statistical self-affinity of a signal.
- **PPE:** Pitch Period Entropy, a measure of the regularity and stability of the pitch.
- **Sex:** The gender of the individual, which can impact the manifestation of Parkinson's Disease symptoms.
- **Age:** The age of the individual, which could be a significant factor in the progression of Parkinson's Disease (not used in this analysis).
- **Motor UPDRS:** Unified Parkinson's Disease Rating Scale; Motor section score, a measure of motor function (not used in this analysis).
- **Total UPDRS:** Unified Parkinson's Disease Rating Scale; Total score, a comprehensive measure of disease progression (used as the target variable).

The target variable, in this case, is the Total UPDRS, a quantitative measure of disease progression. Predicting this output based on the selected input variables is an important task for proactive healthcare and disease management, enabling timely interventions that could potentially slow down the disease progression or manage the symptoms more effectively. The importance of explainable AI methods in this context is to provide a clear understanding of which factors are contributing most to the prediction of disease progression, and how they are doing so. This will aid medical professionals in devising the most suitable treatment plans for their patients.

Before training the model, it is crucial to understand the inter-relationships among the input variables. A widely used tool for this purpose is the correlation matrix, which provides a numerical and visual representation of how variables interact with each other.

The correlation matrix was computed for all the variables present in the Parkinson's Disease Voice Dataset. Each cell in the matrix represents the correlation coefficient between two variables, ranging from -1 to 1. A correlation coefficient close to 1 implies a strong positive correlation, while a coefficient close to -1 implies a strong negative correlation. A coefficient near 0 indicates no linear relationship between the variables.

Figure 4.2 showcases the correlation matrix of the dataset variables. The color-coded matrix makes it visually intuitive to identify the relationships. For instance, a darker color indicates a stronger correlation.

From the correlation matrix, it was observed that some variables exhibited high collinearity, which may lead to multicollinearity issues in the model. For example, the sets of Jitter and Shimmer variables were highly correlated among themselves. To mitigate this, only one representative from each set, namely `Jitter(Abs)` and `Shimmer`, were retained for the analysis.

Additionally, the `Age` variable was excluded from the model as it acted as an identifier for the patients: its values remained constant during the trials, hence providing information to the model that is not expected to be present in unseen data. This removal is in alignment with the best practices of data preprocessing to ensure that the model generalizes well to new data, rather than overfitting to idiosyncrasies in the training data.

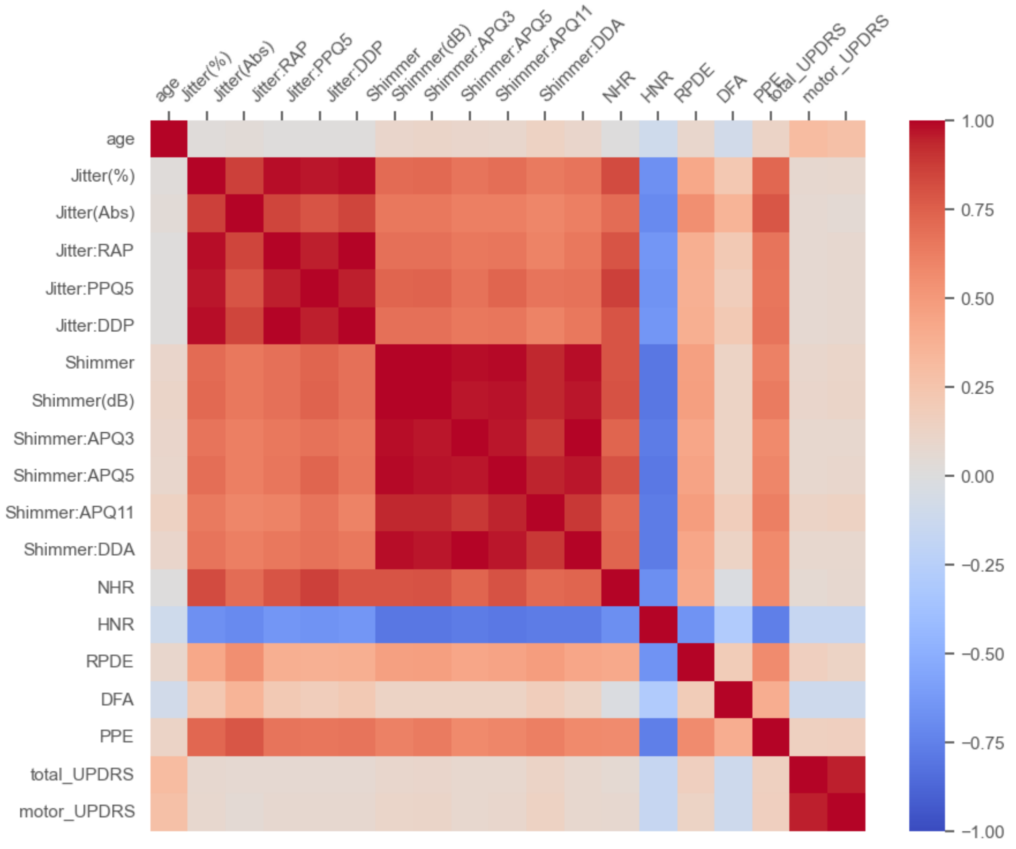


Figure 4.2. Heatmap of the correlation between features in the Parkinson's Disease Voice dataset.

Regarding the XAI methods used in this case, the first method applied on this use case was **sensitivity analysis based on partial derivatives**, which was explained in the previous use case. In order to extract a deeper information of how the model is using the input variables to predict the output, the second method developed in this thesis, $\alpha-$**curves**, was used to analyze the partial derivatives distributions of the model.

Secondly, we applied **SHAP** (Section 2.4.7). This approach applies concepts from cooperative game theory to quantify the contribution of each input feature to the model's prediction for an instance. By evaluating all possible combinations of features and calculating their average marginal contribution to the prediction, SHAP values are obtained, providing a comprehensive understanding of the role of each feature in the model's decision process.

Lastly, **Permutation Importance** was performed (Section 2.4.8). This method quantifies the importance of each feature by evaluating the impact on the model's performance when the feature values are randomly shuffled. The change in performance serves as an indicator of the feature's importance: the larger the decrease in performance upon shuffling a feature's values, the more important the feature is considered to be.

As in the previous use case, the variables are standardized to have a mean of 0 and a standard deviation of 1. The dataset was randomly split into a training set and a test set, with 80% of the data used for training the model and the remaining 20% reserved for testing. The model trained was a Multilayer Perceptron (MLP) with 15 neurons in the hidden layer.

Table 4.2 shows the evaluation metrics for the trained MLP model. Although the metrics might appear suboptimal when compared to other use cases, they are typical in the realm of

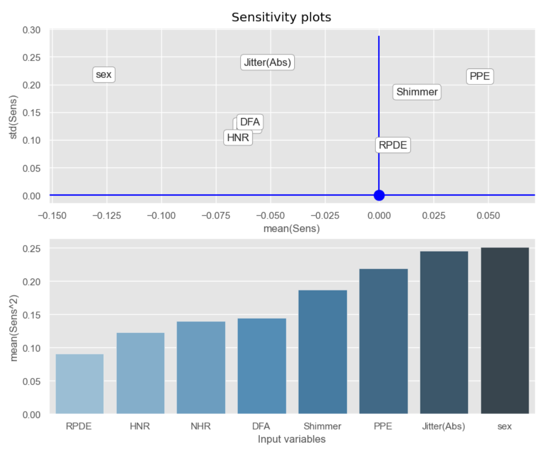
Metric	**Train set**	**Test set**
RMSE	0.17	0.18
R^2	0.41	0.33

Table 4.2. Evaluation metrics of MLP model with 5 neurons to predict the parkinson disease progression indicator in the `Parkinson Disease` train and test dataset.

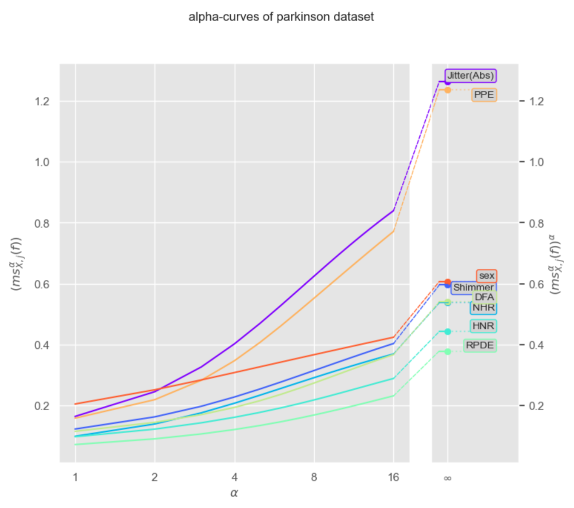
medical datasets. The nature of medical data often encompasses a high degree of variability and noise owing to the myriad of factors that can influence health outcomes (El Khatib *et al.*, 2022). Moreover, medical datasets are frequently characterized by a lack of large sample sizes due to the challenges in collecting such data, which further compounds the difficulty in achieving higher predictive accuracy (Faber and Fonseca, 2014). Despite this, even modest predictive accuracy in medical contexts can hold significant value, enabling healthcare providers to glean insights and make better-informed decisions in patient care and treatment planning (Nguyen *et al.*, 2019).

Following the methodology for sensitivity analysis, figure 4.3(a) reveals a non-linear correlation among all variables in predicting Parkinson's Disease (PD) progression. The most crucial variables identified are the patient's sex (`sex`), absolute jitter (`Jitter (Abs)`), and Pitch Period Entropy (`PPE`), followed by `Shimmer`. Conversely, `DFA` and the noise-related variables `NHR`, `HNR` and `RPDE` are deemed less significant, with `DFA` being the most significant by a little margin. According to literature, Azadi *et al.* (2021) determines that jitter emerged as a pivotal parameter for differentiating PD patients due to its capability to measure frequency changes from cycle to cycle in speech signals. It also highlights the sex of the individual (`sex`) as a crucial variable in this analysis. They found the substantial differences in speech characteristics between men and women, thus necessitating a segregated analysis for male-only and female-only populations. It was observed that the values of the extracted jitter and shimmer features varied distinctly between male and female subjects when compared between PD patients and healthy individuals. This implies that the sex of the individuals significantly impacts the acoustic parameters being studied, hence influencing the diagnostic accuracy. Vizza *et al.* (2019) found that the amplitude variability of the vocal fold vibration, represented by `Shimmer`, exhibit significant variations in PD patients compared to healthy controls. This correlates with the importance in the `Shimmer` variable to diagnose PD based on voice recordings. Regarding `PPE`, Little *et al.* (2008) found that this measure provides a nuanced assessment of abnormal pitch variations, distinguishing PD-induced dysphonic variations from natural pitch variations. By analyzing pitch on a perceptually-relevant, logarithmic scale, `PPE` more accurately captures the non-Gaussian fluctuations in pitch period variation associated with PD-related dysphonia. This methodological shift offers a more precise tool for analyzing voice disorders in PD, enhancing the diagnosis of the disease. Regarding `HNR`, `NHR` and `RPDE` variables, Lahmiri (2017) defends the potential of discerning PD progression through noise-associated variables. Yet, newer studies (Upadhya and Cheeran, 2018; Romero Arias *et al.*, 2023) suggests that these metrics are not as significantly influenced by the disease compared to the Jitter, Shimmer, or PPE measures derived from the patients' voice recordings. In the case of `DFA`, Minamisawa *et al.* (2009) and Kirchner *et al.* (2014) found `DFA` to be a suitable indicator of PD, although Miranda *et al.* (2022) found this variable not as important as `Jitter` regarding model performance.

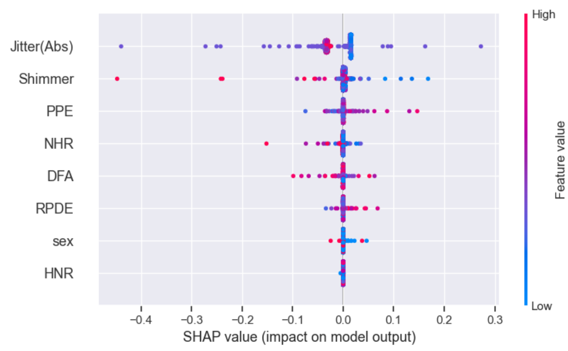
Although results from sensitivity analysis are coherent with the literature reviewed, more information can be retrieved using the $\alpha-$curves methodology presented in this paper. Figure 4.3(b) shows that, on a global scale, `sex` is the variable with the highest importance. Nonetheless,



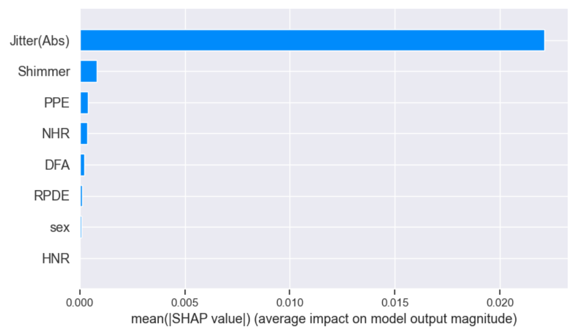
(a) Sensitivity Analysis based on Partial Derivatives



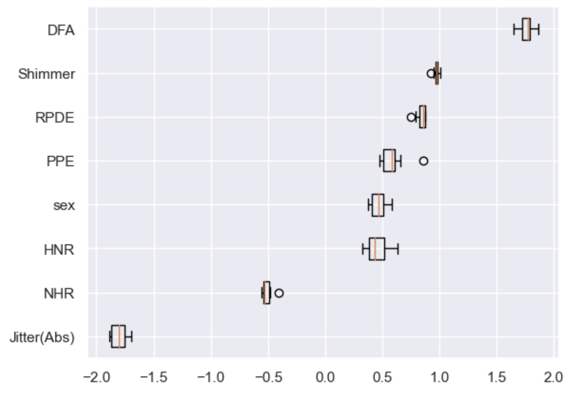
(b) $\alpha-$curves analysis



(c) SHAP Values



(d) SHAP Feature Importance



(e) Permutation Importance

Figure 4.3. XAI techniques plots of parkinson disease progression dataset. 4.3(a) shows sensitivity analysis from `neuralsens` package, 4.3(b) shows the $\alpha-$curves analysis plot, 4.3(c) shows the shap values for each sample of the dataset, 4.3(d) shows the shap feature importances and 4.3(e) shows the input permutation importance.

the variables `Jitter (Abs)` and `PPE` have nearly as much relevance as `sex` globally, yet they are more important for specific cohorts of patients to determine the PD progression. Notably, by $\alpha = 4$, they are already the most influential variables, indicating their notable impact across the majority of patients. This trend might be associated with different subtypes of PD, as elucidated in Tsanas and Arora (2022), where distinct PD subtypes exhibited differential impacts on a patient's voice frequency. At a local level, both `Jitter (Abs)` and `PPE` markedly surpass the rest of the variables in importance. Here, the $\alpha-$curves could be highlighting those patients for whom the PD subtype significantly influences the `PPE` and `Jitter (Abs)` variables. On the contrary, the small slope of the α-curve of the `sex` variable denotes a comparatively consistent impact of the patient's gender on how PD progression affects the patient's voice capabilities. This is corroborated by Azadi *et al.* (2021), who found that even amidst the presence of gender-specific PD symptoms, voice alterations attributable to PD maintained a consistency within each gender group. With respect to `Shimmer`, its ascending curve reflects a low global-level relevance of the variable, with a subgroup of patients where the importance of this variable is similar to the `sex` variable. This potentially correlates once again with PD subtypes, where an specific group of patients presents deeper symptoms of voice amplitude variability than others. Regarding the rest of the variables, `DFA`, `NHR`, `HNR`, and `RPDE` all exhibit a lesser impact on the analysis. Among these, the average impact of `DFA`, `NHR`, and `HNR` is similar to that of `Shimmer`, albeit slightly lower, while `RPDE` demonstrates a significantly lower impact. Interestingly, `DFA` and `NHR` show a degree of relevance in certain portions of the dataset, nearly as much as `Shimmer` or `sex`. The curves of these two variables are almost identical and almost parallel to that of `Shimmer`, albeit lying below it, suggesting that the magnitude of the regions (i.e., types of patients) where these three variables are relevant for PD diagnosis might be similar. Conversely, `HNR` and `RPDE` are less relevant than the others at any level of analysis, being the two least influential variables in the dataset for PD diagnosis. Figure 4.3(e) shows the input permutation results. Among these, the high negative importance associated with `Jitter(Abs)` is particularly noteworthy as it contrasts with the reviewed literature, which often underscores the significant positive role of jitter in differentiating Parkinson's Disease (PD) patients. Similarly, `DFA` showcasing the highest importance is not consistent with the revised literature (Miranda *et al.*, 2022), where this variable is often described as having lesser or varied importance across different studies. On the other hand, the positive importance of `Shimmer` aligns well with literature (Vizza *et al.*, 2019), reaffirming its relevance in PD progression analysis. The lower importances of `PPE` and `sex` hint at a possible oversight of their interactions with other variables or their nuanced influence in PD progression which might not be fully captured in the permutation importance analysis. Analyzing SHAP results presented in figures 4.3(c) and 4.3(d), it emphasizes the importance of the `Jitter (Abs)` variable, with a substantial drop in importance for the `Shimmer` variable, and virtually no significance attributed to the remaining variables. However, the summary plot (4.3(c)) mainly showcases a non-linear relationship between all variables and the output, without providing much more information. It does hint at a diverse level of variable contributions across different patients, suggesting that certain variables may have a higher impact on the output for some individuals. It shall be noted that the SHAP analysis operates under the assumption of input variables' independence and local linearity for each sample, which might overlook potential interactions among variables .As a result, this could lead to a scenario where some variables appear irrelevant, whereas their effects may only be discernible when considered in conjunction with other variables.

Figure 4.3(e) shows the input permutation results. Among these, the high negative importance associated with `Jitter(Abs)` is particularly noteworthy as it contrasts with

the reviewed literature, which often underscores the significant positive role of jitter in differentiating Parkinson's Disease (PD) patients. Similarly, `DFA` showcasing the highest importance is not consistent with the revised literature (Miranda *et al.*, 2022), where this variable is often described as having lesser or varied importance across different studies. On the other hand, the positive importance of `Shimmer` aligns well with literature (Vizza *et al.*, 2019), reaffirming its relevance in PD progression analysis. The lower importances of `PPE` and `sex` hint at a possible oversight of their interactions with other variables or their nuanced influence in PD progression which might not be fully captured in the permutation importance analysis.

Both SHAP and Permutation importance analyses seemingly downplayed the importance of the `sex` variable, a deviation which is not only contrary to the results from the α-curves and sensitivity analysis but also discordant with the prevailing literature on the subject. To delve deeper into this inconsistency, an alternative approach was adopted to determine the importance of the `sex` variable. By deliberately inverting the values of the `sex` variable — converting 1s to 0s and vice versa — a comparative analysis of the model's output between the original and modified `sex` values was performed.

Figure 4.4 show the absolute value of the difference, as percentage of the model's output range, in the model's predictions between the original and the modified `sex` variable. The model produce notably different predictions between the original and altered `sex` values, underlining a pronounced sensitivity to this particular variable. This clearly suggests that the `sex` variable, despite its understated importance in the SHAP and Permutation importance analyses, holds notable significance in the prediction of Parkinson's Disease progression based on voice recordings metrics.

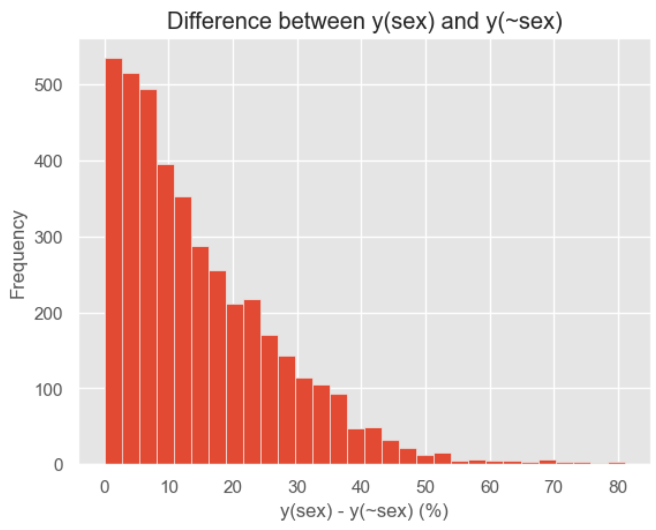


Figure 4.4. Difference, in absolute value, between model predictions for original versus modified `sex` values as percentage of the model's output range.

Based on the information retrieved from the previous methods, the only common information is the relevance of the `Shimmer` variable to predict PD progression. Importance assigned for the rest of the variables depends on the method used to analyze the model, being sensitivity analysis and the $\alpha-$curves method the most coherent with the reviewed literature.

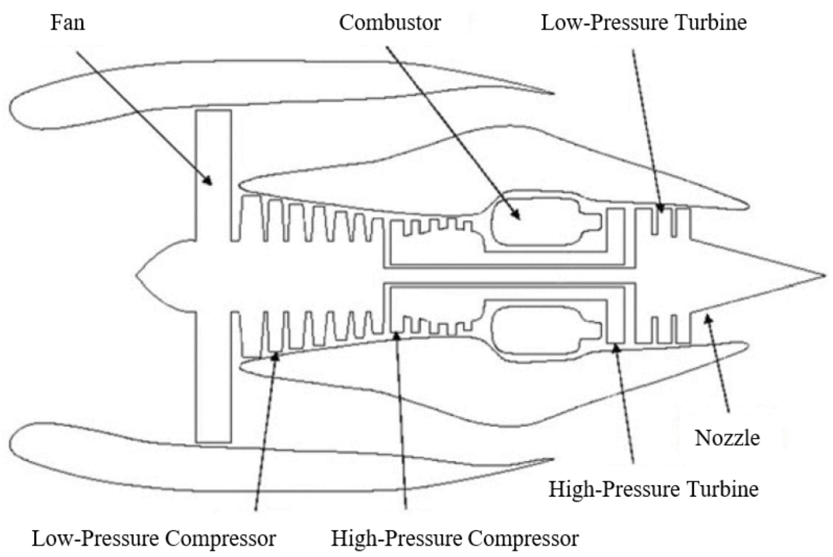


Figure 4.5. Structure of the physics-based simulation model of the C-MAPSS dataset.

4.4. Case 3: Predicting Remaining Useful Life (RUL) of Turbofan Engines

The third use case centers on predicting the Remaining Useful Life (RUL) of turbofan engines using the CMAPSS dataset. The dataset includes sensor information from these engines, simulating real-world conditions where preventive maintenance and failure prediction are crucial. The prediction task is to estimate the RUL, i.e., the number of operational cycles an engine has left before it requires maintenance, which is a regression problem. As in the previous cases, an MLP model was trained for this task, and our XAI methods were applied.

One area where machine learning has demonstrated substantial potential is in predictive maintenance, where it can help to identify early signs of system degradation and facilitate timely intervention. This is of paramount importance in fields such as aerospace, where the failure of critical components can lead to catastrophic consequences.

The NASA Commercial Modular Aero-Propulsion System Simulation (CMAPSS) dataset is a widely-used benchmark for developing and evaluating machine learning models for predictive maintenance. This dataset contains sensor measurements and operational data from jet engines, simulating various degradation scenarios and engine wear. Analyzing this dataset can provide valuable insights into the factors influencing engine performance and degradation, and ultimately help to prevent engine failures.

The objective of this study is to analyze the CMAPSS dataset using a MLP model to predict the remaining useful life (RUL) of jet engines (Saxena *et al.*, 2008), and subsequently apply post-hoc explainability methods to identify the most important variables contributing to these predictions. By doing so, we aim to provide a better understanding of the underlying factors driving engine degradation, and to inform more effective maintenance strategies.

The C-MAPSS dataset is generated through a physics-based simulation model that represents a particular type of turbofan engine, specifically a two-spool engine as depicted in figure 4.5. With the same background model, data is recorded in run-to-failure scenarios (i.e., engines operate normally in the beginning but develop a fault over time) under various conditions. To replicate real-world scenarios, the data is also intentionally corrupted with noise from different sources. Data is stored in four different dataset of increasing complexity based on the simulation conditions (see Table 4.3 for an overview of each dataset).

Table 4.3. Overview of turbofan datasets

Dataset	Operating conditions	Fault modes	Train size (# of engines)	Train size (# of samples)	Test size (# of engines)	Test size (# of samples)
FD001	1	1	100	20631	100	13096
FD002	6	1	260	53759	259	33991
FD003	1	2	100	24720	100	16596
FD004	6	2	248	61249	249	41214

The engines in the training sets are run to failure, while in the test sets the time series end in an unknown timestep before failure. The goal is to predict the Remaining Useful Life (RUL) of each turbofan engine in the last sample of the test set.

Each row of the datasets contains the following information:

- Engine unit number
- Operational cycle
- 3 operational conditions – altitude, speed and fuel usage.
- 21 Sensor readings from different locations in the turbofan engine Saxena *et al.*, 2008K. Liu *et al.*, 2013. These sensor data are contaminated with sensor noise. Table 4.4 shows an overview of these sensors.

Table 4.4. C-MAPSS Sensors Overview

Index	Description	Units
1	Total temperature at fan inlet	ºR
2	Total temperature at LPC outlet	ºR
3	Total temperature at HPC outlet	ºR
4	Total temperature at LPT outlet	ºR
5	Pressure at fan inlet	psia
6	Total pressure in bypass-duct	psia
7	Total pressure at HPC outlet	psia
8	Physical fan speed	rpm
9	Physical core speed	rpm
10	Engine Pressure ratio	-
11	Static pressure at HPC outlet	psia
12	Ratio of fuel flow to Ps30	pps/ppia
13	Corrected fan speed	rpm
14	Corrected core speed	rpm
15	Bypass ratio	-
16	Burner fuel-air ratio	-
17	Bleed enthalpy	-
18	Demanded fan speed	rpm
19	Demanded corrected fan speed	rpm
20	HPT coolant bleed	lbm/s
21	LPT coolant bleed	lbm/s

The datasets are arranged in an n-by-26 matrix, where n corresponds to the number of samples in the dataset. Each row corresponds to the value of each variable during an operational cycle and each columns to a different variable. For the sake of simplicity, in this use case only the dataset FD001 is analyzed to illustrate the methodology proposed. The FD001 dataset was entirely simulated at sea level (1 operating condition) with High-Pressure Compressor (HPC) degradation (1 fault mode). As this dataset only contemplates one operational condition, the three variables related to this information are discarded. Furthermore, variables with information of Sensors 1, 5, 6, 10, 12, 15, 16, 18 and 19 are discarded due to not vary along the samples of the dataset.

For the output variable RUL, it is not provided in the training dataset. Consequently, user must design a methodology to assign the output value for each of the samples provided. The Remaining Useful Life (RUL) variable represents the number of operational cycles remaining before an engine experiences a failure. However, it is standard practice to apply a saturation point to the RUL variable to prevent unrealistic extrapolation. For this analysis, a saturation point of 130 cycles was utilized. This means that any RUL exceeding 130 cycles is assigned a value of 130. This truncation strategy limits the maximum observable RUL value, thereby enhancing the model's ability to generalize by attenuating the influence of extremely high RUL values. Furthermore, it enhances the model's prediction accuracy, particularly for more imminent engine failures, which are generally the primary concern in predictive maintenance applications.

Due to the high correlation among the input features, the CMAPSS dataset required a more complex preprocessing phase. This high correlation was observed and verified through the correlation matrix depicted in figure 4.6. In such cases, machine learning models might struggle to extract meaningful information due to the redundancy in the input data, leading to poor performance on unseen data.

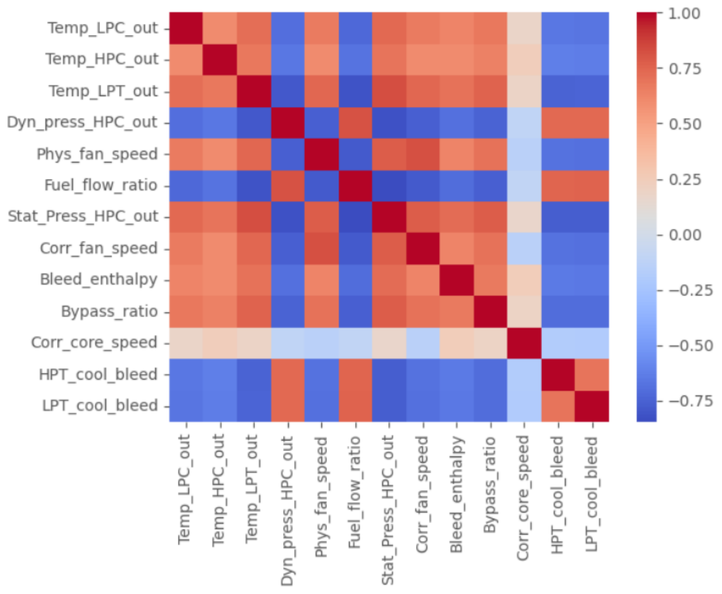


Figure 4.6. Heatmap of the correlation between the input features in the CMAPSS dataset.

To address this issue and enhance the model's learning capability, we employed Principal Component Analysis (PCA), a well-established technique for dimensionality reduction and alleviating multicollinearity issues. PCA was applied to all variables except Sensors 9 (`Phys_core_speed`) and 14 (`Corr_core_speed`) due to their almost null correlation with the other variables. The high correlation between Sensors 9 and 14 force us to choose only one as input variable. After some initial tests, Sensor 14 was chosen as it produces the most accurate model. The first Principal Component (PC1), which captures the largest variance in the data, was selected as the primary input to the model. The weights associated with `PC1` are shown in figure 4.7. The interpretation of these weights, in terms of temperature, pressure, and

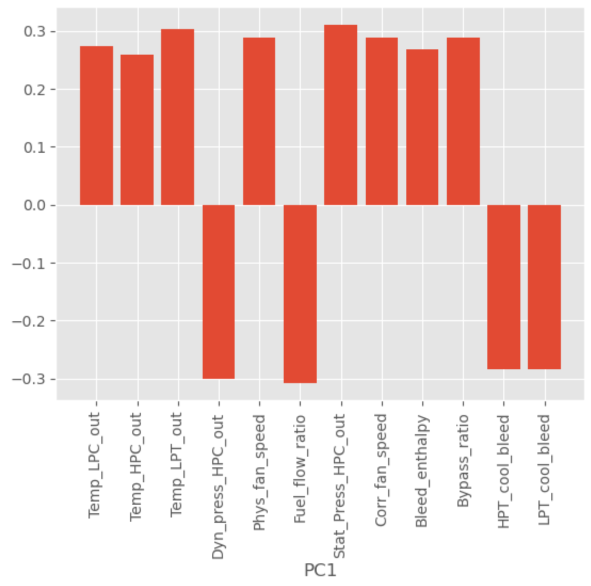


Figure 4.7. Loadings associated to each input variable of Principal Component 1 (PC1).

aircraft-related measurements, is as follows:

- `Temp_LPC_out`, `Temp_HPC_out`, `Temp_LPT_out`: These variables denote the temperature outputs of the Low Pressure Compressor (LPC), High Pressure Compressor (HPC), and Low Pressure Turbine (LPT), respectively. Their positive weights suggest that an increase in these temperatures corresponds to an increase in the `PC1` value. This is indicative of higher engine performance or potentially elevated stress on the components.

- `Dyn_press_HPC_out`, `Fuel_flow_ratio`, `HPT_cool_bleed`, `LPT_cool_bleed`: These variables carry negative weights, implying that an increase in these variables leads to a decrease in the `PC1` value. `Dyn_press_HPC_out` represents the dynamic pressure output of the HPC. A rise in dynamic pressure could suggest higher airspeed or potential pressure irregularities. `Fuel_flow_ratio`, `HPT_cool_bleed`, `LPT_cool_bleed` are related to fuel efficiency and the cooling of high and low pressure turbines. An increase in these variables might hint at inefficiencies or over-utilization of cooling mechanisms.

- `Phys_fan_speed`, `Stat_Press_HPC_out`, `Corr_fan_speed`, `Bleed_enthalpy`, `Bypass_ratio`: These variables exhibit positive weights, indicating that an increase in these variables enhances the `PC1` value. The variables `Phys_fan_speed` and `Corr_fan_speed` could be indicative of the thrust generated by the engine. `Stat_Press_HPC_out` refers to the static pressure output from the HPC, which could denote air density and the operating conditions of the aircraft. The variables `Bleed_enthalpy` and `Bypass_ratio` could provide insights about energy extraction and thrust efficiency of the engine.

These interpretations are indicative of how changes in these variables might influence the `PC1` value, which is designed to capture as much variation in the original data as possible. In summary, a higher value of `PC1` suggests an operating scenario with higher temperatures at the compressor and turbine stages, increased fan speed, higher static pressure output from the High Pressure Compressor (HPC), and increased energy extraction and thrust efficiency. Simultaneously, it would suggest reduced dynamic pressure output from the HPC, reduced fuel efficiency, and potentially over-utilization of turbine cooling mechanisms.

In addition to `PC1`, Sensor 14 (`Corr_core_speed`) was also utilized as an input variable. Despite its high correlation with Sensor 9, it was chosen over the latter due to the lower correlation between this variable and the rest of the features.

This tailored methodology for the CMAPSS use case is expected to enhance the model's ability to generalize to new data by reducing multicollinearity and redundancy in the input data while retaining the most significant predictors. Previous to training the model, all input variables were scaled to have a mean of 0 and standard deviation of 1.

The dataset was randomly split into a training set and a test set, with 80% of the data used for training the model and the remaining 20% reserved for testing. The model trained was a Multilayer Perceptron (MLP) with 6 neurons in the hidden layer.

Metric	**Train set**	**Test set**
RMSE	19.73	18.95
R^2	0.79	0.58

Table 4.5. Evaluation metrics of MLP model with 6 neurons to predict the Remaining Useful Life in the `CMAPSS FD001` train and test dataset.

.

In this use case, we applied three XAI methods to detect interactions present in the trained MLP model, namely Partial Dependence Plots (PDP) for two variables, Friedman's H-index and the interaction invariant developed in the thesis.

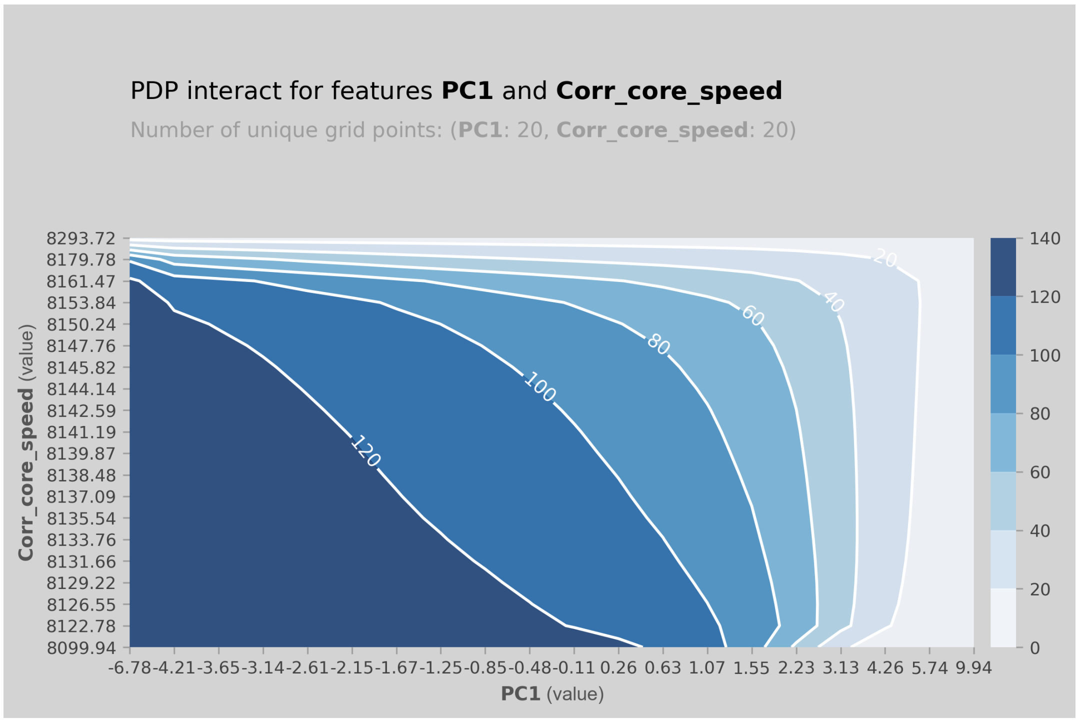
As explained in section 2.4.4, **Partial Dependence Plots** (PDP) provide a visual interpretation of the relationship between a set of features and the predicted outcome. In this case, we employed PDP for two variables to capture and illustrate the interaction effect between the selected features. This approach provides a clear visual representation of how changes in these variables collectively impact the MLP model's predictions.

Following PDP, we applied **Friedman's H-index**, a technique used for detecting and quantifying interactions between features in a predictive model. Described in detail in section 2.4.5, this method computes the strength of interaction between pairs of features by assessing the improvement in the model's prediction error when considering both features together versus independently. A higher H-index value signifies a stronger interaction between the features.

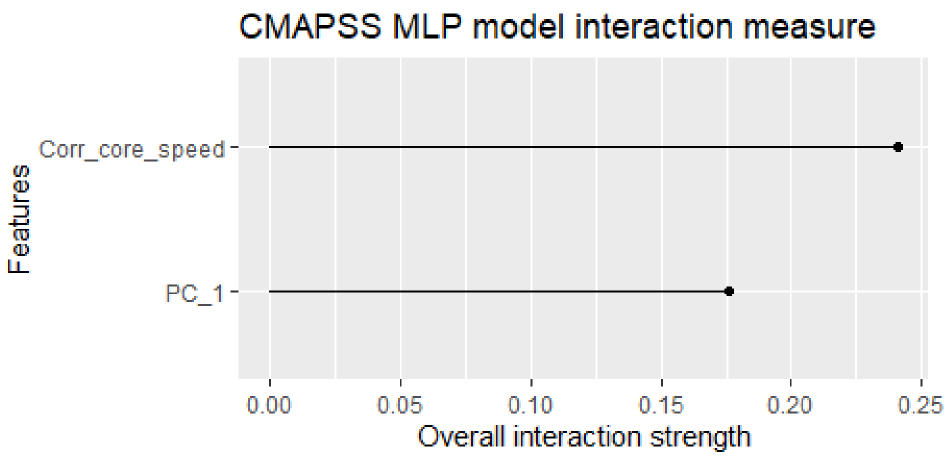
Lastly, we applied the **Interaction Invariant** method as described in section 3.5, which not only quantifies the interactions between features, but also provides visualization techniques similar to PDP.

Figures 4.8 and 4.9 shows the plots resulted from applying the previous mentioned techniques on the trained MLP model.

Figure 4.8(a) shows the PDP analysis for both the `PC1` and `Corr_core_speed` variable. In this plot, it is shown that for lower values of both variables (yellow region), the RUL of the engine is the highest. The RUL decreases as any of the input variables increase. In this case, the iso-level



(a) Partial Dependence Plot for 2 variables



(b) Friedman H Index

Figure 4.8. XAI techniques plots of CMAPSS dataset. 4.8(a) shows the 2 variable Partial Dependence Plots, 4.8(b) shows the Friedman H interaction index.

curves suggest that the effect of one input variable depends on the value of the other, as it is not the same the effect of the core accelerating when the turbine is cool (maximum RUL) or when it is heated (minimum RUL). For example, when `PC1` $= 0$, increasing `Corr_core_speed` from 8100 to 8150 results in a decrease of the output variable from 120 to 100. However, when `PC1` $= 3$, doing the same variation to `Corr_core_speed` leads to an increase of the output variable from 20 to 40. This indicates an interaction between both input variables as the effect of changes in one of the inputs depends on the values of the other variable.

Figure 4.8(b) shows the values of Friedman interaction index for each one of the variables. As this dataset only has two input variables, the only interaction term is between `PC1` and `Corr_core_speed`. It must be noted that this interaction term shall not be equal for both variables, as it measures the extent to which the effect of two features on the predicted output

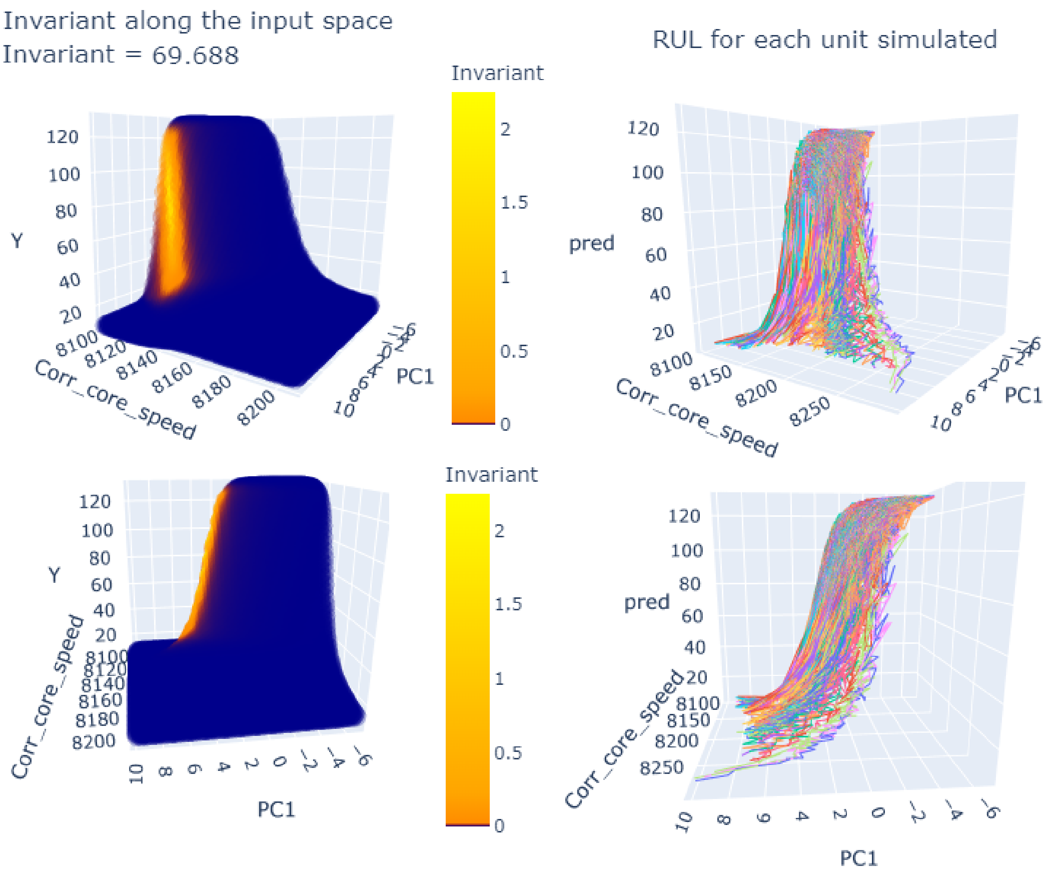


Figure 4.9. Interaction invariant obtained on the mlp model trained on the CMAPSS dataset.

changes when they're considered together, versus when they are considered individually. In this case, the higher interaction term for `Core_core_speed` is due to the partial dependence with respect to this variable being higher than the partial dependence with respect to `PC1`.

Figure 4.9 illustrates the interaction invariant as outlined in 3.5, along with the evolution of RUL across the input space. It also quantifies the interaction measure as calculated by the invariant. The Interaction Invariant analysis of the CMAPSS model uncovers an interaction between `PC1` and `Corr_core_speed`. The 3D plots provided by the interaction invariant plot presents two distinct behaviors, forming a hill-like structure. The peak of the hill represents all the samples with optimal health of the turbofan motor, while the lower points correspond to a null Remaining Useful Life (RUL) of the turbofan motor. The hill exhibits two different slopes, one influenced by the varying values of the `Corr_core_speed` variable and the other by the `PC1` variable. On one slope, characterized by lower values of PC1, there is no implicit interaction and the output value depends heavily on the value of the `Corr_core_speed` variable.

Before we delve deeper into the interpretation derived from the interaction invariant, it is crucial to comprehend what the MLP model is accomplishing. The dataset being modelled is obtained from a physical model which, based on a starting level of degradation of the HPC, simulates the behavior of the turbofan motor and collect the sensor data. During this simulation, the engine tries to maintain an specific level of speed even if the HPC is degrading and, therefore, the overall efficiency of the motor is decreasing. As efficiency decreses, an increase in each input variable might be triggered by the engine's need to enhance fuel consumption and thrust production to adhere to the prescribed speed command. This rise in input variables is reflected in the sensor data, which we utilize to estimate the RUL of the motor during a simulated flight.

Consequently, the MLP can be perceived as a tool to model the impact of degradation level on the RUL of the motor based on the sensor data, with the degradation level acting as a hidden variable resulting from a transformation of the sensor input variables.

Going back to figure 4.9, let us analyze the slope corresponding to low levels of PC1 variables. This slope present no implicit interaction, where a suitable transformation of the input and output space might detach the explicit interaction of the two input variables. We can understand this region as a flight regime where, an increase in the Corr_core_speed or in the PC1 variable (higher temperature/pressures), have an independent and additive relationship with the level of degradation hidden variable. Moreover, this slope contains most of the simulated flights where Corr_core_speed steadily increases. This regime might suggest a high-stress scenario where the motor needs to constantly increase revolutions to follow the commanded speed. In this regime, some of the increase revolutions might be substituted by an increase on the PC1 variable in order to maintain the produced thrust, given by the fact that the effect of each input variable are independent.

The other slope, characterized by high levels of PC1 variable and low levels of Corr_core_speed (in regions where Corr_core_speed $\approx$ 8120), presents an implicit interaction between both input variables. This implicit interaction suggests that the effect of each input variable is not independent and the hidden variable can not be modelled as a superposition of both effects. Analyzing the Corr_core_speed values in this region, a speed of Corr_core_speed = 8120 appears to represent a nominal flight condition, being the starting speed for most flights with an optimal motor health. The presence of implicit interaction in this region suggest that, near this nominal core speed, the impact of the PC1 variable is not independent of the core speed, unlike the other slope.

In conclusion, all three XAI methods detected the interaction between the two input variables. The PDP analysis effectively demonstrated the independent effects of the PC1 and Corr_core_speed variables on the RUL of the engine. While it illustrated the general behavior of these variables, the PDP lacked in highlighting the complex interaction scenarios between them. The Friedman H index, on the other hand, quantified the interaction between the two variables but was limited in its ability to depict the complex interactive landscape over different operational regimes. It provided a numerical measure of interaction but did not offer an explicit visual intuition about the nature of these interactions. In contrast, the interaction invariant method stands out for its capacity to visually represent and quantitatively measure complex interactions between the variables. It successfully unveiled regions of distinct interaction behaviors, allowing us to infer different flight regimes and the corresponding engine responses. By identifying an absence of implicit interaction under high-stress scenarios and uncovering implicit interactions near the nominal core speed, the interaction invariant method provided the most comprehensive and detailed interpretation of the MLP model's behavior.

4.5. Conclusions

This chapter explored the practical application of the methods developed in this thesis through three distinct use cases, including the prediction of NOx emissions in the Boston dataset, the prediction of parkinson disease progression measure, and the prediction of Remaining Useful Life (RUL) in the CMAPSS FD001 dataset. Across each case, we implemented and assessed the proposed Sensitivity Analysis based on Partial Derivatives, $alpha-$curves, and the interaction invariant, juxtaposed with an array of existing XAI techniques.

In the first use case, Sensitivity Analysis provided a valuable perspective to understand the prediction of NOx emissions in Boston, revealing the influences of different variables and their intricate relationships in driving the prediction. The comparison with traditional XAI methods affirmed the superior capabilities of our method, enabling a comprehensive interpretation of the model's behavior.

The second use case focused on the prediction of parkinson disease progression measure. This high-dimensional problem posed a more challenging task for traditional XAI methods, which are often limited by feature independence assumptions and global explanation perspectives. In contrast, our methods successfully analyzed the model's inner processes, provided nuanced understanding of the local-level dependencies, and yielded explanations consistent with existing literature.

Lastly, in the third use case of predicting the RUL in the CMAPSS FD001 dataset, the interaction invariant demonstrated their effectiveness in detecting feature interactions, yielding more thorough and reliable insights than the other applied methods.

Overall, these use cases have underscored that while traditional XAI methods provide useful insights, the techniques developed in this thesis not only match their performance but also offer more profound explanations by detecting feature interactions and analyzing the relationship between inputs and outputs from both global and local perspectives. Thus, the contributions of this thesis enable a higher degree of understanding.

5 Conclusions

Success doesn't mean the absence of failures; it means the attainment of ultimate objectives. It means winning the war, not every battle.
Edwin Bliss (1912—2002)

This last chapter summarizes the developments of this dissertation. The main conclusions that can be drawn from the experiments carried out are set forth and the most original contributions are highlighted. Finally, the open issues that have not been tackled in the thesis, as well as possible future research lines, are discussed.

5.1. Summary and conclusions

The importance of the field of Explainable Artificial Intelligence (XAI) in recent years cannot be overstated. As we are becoming increasingly reliant on artificial intelligence and machine learning in various aspects of our lives, the need for transparency and interpretability has grown more crucial than ever. These complex models have the capacity to impact decision-making in a plethora of fields, from healthcare and environmental science to finance and policy. Yet, without the ability to understand and explain the reasoning behind their predictions, we risk encountering ethical issues, lack of trust, and misuse of these powerful tools. Thus, the need for XAI is driven not just by academic curiosity, but also by societal responsibility, ethical considerations, and regulatory requirements.

This thesis has taken a deep dive into the current state of the art in XAI. We explored key concepts in explainability and interpretability, the types of explanations that can be provided, and important considerations such as model-specific vs. model-agnostic approaches, and local vs. global explanations. Our discussion led us to present a comprehensive taxonomy of XAI, providing a guideline for selecting suitable XAI techniques based on specific requirements. We further delved into a range of commonly used XAI techniques, highlighting their strengths and limitations, and the contexts in which they are most appropriate.

The core of our contribution in this thesis is the development of XAI techniques based on partial derivatives, tailored for understanding and interpreting Multi-Layer Perceptron (MLP) models, but applicable to any model whose partial derivatives can be calculated. We have made significant strides in the analytical computation of first, second, and third partial derivatives for MLP models. The calculation of these derivatives is a critical step, as they represent the rate at which the MLP output changes concerning changes in the input variables. These advancements provide the groundwork for the three novel methods we introduce: Sensitivity Analysis, $\alpha-$curves, and the application of the Interaction Invariant designed in Alfaya *et al.* (2023) to MLP models.

Sensitivity Analysis, our first method, leverages partial derivatives to estimate the influence of input variables on the MLP output. This information is critical for model refinement and feature selection. $\alpha-$curves, our second method, provides a comprehensive view of sensitivity distributions across the input space, enabling the analysis of feature importance and sensitivity behavior at varying levels of detail. Lastly, the Interaction Invariant method helps identify interactions between input variables and their impact on the model output, offering insights into the more complex and non-linear relationships that an MLP model can learn.

To validate the performance and utility of our proposed methods, we conducted several case studies, each showcasing the application of Sensitivity Analysis, $\alpha-$curves, and Interaction Invariant in different domains. This included predicting NOx emissions in the Boston dataset, parkinson disease progression, and the Remaining Useful Life (RUL) of turbofan engines using the CMAPSS dataset. Across these diverse scenarios, our methods consistently provided nuanced, in-depth insights that went beyond traditional XAI techniques. These case studies highlighted the superior capabilities of our proposed methods to retrieve information from machine learning models, thereby offering a powerful tool for better model understanding.

While our proposed methods have shown superior capabilities in retrieving information from MLP models, it is important to note that they do not render other XAI methods obsolete. Each method, whether it's a part of `neuralsens` or a traditional XAI technique, carries its own set of advantages and disadvantages. The best method to use in a particular situation will depend on the specific requirements of the use case, the type of data, and the complexities of the model. It is essential that researchers and practitioners take these factors into account when deciding on the most suitable method to use for explainability purposes. The ultimate aim is to build a comprehensive understanding of the models we create, thereby fostering trust, promoting ethical decision-making, and driving more accurate, reliable predictions.

5.2. Original contributions

Firstly, a central and impactful contribution of this thesis is the establishment of detailed analytical calculations for the first, second, and third partial derivatives of a Multi-Layer Perceptron (MLP) model. Previous work such as Gevrey *et al.*, 2003 and Gevrey *et al.*, 2006 describe how to calculate the first partial derivatives and second partial derivatives for MLP, restraining the MLP architecture to a single hidden layer with sigmoid activation function and a single output neuron. Moreover, the description developed in previous work do not optimize the required computations leveraging matrix operations, which make the computation unfeasible for highly dimensional datasets. Capable of handling an arbitrary number of hidden layers, activation function and output variables; the approach developed in this thesis is not only versatile but also computationally efficient for any MLP architecture. By swiftly and efficiently calculating these partial derivatives, we acquire detailed insights into the MLP model's response to shifts in input variables. This improved computational performance paves the way for the implementation of the advanced XAI methods presented in this thesis.

Secondly, the development of the Sensitivity Analysis based on partial derivatives method represents another important contribution. This technique, based on partial derivatives, offers valuable perspectives on how the input variables influence the model output. Sensitivity Analysis is particularly beneficial for highlighting the most impactful variables and aiding in model refinement and feature selection. However, there is a potential risk of overlooking local non-linear effects.

The third contribution lies in the creation of the $\alpha-$curves method. This technique provides a comprehensive understanding of sensitivity distributions across the input space, providing a more detailed view compared to traditional sensitivity measures. By adjusting the alpha parameter, users can scrutinize sensitivity at different detail levels, facilitating the detection of both average and localized high-sensitivity regions.

Lastly, the application of the Interaction Invariant method in Machine Learning models marks our fourth contribution. This technique focuses on detecting interactions between input variables, enabling users to identify regions with interactions between a pair of input variables. Its resilience to transformations of the input or output space makes it a robust tool in interaction analysis. However, it's important to note that this method primarily captures pairwise interactions and may not fully account for higher-order interactions. Also, the size of the grid where the invariant is calculated could influence the results and their interpretation, indicating the need for careful grid design and mindful interpretation.

To ensure that the methods developed in this thesis reach the widest possible audience and have the most substantial impact, we have made them available to the public through two programming packages: `neuralsens` for Python and `NeuralSens` for R. This act of packaging and publicly distributing the methods is of paramount importance for several reasons. It empowers researchers and practitioners across disciplines to apply these methods without the hurdle of having to code them from the ground up. In providing these tools in Python and R, the most widely used languages for data analysis and machine learning, we ensure broad applicability and foster a diverse user base. Moreover, this public availability invites valuable peer scrutiny, potential improvements, and fosters a culture of open science and collaboration, thus accelerating advancements in Explainable Artificial Intelligence. Ultimately, it is our aspiration that these packages enable users to gain a deeper understanding of their machine learning models, further transparency, and aid in constructing more reliable and fair AI systems.

These methods are already being applied in the social science field. For example, Arroyo-Barrigüete *et al.*, 2023 examines the persistent gender gap in mathematics performance in Spain's education system. In this study, both a linear regression and an MLP model are trained on the collected data to determine if the variables related to mathematical achievement are significantly different between male and female students. The application of sensitivity analysis to the MLP model developed in this thesis unveils non-linear relationships between the output and certain input variables. Subsequently, the specification of the linear model is revised based on insights gained from the sensitivity analysis. This adjustment refines the model's representation, leading to the identification of previously undetected associations that contribute to a comprehensive understanding of the gender gap phenomenon. The seamless transition between the metrics proposed in the sensitivity analysis methods and the coefficients found in the linear regression model facilitated this insightful exploration, ultimately highlighting the practical applicability of the methods developed in this thesis in extracting meaningful insights from complex data.

In other study, Lumacad *et al.*, 2022 delves into the realm of online distance learning (ODL), a pivotal extension of the distance learning paradigm introduced in response to the challenges posed by the COVID-19 pandemic. Despite the benefits that online learning has introduced to the education landscape, it remains critical to identify the key factors that significantly influence learners' online academic performance. A MLP model is trained to predict the academic performance of junior and high school students in the Philippines, using the sensitivity analysis method to discover the most important factors in the academic performance during the online learning period in this region.

The research developed in the Explainable Artificial Intelligence field has been materialized in the following journal publications and seminars:

Journal publications

- J. D. Alfaya, Pizarroso-Gonzalo, J. Portela, and A. Muñoz, *Invariant Interaction Measure*, in edition.
- J. Pizarroso-Gonzalo, D. Alfaya, J. Portela, and A. Muñoz, *Metric Tools for Sensitivity Analysis with Applications to Neural Networks*, (submitted to Expert Systems with Applications, November 2023)
- J. Pizarroso-Gonzalo, J. Portela, and A. Muñoz, *Neuralsens: Sensitivity analysis of neural networks*, Journal of Statistical Software, vol. 102, no. 7, pp. 1–36, February 2022.

Seminars

- D. Alfaya, J. Pizarroso-Gonzalo, J. Portela, and A. Muñoz, *XAI methods based on partial derivatives*, GI^{2}DA seminar, ICADE law school, Comillas Pontifical University, 29 June 2023.
- J. L. Arroyo-Barrigüete, J. Portela, J. Pizarroso-Gonzalo, A. Muñoz, and D. Alfaya, *Utilización del paquete NeuralSens (redes neuronales interpretables)*, Afi School, 28 June 2023.
- J. L. Arroyo-Barrigüete, J. Portela, J. Pizarroso-Gonzalo, and A. Muñoz, *Utilización del paquete NeuralSens (redes neuronales interpretables)*, Faculty of Commerce and Tourism, Complutense University of Madrid, 2 March 2023.
- J. L. Arroyo-Barrigüete, J. Portela, J. Pizarroso-Gonzalo, and A. Muñoz, *Abriendo la caja negra de las redes neuronales: cálculo de sensibilidades con el paquete NeuralSens*, Foundation of the Autonomous University of Madrid, 23 February 2023.
- J. L. Arroyo-Barrigüete, J. Portela, J. Pizarroso-Gonzalo, and A. Muñoz, *Abriendo la caja negra de las redes neuronales: cálculo de sensibilidades con el paquete NeuralSens*, Faculty of Commerce and Tourism, Complutense University of Madrid, 10 November 2022.

5.3. Future work

This dissertation has advanced in the development of Explainable Artificial Intelligence (XAI) and its applications to neural network models, in particular, Multi-Layer Perceptrons. These contributions lead to a number of future lines of research that could be explored. This section summarizes some of them:

- Implementation of partial derivative calculation for other neural network architectures: Extending the developed analytical methods for calculating partial derivatives to other neural network architectures, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). This extension involves adapting the developed methods to the unique characteristics and inner workings of each of these models in order to gain relevant information adapted to the task involved.
- Development of a statistical significance test for input variables in ML models based on Sensitivity Analysis: Drawing inspiration from the statistical test designed by White and Racine, 2001, future work could include developing a similar test that determines when a variable is significant for predicting the output of a machine learning model. This would offer a more formal and statistically robust way to identify influential variables.
- Adapting α-curves methodology for discrete input/output variables: The α-curves methodology, which was initially developed for regression tasks, could be adapted to classification tasks by modifying the metric applied to the discrete input or output variable. This would enhance the applicability of this methodology across a broader spectrum of machine learning problems.
- Expansion of Interaction Invariant methodology to capture higher-order interactions: while the interaction invariant methodology is currently capable of capturing pairwise interactions, there is scope for extending it to detect higher-order interactions. This expansion would enable a more comprehensive understanding of complex relationships between variables in machine learning models.

A
Comparison with automatic differentiation

In this annex, a thorough presentation of the computational times across different configurations for the derivation of first, second, and third partial derivatives is provided. Enclosed are three distinct figures, each delineating the computational times for each order of derivative, rendering a comprehensive view of the time efficiency under varying scenarios. These computations were conducted on a system equipped with 32 GB of RAM, an Intel Core i5-6300HQ processor, and a NVIDIA GeForce GTX 950M graphics card.

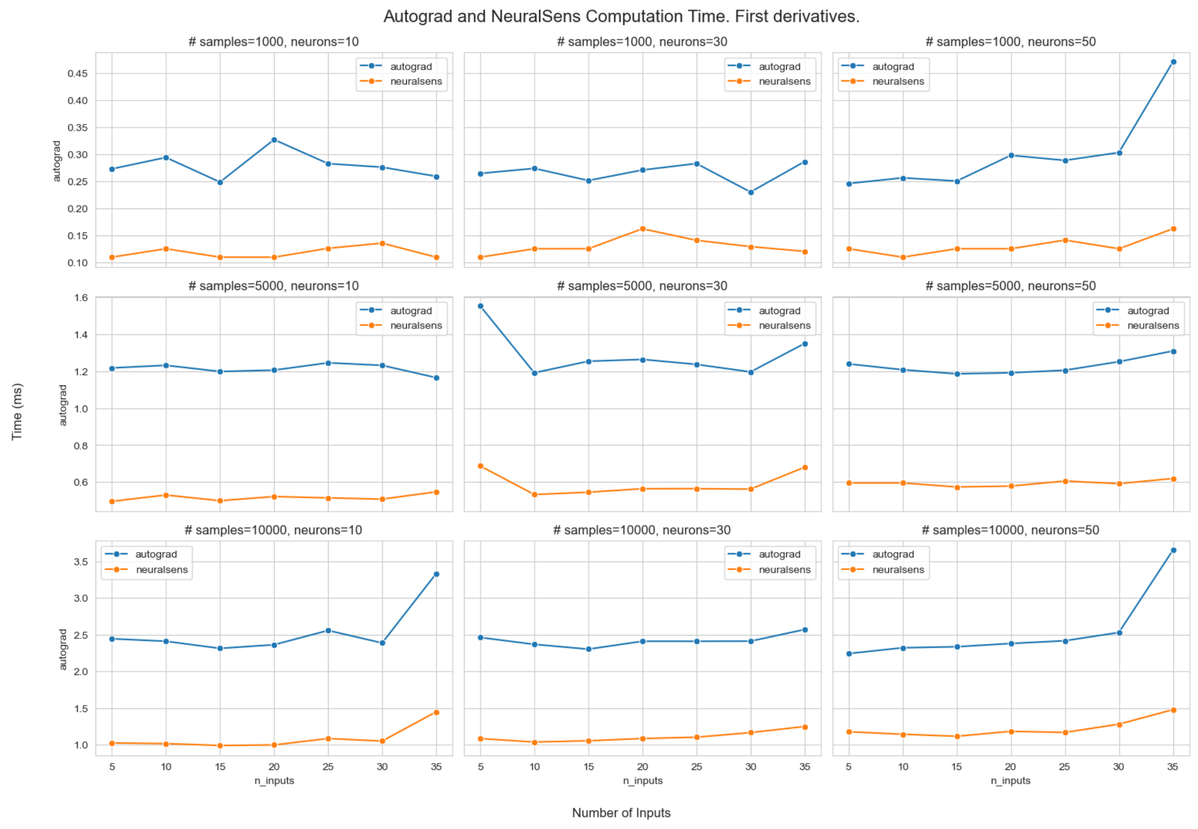


Figure A.1. Comparative analysis of computation times for the calculation of first partial derivatives utilizing analytical calculations (`neuralsens`) and automatic differentiation (`autograd`) amidst varying model complexities and sample sizes.

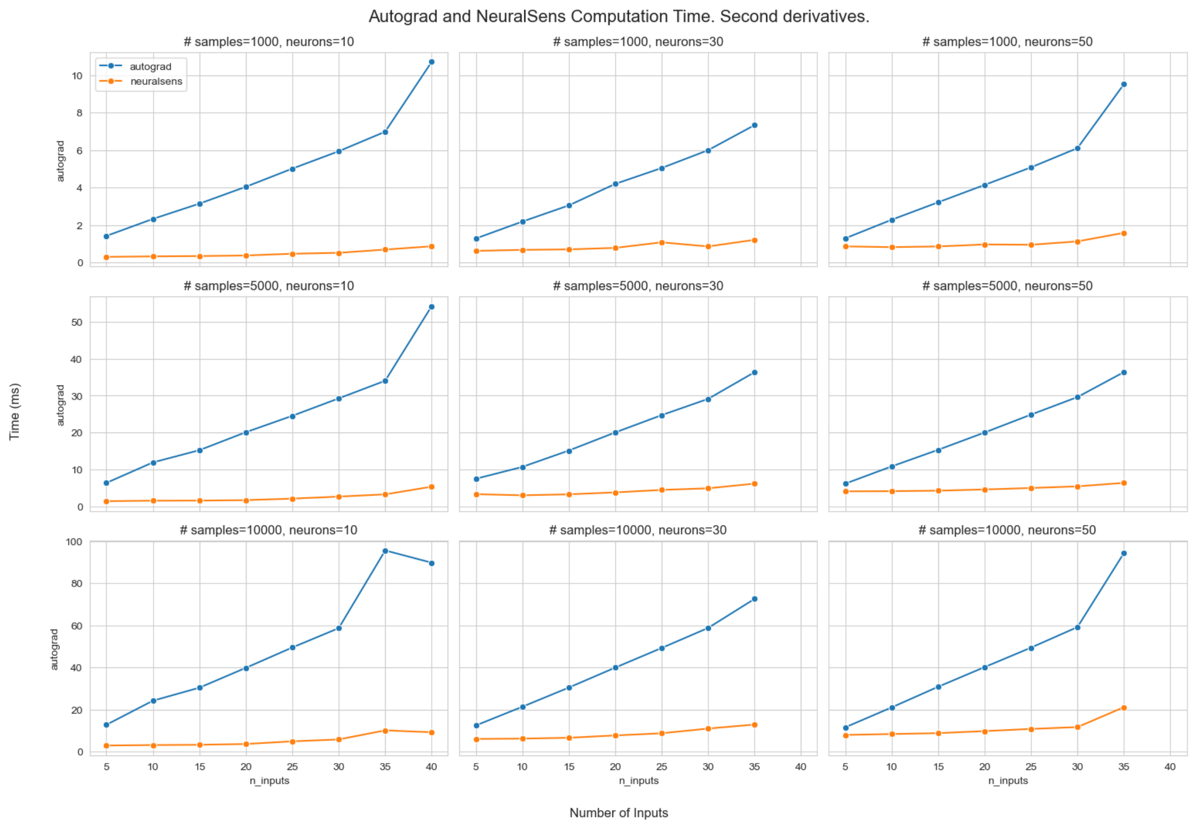


Figure A.2. Comparative analysis of computation times for the calculation of second partial derivatives utilizing analytical calculations (`neuralsens`) and automatic differentiation (`autograd`) amidst varying model complexities and sample sizes.

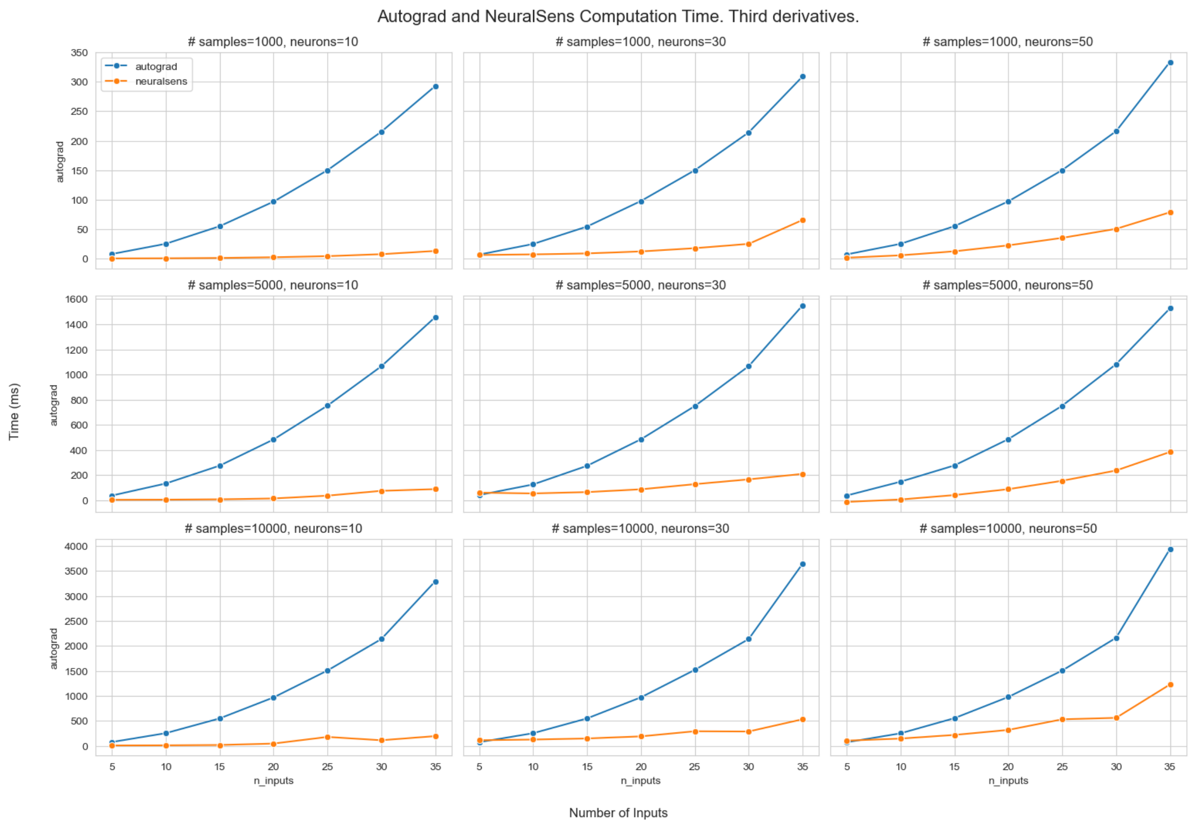


Figure A.3. Comparative analysis of computation times for the calculation of third partial derivatives utilizing analytical calculations (`neuralsens`) and automatic differentiation (`autograd`) amidst varying model complexities and sample sizes.

References

[AJ18] D. Alvarez Melis and T. Jaakkola, “Towards robust interpretability with self-explaining neural networks”, *Advances in neural information processing systems*, vol. 31, 2018.

[Aki+16] R. Akita, A. Yoshihara, T. Matsubara, and K. Uehara, “Deep learning for stock prediction using numerical and textual information”, pp. 1–6, 2016.

[Alf+23] D. Alfaya, J. Pizarroso, J. Portela, and A. Muñoz, “Invariant interaction measure”, *arXiv preprint*, 2023.

[Alt+10] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, “Permutation importance: A corrected feature importance measure”, *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

[AOS22] L. Arras, A. Osman, and W. Samek, “Clevr-xai: A benchmark dataset for the ground truth evaluation of neural network explanations”, *Information Fusion*, vol. 81, pp. 14–40, 2022.

[Arr+20] A. B. Arrieta *et al.*, “Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai”, *Information fusion*, vol. 58, pp. 82–115, 2020.

[Arr+23] J. L. Arroyo-Barrigüete, S. Carabias-López, F. Borrás-Pala, and G. Martın-Antón, “Gender differences in mathematics achievement: The case of a business school in spain”, *SAGE Open*, vol. 13, no. 2, p. 21 582 440 231 166 922, 2023.

[Aza+21] H. Azadi, M.-R. Akbarzadeh-T, A. Shoeibi, and H. R. Kobravi, “Evaluating the effect of parkinson’s disease on jitter and shimmer speech features”, *Advanced Biomedical Research*, vol. 10, 2021.

[Bac+15] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, “On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation”, in *PloS one*, Public Library of Science, vol. 10, 2015, e0130140.

[BCV13] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives”, *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[Bec18] M. W. Beck, “Neuralnettools: Visualization and analysis tools for neural networks”, *Journal of statistical software*, vol. 85, no. 11, p. 1, 2018.

[Ben12] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures”, in *Neural Networks: Tricks of the Trade: Second Edition*, Springer, 2012, pp. 437–478.

[Ber11] T. Bertin-Mahieux, *YearPredictionMSD*, UCI Machine Learning Repository, 2011.

[BN06] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.

[BNJ03] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation”, Jan, vol. 3, 2003, pp. 993–1022.

[Bor+22] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, “Deep neural networks and tabular data: A survey”, *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[Bre01] L. Breiman, “Random forests”, *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[Bro22] J. Brownlee, *When to use mlp, cnn, and rnn neural networks*, Aug. 2022. [Online]. Available: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/.

[BS14] R. Bro and A. K. Smilde, “Principal component analysis”, *Analytical methods*, vol. 6, no. 9, pp. 2812–2831, 2014.

[Car+15] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, “Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission”, in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1721–1730.

[Car+20] Z. Car, S. Baressi Šegota, N. Anđelić, I. Lorencin, V. Mrzljak, *et al.*, “Modeling the spread of covid-19 infection using a multilayer perceptron”, *Computational and mathematical methods in medicine*, vol. 2020, 2020.

[Chu+23] Y.-N. Chuang *et al.*, “Efficient xai techniques: A taxonomic survey”, *arXiv preprint arXiv:2302.03225*, 2023.

[CV95] C. Cortes and V. Vapnik, “Support-vector networks”, *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[Cyb89] G. Cybenko, “Approximation by superpositions of a sigmoidal function”, *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[DBL95] Y. Dimopoulos, P. Bourret, and S. Lek, “Use of some sensitivity criteria for choosing networks with good generalization ability”, *Neural Processing Letters*, vol. 2, pp. 1–4, 1995.

[Dev+18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding”, *arXiv preprint arXiv:1810.04805*, 2018.

[Dia16] N. Diakopoulos, “Accountability in algorithmic decision making”, *Communications of the ACM*, vol. 59, no. 2, pp. 56–62, 2016.

[Dim+99] I. Dimopoulos, J. Chronopoulos, A. Chronopoulou-Sereli, and S. Lek, “Neural network models to study relationships between lead concentration in grasses and permanent urban descriptors in athens city (greece)”, *Ecological modelling*, vol. 120, no. 2-3, pp. 157–165, 1999.

[DK17] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning”, *arXiv preprint arXiv:1702.08608*, 2017.

[DS21] M. Desai and M. Shah, “An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (mlp) and convolutional neural network (cnn)”, *Clinical eHealth*, vol. 4, pp. 1–11, 2021.

[Efr+04] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, “Least angle regression”, 2004.

[El +22] M. El Khatib, S. Hamidi, I. Al Ameeri, H. Al Zaabi, and R. Al Marqab, “Digital disruption and big data in healthcare-opportunities and challenges”, *ClinicoEconomics and Outcomes Research*, pp. 563–574, 2022.

[Est+17] A. Esteva *et al.*, “Dermatologist-level classification of skin cancer with deep neural networks”, *nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[FF14] J. Faber and L. M. Fonseca, “How sample size influences research outcomes”, *Dental press journal of orthodontics*, vol. 19, pp. 27–29, 2014.

[FP08] J. H. Friedman and B. E. Popescu, “Predictive learning via rule ensembles”, *The annals of applied statistics*, pp. 916–954, 2008.

[Fri01] J. H. Friedman, "Greedy function approximation: A gradient boosting machine", *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[GA19] D. Gunning and D. Aha, "Darpa's explainable artificial intelligence (xai) program", *AI magazine*, vol. 40, no. 2, pp. 44–58, 2019.

[Gar91] G. D. Garson, "Interpreting neural-network connection weights", *AI expert*, vol. 6, no. 7, pp. 46–51, 1991.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[GDL03] M. Gevrey, I. Dimopoulos, and S. Lek, "Review and comparison of methods to study the contribution of variables in artificial neural network models", *Ecological modelling*, vol. 160, no. 3, pp. 249–264, 2003.

[GDL06] M. Gevrey, I. Dimopoulos, and S. Lek, "Two-way interaction of input variables in the sensitivity analysis of neural network models", *Ecological modelling*, vol. 195, no. 1-2, pp. 43–50, 2006.

[GF18] P. Goyal and E. Ferrara, "Graph attention networks", in *International Conference on Learning Representations*, 2018.

[Gha+22] Y. Ghasemi, H. Jeong, S. H. Choi, K.-B. Park, and J. Y. Lee, "Deep learning-based object detection in augmented reality: A systematic review", *Computers in Industry*, vol. 139, p. 103 661, 2022.

[Gha15] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence", *Nature*, vol. 521, no. 7553, pp. 452–459, 2015.

[Gol+15] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation", *Journal of Computational and Graphical Statistics*, vol. 24, no. 1, pp. 44–65, 2015.

[Gun17] D. Gunning, "Explainable artificial intelligence (xai)", *Defense advanced research projects agency (DARPA), nd Web*, vol. 2, no. 2, p. 1, 2017.

[Hei+20] A. A. Heidari, H. Faris, S. Mirjalili, I. Aljarah, and M. Mafarja, "Ant lion optimizer: Theory, literature review, and application in multi-layer perceptron neural networks", *Nature-Inspired Optimizers: Theories, Literature Reviews and Applications*, pp. 23–46, 2020.

[HF16] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review", *International Journal of Forecasting*, vol. 32, no. 3, pp. 914–938, 2016.

[Hol16] A. Holzinger, "Interactive machine learning for health informatics: When do we need the human-in-the-loop?", *Brain Informatics*, vol. 3, no. 2, pp. 119–131, 2016.

[HT87] T. Hastie and R. Tibshirani, "Generalized additive models: Some applications", in 398, vol. 82, Taylor & Francis, 1987, pp. 371–386.

[IPP16] J. Irani, N. Pise, and M. Phatak, "Clustering techniques and the similarity measures used in clustering: A survey", *International journal of computer applications*, vol. 134, no. 7, pp. 9–14, 2016.

[Jac86] P. Jackson, "Introduction to expert systems", 1986.

[Jam+13] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[Kag16] Kaggle, "House prices: Advanced regression techniques", 2016. [Online]. Available: `https://www.kaggle.com/c/house-prices-advanced-regression-techniques`.

[KB21] M. Kastrati and M. Biba, "A state-of-the-art survey of advanced optimization methods in machine learning.", in *RTA-CSIT*, 2021, pp. 1–10.

[Kir+14] M. Kirchner, P. Schubert, M. Liebherr, and C. T. Haas, “Detrended fluctuation analysis and adaptive fractal analysis of stride time data in parkinson’s disease: Stitching together short gait trials”, *PloS one*, vol. 9, no. 1, e85787, 2014.

[Lah17] S. Lahmiri, “Parkinson’s disease detection based on dysphonia measurements”, *Physica A: Statistical Mechanics and its Applications*, vol. 471, pp. 98–105, 2017.

[LeC+98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[Lek+96] S. Lek, M. Delacoste, P. Baran, I. Dimopoulos, J. Lauga, and S. Aulagnier, “Application of neural networks to modelling nonlinear relationships in ecology”, *Ecological modelling*, vol. 90, no. 1, pp. 39–52, 1996.

[LGS13] K. Liu, N. Z. Gebraeel, and J. Shi, “A data-level fusion model for developing composite health indices for degradation modeling and prognostic analysis”, *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 652–664, 2013.

[Lip18] 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, pp. 31–57, 2018.

[Lit+08] M. Little, P. McSharry, E. Hunter, J. Spielman, and L. Ramig, “Suitability of dysphonia measurements for telemonitoring of parkinson’s disease”, *Nature Precedings*, pp. 1–1, 2008.

[Liu+18] J. Liu *et al.*, “Anomaly detection in manufacturing systems using structured neural networks”, pp. 175–180, 2018.

[LL17] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions”, vol. 30, 2017.

[LLA22] I. Lauriola, A. Lavelli, and F. Aiolli, “An introduction to deep learning in natural language processing: Models, techniques, and tools”, *Neurocomputing*, vol. 470, pp. 443–456, 2022.

[Lor+19a] I. Lorencin, N. Anđelić, V. Mrzljak, and Z. Car, “Genetic algorithm approach to design of multi-layer perceptron for combined cycle power plant electrical power output estimation”, *Energies*, vol. 12, no. 22, p. 4352, 2019.

[Lor+19b] I. Lorencin, N. Anđelić, V. Mrzljak, and Z. Car, “Multilayer perceptron approach to condition-based maintenance of marine codlag propulsion system components”, *Pomorstvo*, vol. 33, no. 2, pp. 181–190, 2019.

[Lum+22] G. S. Lumacad, J. V. C. Damasing, S. B. M. Tacastacas, and A. R. T. Quipanes, “Analyzing sensitive factors affecting online academic performance in the new normal: A machine learning perspective”, in *2022 XVII Latin American Conference on Learning Technologies (LACLO)*, IEEE, 2022, pp. 01–07.

[Lun+20] S. M. Lundberg, G. Erion, *et al.*, “From local explanations to global understanding with explainable ai for trees”, *Nature machine intelligence*, vol. 2, no. 1, pp. 56–67, 2020.

[Mac67] J. MacQueen, “Some methods for classification and analysis of multivariate observations”, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, 1967.

[Man+19] S. Maniraj, A. Saini, S. Ahmed, and S. Sarkar, “Credit card fraud detection using machine learning and data science”, 9, vol. 8, 2019, pp. 110–115.

[MAP06] V. Metsis, I. Androutsopoulos, and G. Paliouras, “Spam filtering with naive bayes-which naive bayes?”, in *CEAS*, Mountain View, CA, vol. 17, 2006, pp. 28–69.

[MBC18] C. Molnar, B. Bischl, and G. Casalicchio, “Iml: An r package for interpretable machine learning”, *JOSS*, vol. 3, no. 26, p. 786, 2018. [Online]. Available: `https://joss.theoj.org/papers/10.21105/joss.00786`.

[MC98] A. Muñoz and T. Czernichow, “Variable selection using feedforward and recurrent neural networks”, *Engineering Intelligent Systems for Electrical Engineering and Communications*, vol. 6, no. 2, pp. 91–102, 1998.

[McG+17] A. McGovern *et al.*, “Using artificial intelligence to improve real-time decision-making for high-impact weather”, *Bulletin of the American Meteorological Society*, vol. 98, no. 10, pp. 2073–2090, 2017.

[Mil19] T. Miller, “Explanation in artificial intelligence: Insights from the social sciences”, *Artificial intelligence*, vol. 267, pp. 1–38, 2019.

[Mio+17] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, “Deep learning for healthcare: Review, opportunities and challenges”, *Briefings in Bioinformatics*, vol. 19, no. 6, pp. 1236–1246, 2017.

[Mir+21] A. Mirestean *et al.*, “Powering the digital economy: Opportunities and risks of artificial intelligence in finance”, *Departmental Papers*, vol. 2021, no. 024, p. 1, 2021.

[Mir+22] J. A. Z. Miranda, J. E. C. Pillajo, F. L. G. Arévalo, and J. D. V. Sánchez, “Selección de funciones de voz mediante algoritmos genéticos para la detección de la enfermedad de parkinson”, *Revista de Investigación en Tecnologıas de la Información*, vol. 10, no. 21, pp. 140–150, 2022.

[Mit97] T. M. Mitchell, *Machine learning*. McGraw-Hill, 1997.

[MKK14] K. Maeda, H. Katagiri, and T. Kuremoto, “Comparison of optimization methods for deep neural networks”, in *2014 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2014, pp. 2509–2516.

[Mni+15] V. Mnih *et al.*, “Human-level control through deep reinforcement learning”, *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[Mol22] C. Molnar, *Interpretable Machine Learning, A Guide for Making Black Box Models Explainable*, 2nd ed. 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book.

[Mon+14] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, “On the number of linear regions of deep neural networks”, *Advances in neural information processing systems*, vol. 27, 2014.

[MPV21] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[MRW19] B. Mittelstadt, C. Russell, and S. Wachter, “Explaining explanations in ai”, pp. 279–288, 2019.

[MTY09] T. Minamisawa, K. Takakura, and T. Yamaguchi, “Detrended fluctuation analysis of temporal variation of the center of pressure (cop) during quiet standing in parkinsonian patients”, *Journal of Physical Therapy Science*, vol. 21, no. 3, pp. 287–292, 2009.

[Ngu+19] B. D. Nguyen, T.-T. Do, B. X. Nguyen, T. Do, E. Tjiputra, and Q. D. Tran, “Overcoming data limitation in medical visual question answering”, in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part IV 22*, Springer, 2019, pp. 522–530.

[Nie+21] I. E. Nielsen, D. Dera, G. Rasool, N. Bouaynaya, and R. P. Ramachandran, “Robust explainability: A tutorial on gradient-based attribution methods for deep neural networks”, *CoRR*, vol. abs/2107.11400, 2021. arXiv: 2107.11400. [Online]. Available: https://arxiv.org/abs/2107.11400.

[OJD02] J. D. Olden, M. K. Joy, and R. G. Death, “Illuminating the “black box”: A randomization approach for understanding variable contributions in artificial neural networks”, *Ecological modelling*, vol. 154, no. 1-2, pp. 135–150, 2002.

[OMS17] C. Olah, A. Mordvintsev, and L. Schubert, “Feature visualization”, *Distill*, vol. 2, no. 11, e7, 2017.

[Pan+17] T. N. Pandey, A. K. Jagadev, S. K. Mohapatra, and S. Dehuri, “Credit risk analysis using machine learning classifiers”, in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, IEEE, 2017, pp. 1850–1854.

[Pas+19] A. Paszke *et al.*, “Pytorch: An imperative style, high-performance deep learning library”, in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[Ped19] T. L. Pedersen, “Ggforce: Accelerating ‘ggplot2’”, *R package version 0.3*, vol. 1, p. 477, 2019.

[Piz+23] J. Pizarroso, D. Alfaya, J. Portela, and A. Muñoz, “Metric tools for sensitivity analysis with applications to neural networks”, *arXiv preprint arXiv:2305.02368*, 2023.

[Plu22] K. Pluciński, *Overview of explainable ai methods in nlp*, Mar. 2022. [Online]. Available: https://deepsense.ai/overview-of-explainable-ai-methods-in-nlp/.

[PPM22] J. Pizarroso-Gonzalo, J. Portela, and A. Muñoz, “Neuralsens: Sensitivity analysis of neural networks”, *Journal of Statistical Software*, vol. 102, no. 7, pp. 1–36, 2022.

[PT20] A. Pereira and C. Thomas, “Challenges of machine learning applied to safety-critical cyber-physical systems”, *Machine Learning and Knowledge Extraction*, vol. 2, no. 4, pp. 579–602, 2020.

[Qui86] J. R. Quinlan, “Induction of decision trees”, *Machine learning*, vol. 1, pp. 81–106, 1986.

[RGM+20] A. Rios, V. Gala, S. Mckeever, *et al.*, “Explaining deep learning models for structured data using layer-wise relevance propagation”, *arXiv preprint arXiv:2011.13429*, 2020.

[Rif+20] J. Riffi, M. A. Mahraz, A. El Yahyaouy, H. Tairi, *et al.*, “Credit card fraud detection based on multilayer perceptron and extreme learning machine architectures”, in *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, IEEE, 2020, pp. 1–5.

[Rip+11] B. Ripley *et al.*, “Mass: Support functions and datasets for venables and ripley’s mass”, *R package version*, vol. 7, pp. 3–29, 2011.

[RN09] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd. Prentice Hall Press, 2009.

[RRP23] T. Romero Arias, I. Redondo Cortés, and A. Pérez Del Olmo, “Biomechanical parameters of voice in parkinson’s disease patients”, *Folia phoniatrica et logopaedica: official organ of the International Association of Logopedics and Phoniatrics (IALP)*, vol. 10, p. 000 533 289, 2023.

[RSG16] 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*, 2016, pp. 1135–1144.

[RSG18] M. T. Ribeiro, S. Singh, and C. Guestrin, “Anchors: High-precision model-agnostic explanations”, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[Rud16] S. Ruder, “An overview of gradient descent optimization algorithms”, *arXiv preprint arXiv:1609.04747*, 2016.

[Sal+22] R. Saleem, B. Yuan, F. Kurugollu, A. Anjum, and L. Liu, “Explaining deep neural networks: A survey on the global interpretation methods”, *Neurocomputing*, 2022.

[Sam+21] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Müller, "Explaining deep neural networks and beyond: A review of methods and applications", *Proceedings of the IEEE*, vol. 109, no. 3, pp. 247–278, 2021.

[San96] A. M. San Roque, "Aplicación de técnicas de redes neuronales artificiales al diagnóstico de procesos industriales", Ph.D. dissertation, Universidad Pontificia Comillas, 1996.

[SAR21] M. Sahakyan, Z. Aung, and T. Rahwan, "Explainable artificial intelligence for tabular data: A survey", *IEEE access*, vol. 9, pp. 135 392–135 422, 2021.

[Sax+08] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation", in *2008 international conference on prognostics and health management*, IEEE, 2008, pp. 1–9.

[SB18] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[Sch+20] T. Schnake *et al.*, "Xai for graphs: Explaining graph neural network predictions by identifying relevant walks", *CoRR*, vol. abs/2006.03589, 2020. arXiv: 2006.03589. [Online]. Available: https://arxiv.org/abs/2006.03589.

[Sel+17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization", in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[Ser+22] O. Serradilla, E. Zugasti, J. Rodriguez, and U. Zurutuza, "Deep learning models for predictive maintenance: A survey, comparison, challenges and prospects", *Applied Intelligence*, vol. 52, no. 10, pp. 10 934–10 964, 2022.

[SGK17] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences", in *International conference on machine learning*, PMLR, 2017, pp. 3145–3153.

[Sha97] L. S. Shapley, "A value for n-person games", *Classics in game theory*, vol. 69, 1997.

[She18] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network", *CoRR*, vol. abs/1808.03314, 2018. arXiv: 1808.03314. [Online]. Available: http://arxiv.org/abs/1808.03314.

[Sib73] R. Sibson, "Slink: An optimally efficient algorithm for the single-link cluster method", *The Computer Journal*, vol. 16, no. 1, pp. 30–34, 1973.

[SK09] I. Sobol' and S. Kucherenko, "Derivative based global sensitivity measures and their link with global sensitivity indices", *Mathematics and Computers in Simulation*, vol. 79, no. 10, pp. 3009–3017, 2009.

[Spe22] T. Speith, "A review of taxonomies of explainable artificial intelligence (xai) methods", in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022, pp. 2239–2250.

[Sri+22] G. Srivastava *et al.*, *Xai for cybersecurity: State of the art, challenges, open issues and future directions*, 2022.

[Ste+21] I. Stepin, J. M. Alonso, A. Catala, and M. Pereira-Fariña, "A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence", *IEEE Access*, vol. 9, pp. 11 974–12 001, 2021.

[Sun19] R. Sun, *Optimization for deep learning: Theory and algorithms*, 2019.

[SVZ13] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps", *arXiv preprint arXiv:1312.6034*, 2013.

[TA22] A. Tsanas and S. Arora, “Data-driven subtyping of parkinson’s using acoustic analysis of sustained vowels and cluster analysis: Findings in the parkinson’s voice initiative study”, *SN Computer Science*, vol. 3, no. 3, p. 232, 2022.

[TL09] A. Tsanas and M. Little, *Parkinsons Telemonitoring*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C5ZS3N, 2009.

[TRL20] M. Tsang, S. Rambhatla, and Y. Liu, “How does this interaction affect me? interpretable attribution for feature interactions”, *Advances in neural information processing systems*, vol. 33, pp. 6147–6159, 2020.

[UC18] S. S. Upadhya and A. Cheeran, “Investigation of pitch and noise features extracted from voice samples of healthy and parkinson affected people using statistical tests”, *Journal of Engineering Science and Technology*, vol. 13, no. 9, pp. 2792–2804, 2018.

[Vas+17] A. Vaswani *et al.*, “Attention is all you need”, *Advances in neural information processing systems*, vol. 30, 2017.

[vdBK20] M. van den Berg and O. Kuiper, “Xai in the financial sector: A conceptual framework for explainable ai (xai)”, *https://www. hu. nl/-/media/hu/documenten/onderzoek/projecten/*, 2020.

[VEA22] D. Vale, A. El-Sharif, and M. Ali, “Explainable artificial intelligence (xai) post-hoc explainability methods: Risks and limitations in non-discrimination law”, *AI and Ethics*, pp. 1–12, 2022.

[VH08] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.”, *Journal of machine learning research*, vol. 9, no. 11, 2008.

[Vis20] G. Visani, *Explainable machine learning. xai review: Model agnostic tools*, Dec. 2020. [Online]. Available: https://towardsdatascience.com/explainable-machine-learning-9d1ca0547ae0.

[Viz+19] P. Vizza *et al.*, “Methodologies of speech analysis for neurodegenerative diseases evaluation”, *International journal of medical informatics*, vol. 122, pp. 45–54, 2019.

[Wal+22] S. Walia, K. Kumar, S. Agarwal, and H. Kim, “Using xai for deep learning-based image manipulation detection with shapley additive explanation”, *Symmetry*, vol. 14, no. 8, p. 1611, 2022.

[Wan+18] Y. Wang *et al.*, “A comparison of word embeddings for the biomedical natural language processing”, *Journal of biomedical informatics*, vol. 87, pp. 12–20, 2018.

[WFH16] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th. Morgan Kaufmann, 2016.

[WMF17] S. Wachter, B. Mittelstadt, and L. Floridi, “Why a right to explanation of automated decision-making does not exist in the general data protection regulation”, *International Data Privacy Law*, vol. 7, no. 2, pp. 76–99, 2017.

[WR01] H. White and J. Racine, “Statistical inference, the bootstrap, and neural-network modeling with application to foreign exchange rates”, *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 657–673, 2001.

[XLA22] Y. Xu, F. Li, and A. Asgari, “Prediction and optimization of heating and cooling loads in a residential building based on multi-layer perceptron neural network and different optimization algorithms”, *Energy*, vol. 240, p. 122 692, 2022.

[Yam+18] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: An overview and application in radiology”, *Insights into imaging*, vol. 9, pp. 611–629, 2018.

[YC10] I.-C. Yeh and W.-L. Cheng, “First and second order sensitivity analysis of mlp”, *Neurocomputing*, vol. 73, no. 10-12, pp. 2225–2233, 2010.

[Zen+18] X. Zeng, Z. Zhen, J. He, and L. Han, “A feature selection approach based on sensitivity of rbfnns”, *Neurocomputing*, vol. 275, pp. 2200–2208, 2018.

[ZF14] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks”, *European conference on computer vision*, pp. 818–833, 2014.

[ZMC94] J. M. Zurada, A. Malinowski, and I. Cloete, “Sensitivity analysis for minimization of input data dimension for feedforward neural network”, in *Proceedings of IEEE International Symposium on Circuits and Systems-ISCAS’94*, IEEE, vol. 6, 1994, pp. 447–450.

[ZYL22] T. Zhang, F. Yin, and Z.-Q. Luo, “Fast generic interaction detection for model interpretability and compression”, in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=fQTlgI2qZqE.