JOAQUÍN EDUARDO FIGUEROA BARRAZA

Frameworks for Interpretability of Deep Learning-Based Prognostics and Health Management

São Paulo
2023

JOAQUÍN EDUARDO FIGUEROA BARRAZA

Frameworks for Interpretability of Deep Learning-Based Prognostics and Health Management

A Doctoral Thesis submitted in partial fulfillment of the requirements for the degree of DOCTOR OF SCIENCE for the Polytechnic School of University of Sao Paulo, Brazil.

São Paulo
2023

JOAQUÍN EDUARDO FIGUEROA BARRAZA

Frameworks for Interpretability of Deep Learning-Based Prognostics and Health Management

A Doctoral Thesis submitted in partial fulfillment of the requirements for the degree of DOCTOR OF SCIENCE for the Polytechnic School of University of Sao Paulo, Brazil.

Concentration Area:
Naval Architecture and Ocean Engineering

Advisor:
Dr. Prof. Marcelo Ramos Martins

São Paulo
2023

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 19 de Janeiro de 2023

Assinatura do autor: JOAQUÍN

Assinatura do orientador:

Catalogação-na-publicação

Barraza, Joaquín Eduardo Figueroa
Frameworks for interpretability of deep learning-based prognostics and health management / J. E. F. Barraza -- versão corr. -- São Paulo, 2023.
187 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Naval e Oceânica.

1.interpretabilidade 2.redes neurais 3.prognóstico e gestão da saúde 4.seleção de variáveis 5.contrafactual I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Naval e Oceânica II.t.

Agradecimentos

Gostaria de agradecer ao Professor Marcelo Ramos Martins pela oportunidade de ser seu orientando e de fazer parte do LabRisco. Agradeço pela compreensão, paciência e sabedoria. Sua forma de fazer pesquisa foi um grande ensinamento para mim e levarei para a vida.

Ao Professor Enrique López Droguett pelo apoio durante a minha formação acadêmica, tanto na graduação como no mestrado e doutorado. Obrigado por ter confiado em mim e dado a oportunidade de entrar nesta área.

Aos meus colegas do LabRisco. Suas qualidades como pessoas e profissionais contribuíram ao meu desenvolvimento, tanto pessoal quanto profissional. Particularmente gostaria de agradecer à Valentina e ao Danilo.

Ao meu amigo Luis Felipe, pelas longas conversas, as risadas e o apoio nos momentos difíceis. Ter vindo com você para o Brasil foi uma das maiores e melhores experiencias da minha vida.

Aos meus amigos que mesmo na distância, foram um grande apoio neste processo. Particularmente, gostaria de agradecer ao Sergio, pelos conselhos, pelas conversas e pelo apoio, fazendo com que a distância fosse muito menor.

Aos meus pais e irmãos pelo amor incondicional e apoio infinito. Devo a eles tudo o que eu sou. Também gostaria de agradecer aos meus padrinhos pela proximidade e preocupação.

Finalmente, à Barbara, pelo amor, carinho, paciência, apoio, pelas palavras de ânimo nos momentos difíceis, por acreditar sempre em mim, mesmo quando eu não acredito. Não poderia ter tido uma melhor parceira neste processo e na vida toda.

Sou eternamente grato a quem colocou todas essas grandes pessoas no meu caminho.

Gostaria de agradecer o apoio da Agencia Nacional de Investigación y Desarrollo (ANID – Doctorado Becas Chile - 72200158).

Abstract

In the last five years, the use of deep learning algorithms for prognostics and health management (PHM) has led to a performance increase in fault diagnostics, prognostics, and anomaly detection. However, the lack of explanation and interpretability of these models results in a resistance towards their credibility and deployment. This means that even though deep learning-based models may achieve great performance, the understanding and explanation of how a deep learning-based PHM model obtains its results is still an open area of research. In this thesis, three techniques for interpretability of deep learning models in the context of prognostics and health management are proposed. The first one is comprised of a technique for feature selection and a methodology for quantitative evaluation of the technique's performance and comparison with other techniques. The proposed technique consists of a hidden layer next to the input layer whose weights determine the importance of each feature within the model. These weights are trained jointly with the rest of the network. The layer is referred to as *feature selection* (FS) layer. Moreover, the methodology for evaluation proposes the use of a novel metric referred to as *ranking quality score* (RQS). For the second framework, a multi-task neural network, referred to as Sparse Counterfactual Generation Neural Network (SCF-Net), is proposed for simultaneous fault diagnosis and counterfactual generation. Thus, the network has the ability to diagnose health states and deliver information referring to the minimal changes in the input values that lead to a change in the predicted health state by the model. In the third framework, the two previous approaches are combined in a network architecture referred to as Feature Selection and Sparse Counterfactual Generation network (FS-SCF). Also, a methodology is proposed for calculation of causality-based values for each feature, such as necessity, sufficiency, (necessity or sufficiency) and (necessity and sufficiency). This is used to further analyze the model and to interpret the results obtained from the FS layer. For these three frameworks, several case studies are used for testing, and compared to other existing techniques. Results across the three frameworks show a successful increase in interpretability while keeping task performance at the same level. Thus, the accuracy/interpretability tradeoff is successfully addressed in this thesis. Future lines of research include testing in other kinds of neural networks, such as convolutional neural networks, recurrent neural networks, and transformers. In the case of counterfactual-based approaches, future works include their adaption for regression tasks, due to the fact that they are limited to classification. This could

increase the types of applications in PHM. For example, remaining useful life (RUL) prediction.

Resumo

Nos últimos cinco anos, o uso de algoritmos de aprendizagem profunda para prognóstico e gestão de saúde (PHM) levou a um aumento de desempenho em diagnóstico de falhas, prognóstico e detecção de anomalias. No entanto, a falta de explicação e interpretabilidade desses modelos resulta em uma baixa credibilidade e uma resistência para sua implementação em aplicações da indústria. Isso significa que, embora os modelos baseados em aprendizagem profunda possam alcançar um ótimo desempenho, a compreensão e explicação de como um modelo de PHM baseado em aprendizagem profunda obtém seus resultados ainda é uma área aberta de pesquisa. Nesta tese, são propostos três *frameworks* de interpretabilidade de modelos de aprendizagem profunda no contexto de prognóstico e gestão em saúde. O primeiro é composto por uma técnica de seleção de variáveis e uma metodologia para avaliação quantitativa do desempenho da técnica e comparação com outras técnicas. A técnica proposta consiste em uma camada oculta próxima à camada de entrada cujos pesos determinam a importância de cada recurso dentro do modelo. Esses pesos são treinados em conjunto com o restante da rede. A camada é chamada de *feature selection layer* (FS). Além disso, a metodologia de avaliação propõe o uso de uma nova métrica denominada *ranking quality score* (RQS). Para o segundo *framework*, uma rede neural multitarefa, denominada Sparse Counterfactual Generation Neural Network (SCF-Net), é proposta para diagnóstico de falhas e geração de *counterfactuals* simultaneamente. Assim, a rede tem a capacidade de diagnosticar estados de saúde e entregar informações referentes às mudanças mínimas nos valores de entrada que levam a uma mudança no estado de saúde previsto pelo modelo. No terceiro *framework*, as duas abordagens anteriores são combinadas em uma arquitetura de rede chamada de Feature Selection e Sparse Counterfactual Generation network (FS-SCF). Além disso, é proposta uma metodologia para cálculo de valores baseados em causalidade para cada variável, tais como necessidade, suficiência, (necessidade ou suficiência) e (necessidade e suficiência). Isto é usado para analisar melhor o modelo e interpretar os resultados obtidos da camada FS. Para esses três *frameworks*, vários estudos de caso são usados para teste e comparados com outras técnicas existentes. Os resultados nos três *frameworks* mostram um aumento bem-sucedido na interpretabilidade, mantendo o desempenho da tarefa no mesmo nível. Assim, o *tradeoff* entre a precisão e a interpretabilidade é abordado com sucesso nesta tese. As futuras linhas de pesquisa incluem testes em outros tipos de redes neurais, como redes neurais convolucionais, redes neurais

recorrentes e redes *transformers*. No caso de abordagens baseadas em *counterfactuals*, trabalhos futuros incluem sua adaptação para tarefas de regressão, pelo fato de, por ora, estarem limitadas à classificação. Isso poderia aumentar os tipos de aplicativos em PHM, como por exemplo, para previsão de vida útil restante (RUL).

List of figures

List of tables

Abbreviations

AFS	Attention-based feature selection
ANN	Artificial neural network
CBM	Condition-based maintenance
CF	Counterfactual
CM	Corrective maintenance
C-MAPSS	Commercial Modular Aero-Propulsion System Simulation
CNN	Convolutional neural network
CWR	Case Western Reserve University Bearing Data Center
DC	Desired class
DL	Deep learning
DNN	Deep neural network
FS	Feature selection
FS-SCF	Feature Selection Sparse Counterfactual Generation Network
IoT	Internet of things
LIME	Local interpretable model-agnostic explanation
LSTM	Long short-term memory network
MES	Minimal explanation set
MI	Mutual information
ML	Machine learning
MSE	Mean squared error
MTL	Multi-task learning
NaS	Necessity and sufficiency
NGTP	Offshore Natural Gas Treatment Plant
NoS	Necessity or sufficiency
O&G	Oil and gas
PdM	Predictive maintenance
PHM	Prognostics and health management
PM	Performance metric
PrM	Preventive maintenance
RBF	Relation-between-features
RF	Random forest

RLV	Random latent variable
RNN	Recurrent neural network
RQS	Ranking quality score
RUL	Remaining useful life
SCF-Net	Sparse Counterfactual Generation Network
SHAP	Shapley additive explanations
SVM	Support vector machine

Contents

1. INTRODUCTION

1.1. Contextualization

The current development of the fourth industrial revolution, also referred to as Industry 4.0 (LASI et al., 2014), aims to the automation of manufacturing and industrial processes with the use of Big Data, Internet of Things (IoT) and smart sensors, among other tools. One expected impact of this process relates to maintenance. With the improvement of self-monitoring and techniques able to extract relevant information from large amounts of data being collected online, machines are able to diagnose current faults and/or predict failures in the future. Thus, maintenance actions are programmed dynamically according to the evolution of the asset's health state. This determines the evolution from preventive maintenance policies to predictive maintenance policies.

Among various industry sectors, the oil and gas (O&G) industry is also affected by Industry 4.0 and predictive maintenance. It is considered a capital-intensive industry and one where an accident could threaten human lives and even lead to environmental disasters. As such, its operations require high availability and safety standards. To do so, O&G facilities (e.g. FPSOs, drilling ships, refineries) seek to improve the effectiveness of their operations and maintenance policies to reduce downtimes and increase reliability. Corrective and preventive maintenance policies have been widely used despite the fact of being suboptimal in terms of costs and safety. In the case of the former, unplanned maintenances lead to unscheduled repairs, which could be very time consuming and expensive. Also, it does not prevent failures from occurring, which increases risks greatly. In turn, preventive maintenance policies do not consider variables that could change in time (such as environmental conditions or operational settings), requiring a larger or smaller time window between scheduled maintenances. Both kinds of policies have issues that reduce equipment's availability. In this sense, predictive maintenance aims to address the previously mentioned issues in order to increase safety while reducing maintenance costs and downtimes. As the fourth industrial revolution is occurring, the use of predictive maintenance in companies is becoming a reality. According to (MURRAY, 2019), important companies in the O&G industry such as Shell, Chevron, ExxonMobil, Equinor, Repsol, among others, are using predictive maintenance to improve performance and reduce costs. Indeed, from 2017 to 2019,

Repsol reduced corrective maintenance activities in a 15% and saved $200 million annually in operational expenses.

To achieve a state in which machines are able to collect and analyze large amounts of online sensor data to diagnose its health state and to predict failures, machine learning (ML) and deep learning (DL) techniques have gained popularity among researchers. They are suitable kinds of techniques as they can handle large amounts of data and, at the same time, reach high levels of performance. However, despite existing cases of companies adopting predictive maintenance policies, the increasing popularity in research does not reflect in the industry to the same level. This is mainly because companies are hesitant to deploy models whose results cannot be understood, explained, nor interpreted. Indeed, it is often mentioned in the literature there is a trade-off between performance and interpretability. This means that models that present the highest performance are less interpretable than those presenting low performance. Thus, interpretability for complex ML and DL models (such as neural networks) without performance loss is a necessity whose achievement could lead to more trust from companies in ML and DL, and consequently, more presence in the industry. This is a crucial step towards Industry 4.0. Ultimately, interpretability of ML and DL models and their inner dynamics could lead to research improvements, as they could give information that could help researchers have better understanding about physics of failure.

1.2. Objectives

This thesis' main objective is **to develop frameworks for interpretability of deep learning models for prognostics and health management (PHM).** This objective is divided in three contributions:

- A framework for feature selection of deep neural networks (DNN) in PHM for interpretability enhancement. This also includes a proposed metric for quantitative evaluation of feature selection techniques.
- A multi-task inherently interpretable neural network for simultaneous fault diagnosis and counterfactual generation.
- An interpretable neural network for fault diagnosis with embedded feature selection and counterfactual generation. This includes a methodology for

necessity and sufficiency quantification, creation of causality-based feature rankings and comparison between these and feature selection-based rankings.

To achieve the proposed objective, the proposed frameworks are evaluated using different case studies. This determines its feasibility, applicability and flexibility.

To evaluate the proposed techniques' performance, they must be compared to other approaches. This includes more classical techniques as well as state-of-the-art techniques.

1.3. Literature Review

During the last decade, Deep Learning (DL) algorithms have gained great popularity in various areas. Though their theoretical foundations were developed mostly between the 1940s and the 1970s (IVAKHNENKO, 1971; IVAKHNENKO; LAPA, 1967; MCCULLOCH; PITTS, 1943; MINSKY; PAPERT, 1969; ROSENBLATT, 1958; WERBOS; JOHN, 1974), technological limitations presented in (MINSKY; PAPERT, 1969) hindered their progress and research went into a hiatus. Nowadays, areas such as natural language processing (GOLDBERG, 2017), healthcare (ESTEVA et al., 2017), computer vision (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), autonomous driving (DREOSSI et al., 2017), among others, have embraced DL-based models in order to reach better performance. In prognostics and health management (PHM), DL models have been used for diagnostics (BARRAZA et al., 2020; CHEN et al., 2017; COFRE-MARTEL et al., 2019; GAN; WANG; ZHU, 2016; SAN MARTIN et al., 2019; VERSTRAETE et al., 2017; YU et al., 2019), prognostics (ARIA et al., 2020; BEN ALI et al., 2015; FIGUEROA BARRAZA et al., 2020; RUIZ-TAGLE PALAZUELOS; DROGUETT; PASCUAL, 2020; VERSTRAETE; DROGUETT; MODARRES, 2020; YUAN; WU; LIN, 2016; ZHANG et al., 2018) and anomaly detection (PARK et al., 2019; REDDY et al., 2016), showing promising results as well. In (VERSTRAETE et al., 2017), Verstraete et al. use convolutional neural networks (CNN) for fault diagnosis in rolling element bearings, feeding the network with image representations of the raw vibration signal, thus, leaving the feature extraction process to the network. In (BARRAZA et al., 2020), authors use a recently developed DL architecture called capsule networks to identify and quantify structural damage using transmissibility measures converted to images. They achieve better performance than neural networks and CNNs, also achieving better generalization. In (ZHANG et al., 2018), Zhang

et al. use long short-term memory networks (LSTM) to calculate the remaining useful life (RUL) of turbofan engines, taking advantage of the benefits of LSTM when working with time series.

Despite DL showing promising results, there is hesitance to adopt these models due to their black-box nature. Their high level of complexity impedes a proper interpretation of the results, which is an important issue in many areas such as safety critical applications. None of the PHM related works mentioned above describe tools for interpretability of the model's being utilized nor the results obtained. In Europe, the General Data Protection Regulation (GDPR), taking effect as of 2018, includes a *right to explanation* regarding algorithmic decision-making. This means that, in a situation where a decision that significantly affects someone is made based on an algorithm, that person has the right to know how and why that decision has been made. A clear example of this is loan application. Organizations are struggling to comply with this regulation (LI; YU; HE, 2019) because there are only few algorithms whose results can be interpreted. Unfortunately, these less complex algorithms do not show results as promising as DL-based models. Furthermore, there is no consensus on what an appropriate explanation is. To address this last issue, researchers have proposed different taxonomies to organize concepts and techniques (CARVALHO; PEREIRA; CARDOSO, 2019; DOSHI-VELEZ; KIM, 2017; FAN; XIONG; WANG, 2020). Most recently, Fan et. al. (FAN; XIONG; WANG, 2020) propose a taxonomy in which techniques are primarily divided according to when are they applied, with respect to the training process. They identify techniques that are applied after training using external tools (referred to as *post-hoc* analysis), and those that modify the inner dynamics of the model for it to be more easily understood (referred to as *ad-hoc* modeling). Among post-hoc techniques, they identify approaches such as feature analysis, model inspection, saliency maps, proxy models, mathematical and/or physical analysis, and explanations by case and by text. In turn, ad-hoc techniques are classified into interpretable representation through regularizations and model renovations through interpretable parameters. This taxonomy serves as a first step towards a concrete definition of interpretability and how to measure it. However, these kinds of analysis must be in line with development of new techniques for interpretability. In the context of PHM, interpretability is necessary to build trustworthy models. For a company to rely on a DL model to diagnose and/or predict a physical asset's health states, it is natural for people to demand not only a satisfactory level of performance, but also an understanding of how the model works. In (REZAEIANJOUYBARI; SHANG, 2020), Rezaeianjouybari and Shang do

an extensive review of DL in PHM, stating there is a challenge for researchers in the area of developing techniques for model interpretation, in order to overcome the unwillingness of some companies to adopt DL.

According to the authors in (CARVALHO; PEREIRA; CARDOSO, 2019), model interpretability frameworks can be divided into local and global interpretability. While local interpretability aims to explain single predictions, global interpretability aims to give an interpretation of how the model features interact to make predictions. Two of the most accepted frameworks for local interpretability that can be used in DL models are the Local Interpretable Model-agnostic Explanation (LIME) (RIBEIRO; GUESTRIN, 2016) and Shapley Additive Explanations (SHAP) (LUNDBERG; LEE, 2017) algorithms. LIME attempts to explain a single prediction by approximating the model locally using perturbations of the corresponding datapoint. SHAP adapts the Shapley values developed for game theory to determine what is the contribution of each feature in a single prediction. Both techniques are model agnostic, meaning they can be used for any model, including neural networks. However, according to the taxonomy presented in (FAN; XIONG; WANG, 2020), they are post-hoc techniques, meaning they require a trained model for analysis, and rely on input perturbations. This makes them prone to output misleading results in certain situations, such as adversarial attacks, as is discussed in (SLACK; HILGARD; JIA, 2020). Since they are algorithms based on perturbations, results can be negatively affected if these perturbations come from a source different than the phenomenon trying to be analyzed (for example, degradation). Also, the analysis done by these two techniques does not consider the inner dynamics of the model and how it treats different features.

Within interpretability techniques, feature selection is a relevant aspect, as it indicates how much participation each variable has within the model. When done before training, feature importance calculation is used to select the most relevant features to feed the model. Also, it helps detecting irrelevant features, which reduces overfitting and may lead to an improvement in performance. Furthermore, a model becomes easier to comprehend when it has less variables. There are numerous methods for calculating feature importance, which, according to the literature (JOVIĆ; BRKIĆ; BOGUNOVIĆ, 2015) can be divided into three categories: wrapper, filter, and embedded methods. Wrapper methods are those where the model is trained with different combinations of input features to determine which gives the best results. These methods clearly lack scalability due to time consumption (BLUM; LANGLEY, 1997). Filter methods use statistical metrics before training to determine feature

importance. Examples of these metrics are Pearson correlation coefficient, $\chi 2$ test, and mutual information. These kinds of models do not relate to the model or its predictions after training. Also, they only analyze the dependence of each feature with the output individually, not considering the interactions among features (FERREIRA; FIGUEIREDO, 2012). Embedded methods refer to algorithms with built-in techniques for determining feature importance. These include Random Forests (RF), Lasso Regression and Ridge Regression, among others (HAMEED et al., 2018; MALDONADO; LÓPEZ, 2018; SAHA; SARKAR; MITRA, 2009). The drawback of this kind of techniques is that they cannot be detached from the algorithm. Also, neural networks do not have an intrinsic feature selection technique. Thus, an embedded technique cannot be used in a neural network without training more than one model.

Despite the fact of neural networks not having an embedded technique for calculating feature importance, wrapper and filter methods can still be used. However, the objective of DL models is to use the least data preprocessing possible. Neural networks have an important number of hyperparameters to be tuned which turn the search for an appropriate model into a slow and intricate process. A data preprocessing stage makes this process even slower. Also, the use of wrapper or filter methods on DL models would amplify their aforementioned drawbacks. To address these issues, researchers have studied different ways to adapt DL models in order to determine feature importance (CHANG; RAMPASEK; GOLDENBERG, 2017; FENG et al., 2017; GUI; GE; HU, 2019a; HELLEPUTTE; DUPONT, 2009; MBUVHA; BOULKAIBET; MARWALA, 2019; NEZHAD et al., 2017; ROY; MURTY; MOHAN, 2015; ŠKRLJ et al., 2020; ZOU et al., 2015). In (CHANG; RAMPASEK; GOLDENBERG, 2017), Chang et al. use variational dropout in the input layer as a means to determine feature importance. In their approach, the individual dropout rate for each feature indicates how much the model is allowed to remove that feature. Thus, features with low dropout rate are more relevant than those with a high dropout rate. These values are used to build a ranking of features. They test their approach in two simulated datasets and six real-world datasets from diverse areas (none of them related to PHM), and achieve results similar to those using Random Forest for ranking features. In (GUI; GE; HU, 2019a), authors propose an approach based on attention mechanisms. It consists of a feature weights generation module (also called “attention module”) made of parallel hidden layers incorporated to the neural network next to the input layer. Each part of the module outputs a value between 0 and 1 which is then multiplied by its corresponding feature to enter the rest

of the network, which they call “learning module”. They test their approach using the MNIST dataset (LECUN; CORTES; BURGES, 1998), the noisy-MNIST dataset with its three variants (BASU et al., 2015) and two other small datasets used for feature selection problems (LI et al., 2016). Results show their framework achieves better results when compared with other filter and embedded methods. To the best of our knowledge, there are no established protocols to compare techniques for feature selection. This results in techniques usually being compared through unclear criteria, such as visual analysis (CAI et al., 2018; CHANG; RAMPASEK; GOLDENBERG, 2017; NARDONE; CIARAMELLA; STAIANO, 2019; ŠKRLJ et al., 2020). This can lead to incorrect results.

Besides feature selection techniques (BARRAZA; DROGUETT; MARTINS, 2021), there are several other kinds of interpretability and explainability techniques for deep learning models, including surrogate models (LUNDBERG; LEE, 2017; RIBEIRO; GUESTRIN, 2016), saliency methods (SUNDARARAJAN; TALY; YAN, 2017), explanation by text (DONG et al., 2017), and model interpretation through counterfactuals. Specifically, the idea of counterfactuals (CF) refers to alternative scenarios that may have changed the actual course of outcomes. Brought to the field of machine learning by Watcher et al. (WACHTER; MITTELSTADT; RUSSELL, 2017), counterfactuals are altered input values that generate a change in the model’s output class. Counterfactuals are used for interpretability of predictions as they give information on what changes are needed to the input feature values to change the prediction’s class. Furthermore, they are related to causality (MORGAN; WINSHIP, 2015; PEARL, 2018; PEARL; MACKENZIE, 2018). Judea Pearl suggests a division of causation into three hierarchic levels: association, intervention, and counterfactuals (PEARL; MACKENZIE, 2018). Association refers to the analysis of each variable in a system individually and how they correlate to other variables within the same system. Intervention is the exploration of the system through observation of the effects of intervening in the system. Finally, counterfactual reasoning consists in the process of imagining different outcomes given different interventions. Through counterfactual reasoning, humans identify causal relations by determining if a variation in one of the inputs would have changed the current outcome of an experiment or observation. Among the desirable properties of counterfactuals (which are described in detail in the following section), sparsity is particularly important in the interaction the model has with the end-user. A counterfactual generated with the least number of features being modified will

more likely be understood by the user, considering that humans have limited capacity to process more than three variables simultaneously (HALFORD et al., 2005).

Recently, counterfactuals have been used to inject interpretability into deep learning models (GUO; NGUYEN; YADAV, 2021; LIU et al., 2019; MAKHZANI et al., 2015; NEMIROVSKY et al., 2020; SAUER; GEIGER, 2021; YANG et al., 2021). In (NEMIROVSKY et al., 2020), authors use a generative adversarial network (GAN) to generate counterfactuals through residuals. Instead of feeding the generator network with random noise, they feed it with training data samples. Those samples are then modified through the obtained residuals. These modified samples, together with unaltered samples, are fed to a discriminator which determines if the input image is real or modified, and to a classifier, which determines the class of the input value. In (SAUER; GEIGER, 2021), authors propose the use a conditional GAN (cGAN) (MIRZA; OSINDERO, 2014) to generate counterfactuals. After training, they feed the network with different class labels to generate counterfactuals of such classes. GANs are particularly useful for generating counterfactuals, as they were designed to generate new instances as close to the original data distribution as possible. This is a requirement of counterfactuals. However, other approaches may be used as well. In (GUO; NGUYEN; YADAV, 2021), authors propose a framework for simultaneous predictive modeling and counterfactual generation using an encoder, a predictor and a counterfactual generator in the same network. Without using GANs, they are able to generate valid counterfactuals comparable to other techniques. However, they lack the possibility given by GANs of generating several different counterfactuals for the same input value, which is a desirable property for counterfactual generation techniques (MOTHILAL; SHARMA; TAN, 2020).

Counterfactuals are particularly useful in the context of prognostics and health management. With the information of how the input values must be modified to change from a healthy operation state to a failure state, one can get more information on how degradation makes a component or system to develop failures. On the contrary, if the model diagnoses a failure, a counterfactual could give valuable information on maintenance actions to be taken in order for the component or system to be operational again by indicating what feature should be altered in order for the diagnosed state to change from failure to healthy operation. To the best of our knowledge, there is a lack of counterfactuals-based techniques applied in the context of PHM. Furthermore, existing techniques present issues such as being post-hoc techniques or lacking diversity, as mentioned in the previous paragraph.

Feature selection and counterfactual generation are two approaches to interpretability very different from each other. One gives information about what features are more important to the model (or to an individual output, in the case of local techniques), while the other generates alternative input values that change the class output of the analyzed input. While these two approaches seem very different, there is a study in which both are analyzed simultaneously. In (KOMMIYA MOTHILAL et al., 2021), authors attempt to unify the two approaches. Through the use of actual causality (HALPERN, 2016), the authors are able to evaluate the necessity and sufficiency of each input feature when using different post-hoc counterfactual generation approaches. With this, they create feature rankings according to the obtained values. They compare the obtained rankings with feature rankings obtained from post-hoc feature attribution techniques. On the other hand, they evaluate the necessity and sufficiency of the most important features according to each feature attribution technique and analyze if there is a correlation, i.e., if the most important features are more necessary and/or sufficient than less important features. Results show that for techniques such as LIME and SHAP, features different from those with higher attribution scores are sometimes more necessary and sufficient. Furthermore, they found cases where correlation values between feature rankings are close to zero or even negative. They highlight the need of testing several explanations methods rather than blindly trusting one of them, because important differences can be found, as shown in this study. The need for testing several explanations techniques comes from the fact that many machine learning models (particularly, neural networks), are not inheritably interpretable. Thus, external techniques are needed for explaining, which creates the possibility of reaching different explanations. This is identified and explained by Leo Breiman (BREIMAN, 2001), and refers to it as the Rashomon effect. He shows that for a determined task, there is a multiplicity of equally accurate models that are different from each other. He demonstrates this with linear regressions. With two different sets of feature coefficients, the two regressions achieve similar performance. As these coefficients are typically interpreted as importance values for the corresponding feature, this shows that different models that achieve the same results may generate different explanations. This is important to note, as it is a common belief that phenomena have unique associated explanations. This shows that in machine learning there is a plurality of explanations for the same phenomenon. Because of this, it is important for models to be inherently interpretable, rather than using external techniques to interpret black-box models.

1.4. Thesis Organization

The remainder of the thesis is organized as follows: in section 2, the theoretical background upon which this thesis is developed is presented. Section 3 presents the proposed frameworks for interpretable neural networks in the context of PHM. In section 4, the case studies used in this thesis to demonstrate the proposed frameworks are described. Section 5 shows the results of applying the proposed framework in the case studies, and the discussion of such results. Finally, section 6 presents this work's concluding remarks and future works.

Since this thesis presents three different frameworks, there is a natural subdivision of the sections according to these three frameworks. Specifically, sections 3 and 5 are subdivided according to these three frameworks.

2. THEORETICAL BACKGROUND

This section details the concepts and theory needed for the thesis's development. This includes an extension of the literature review presented above, describing relevant research that helps to build the theoretical basis of this work, as well as to find gaps that justify this work's existence and development.

2.1. Condition-Based Maintenance (CBM) and Predictive Maintenance (PdM)

Although there are several ways to plan and execute maintenance operations, maintenance policies have been divided into two main categories (WANG, 2002): corrective maintenance (CM) and preventive maintenance (PrM). According to Wang, corrective maintenance refers to the set of actions done when an asset has failed in order to restore it to a determined operating condition, whereas preventive maintenance refers to actions done in order to maintain an operating condition, before the asset fails. While CM consists on letting the asset fail before repairing it, PrM establishes maintenance actions over predefined periods of time. CM is suboptimal due to increases in downtimes, risks, and costs. On the other hand, PrM is suboptimal due to unnecessary maintenance actions. The process of degradation is constantly affected by diverse factors, including environmental and operating conditions. This could lead to maintenance actions before needed, increasing maintenance costs. According to (JARDINE; LIN; BANJEVIC, 2006), physical assets in the industry have become more complex with the constant development of new technologies, which leads to an increase in reliability requirements. This, in turn, leads to a higher cost in maintenance actions. The inefficiency of the two maintenance policies described encourage the development of dynamic policies to address the issues mentioned above.

As an alternative to CM and PrM policies, maintenance can be done based on the asset's health state. Through the use of sensors, assets can be monitored to find indicators of the current and future component's health condition. This results in a dynamic maintenance policy that tries to reduce unexpected downtimes and maintenance costs at the same time. In the literature, this kind of policy is referred to as either condition-based maintenance (CBM) or predictive maintenance (PdM). It is common for the two concepts to be used

interchangeably. For example, in (JARDINE; LIN; BANJEVIC, 2006), authors define CBM as “a maintenance program that recommends maintenance decisions based on the information collected through condition monitoring”. In (SELCUK, 2017), CBM is described as “a set of maintenance processes and capabilities derived from real-time assessment of weapon system condition obtained from embedded sensors and/or external test and measurements using portable equipment. The goal of CBM is to perform maintenance only upon evidence of need”. On the other hand, in (MOBLEY, 2001), predictive maintenance is defined as “a condition-driven preventive maintenance program. Instead of relying on industrial or in-plant average-life statistics (i.e., mean-time-to-failure) to schedule maintenance activities, predictive maintenance uses direct monitoring of the mechanical condition, system efficiency, and other indicators to determine the actual mean-time-to-failure or loss of efficiency for each machine-train and system in the plant”. In (SUSTO et al., 2015), PdM is described as “where maintenance is performed based on an estimate of the health status of a piece of equipment”. In (SELCUK, 2017), PdM is related to maintenance actions based on the early detection of signs of failure. Authors in (HASHEMIAN; BEAN, 2011) use the two concepts (CBM and PdM) as synonyms. Furthermore, authors in (FLORIAN; SGARBOSSA; ZENNARO, 2021) define PdM as a kind of CBM. However, in the rest of the thesis, they use both concepts as synonyms. Authors in (BERGHOUT et al., 2021) follow the same approach. As a contrast, authors in (CARABIN; WEHRLE; VIDONI, 2020) use the concept of “condition-based predictive maintenance” to refer to the same policy.

Despite the confusion between the two terms, the authors in (VAN HORENBEEK; PINTELON, 2013) mention the two terms and describe their differences. According to them, PdM allows the planning of maintenance actions according to predictive information the data may give about an asset’s health state, whereas CBM uses the information about the current state of the asset to determine if maintenance actions must be taken, based on degradation thresholds. According to IBM (YARMOLUK; TRUEMPI, 2019), the main difference between CBM and PdM is timing. Both are maintenance policies based on the constant monitoring of assets through a variety of sensors. However, CBM relates to the assessment of the current health condition through predefined thresholds and techniques for diagnosis, while PdM focuses on predicting the future behavior of the asset and early detection of failures. According to the article, CBM relates to the use of monitoring for maintenance before the 2010s, mainly due to the fact that measurements were useful for indicating failure

or fault conditions but not for giving predictive information. Thus, due to technological constraints, CBM did not allow long-term maintenance planning. On the contrary, PdM is described as an evolution of CBM, in which the rise of IoT, Big Data, and new technologies allow the collection of more complex kind of data and in larger volume. Together with the development of techniques for data analysis and pattern recognition (including ML and DL), not only analysis of the current health state of an asset can be made, but also the evolution of the current health state can be analyzed in order to find trends and degradation indicators which could predict a failure. With this information, failures can be anticipated and thus plan future maintenance actions.

Considering the information presented above, CBM is considered in this work as a maintenance policy in which rudimentary monitoring and simple analysis techniques are used to aid in the identification of faults and failures, in order to take maintenance actions. PdM, on the other hand, is considered as a policy in which high-technology sensors and powerful techniques for data analysis are used in order to diagnose and predict faults and failures. Even though according to the literature, diagnosis is related to CBM more than to PdM, the use of modern techniques for analysis of large amounts of data allows a more accurate diagnosis of a health state than that achieved by CBM, which is why it is believed that they must be considered.

2.2. Prognostics and Health Management (PHM)

Following the discussion in the previous subsection, prognostics and health management (PHM) refers to the set of techniques, processes and approaches needed to implement CBM and mainly PdM policies. According to (REZAEIANJOUYBARI; SHANG, 2020), PHM uses monitored data from an equipment or system for anomaly detection, fault diagnosis and prognostics. The authors state that PHM reduces costs for owners, operators and society. PHM encompasses not only data-driven techniques, but also experience-based techniques, model-based approaches, and statistical based approaches (BAUR; ALBERTELLI; MONNO, 2020). However, in this thesis, PHM is narrowed down to data-driven techniques only. According to (ATAMURADOV et al., 2017) PHM consists of seven steps:

- Data acquisition: Is the collecting and storage of data. Without data, a PHM framework is not possible.
- Data preprocessing: Refers to the process of cleaning and analyzing data before the main processing step. This step involves the correction of errors that come with raw data, as well as feature extraction, evaluation and selection. The idea of feature extraction is to find information within the data that relates to degradation and/or failure. Feeding a model with this kind of data results in better performance, as compared to the use of raw data. Feature extraction techniques are divided into time-domain based, frequency based and time-frequency based (JARDINE; LIN; BANJEVIC, 2006).
- Detection: Is the process failure and/or anomaly detection through data analysis techniques.
- Diagnostics: Is the process of not only detecting a failure, but also describing it in terms of which component is failed, the failure mode, and the level of failure severity.
- Prognostics: Refers to the process of determining the remaining useful life (RUL) of a component or predicting a future health state.
- Decision making: Is the analysis of the results of a chosen technique for detection, diagnosis, and/or prognostics and selecting and maintenance action.
- Human-machine interface: Refers to the interaction of a technician with the machine after or during the execution of the previous steps. This includes the interaction through a graphical user interface (GUI) for visualization, analysis and task execution.

Some data-driven techniques used for PHM include artificial neural networks (ANN), deep learning-based models, support vector machines (SVM), Bayesian methods, Markov models, gaussian processes (GP), random forests (RF), among others. Currently, though, PHM is being particularly related to machine learning (ML) and deep learning (DL) (BIGGIO; KASTANIS, 2020; FINK et al., 2020; REZAEIANJOUYBARI; SHANG, 2020). This is due to the success of these kinds of techniques in many areas and the development of industry 4.0. Among other, some of the current challenges of ML and DL for PHM are:

- Data scarcity
- Highly specialized models

- Model selection
- Benchmarking
- Domain adaptation
- Interpretability

Interpretability, as mentioned in the Introduction section, presents a challenge for ML and DL in PHM. Due to their "black-box" nature, ML and DL are hard to understand, both in terms of the model and the results. According to (REZAEIANJOUYBARI; SHANG, 2020), explainable DL attempts to "open the black-box" in order for models to be more transparent. This challenge is the principal focus of this thesis.

2.3. Machine Learning

Machine learning (ML) refers to a set of algorithms in which data is used to adjust its parameters in order to do a predefined task. According to (GOODFELLOW et al., 2016), there are three main characteristics in a ML algorithm: task, performance and experience. In order to a ML algorithm to learn, it must use experience to improve performance in the predefined task. These three characteristics are described in the following subsections.

2.3.1. Task

Task refers to what it is expected for the model to do after training. In the context of PHM, the three most commonly used tasks are classification, regression, and anomaly detection.

2.3.1.1. Classification

In this kind of task, the algorithm must decide to which of k possible categories the input belongs. This means the algorithm must generate a function $y = f(x),\ f: \mathbb{R}^n \rightarrow \{1,2, \dots k\}$,

referred to as *mapping function*, that takes the input data and outputs a discrete value. Some examples of algorithms used for classification are:

- Artificial Neural Networks
- Support Vector Machines (SVM)
- Decision Trees
- Naive Bayes

In the context of PHM, classification tasks have two main applications: health state diagnostics and prognostics. In the former, present and past information is used to identify the present health state of a physical asset. In the latter, the same information is used to predict the physical asset's future health state. One example of health state prognosis in the O&G industry are presented in (FIGUEROA BARRAZA et al., 2020), in which neural networks-based algorithms are used to predict the health state of a water injection pump for well production stimulation. In this example, there are $k = 4$ classes, which represent the four possible health states for the pump: normal, incipient failure, degraded failure, and critical failure.

2.3.1.2. Regression

Regression is a task where the mapping function $f(x)$ outputs continuous values. Thus, $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$, where n is the number of input features and m is the dimension of the output vector. Some examples of algorithms used for regression are:

- Artificial Neural Networks
- Linear regression
- Polynomial regression
- Lasso regression
- Logistic regression
- Support Vector Machines
- Random Forests

In the context of PHM, regression is used for prognostics, diagnostics, and remaining useful life (RUL) estimation. In the case of prognostics and diagnostics, regression

algorithms are used to measure variables that indicate the health state of the asset. For example, authors in (LI et al., 2018) use Random Forest regression to estimate the state of health (SoH) of lithium-ion batteries. RUL estimation refers to the amount of time left for the physical asset to fail. This is crucial for predictive maintenance actions. As an example, authors in (LI; DING; SUN, 2018) use convolutional neural networks (CNN) for RUL estimation of a turbofan engine. The dataset used in this work, referred to in the literature as Commercial Modular Aero Propulsion System Simulation (C-MAPSS) (FREDERICK; DECASTRO; LITT, 2007), has been widely used as benchmark for RUL estimation models. This shows how important RUL estimation is within predictive maintenance.

2.3.1.3. Anomaly Detection

Anomaly detection refers to the identification of rare events or observations within the data. To do this, data must be analyzed in order to define an expected behavior an identify those observations that do not belong to it. Examples of algorithms used for anomaly detection are:

- Artificial Neural Networks (autoencoders)
- Principal Component Analysis (PCA)
- Isolation Forests
- One-class SVM
- Bayesian networks
- Hidden Markov Models (HMM)

In the context of PHM, anomaly detection has an important role. Typically, most of an asset's collected data will correspond to normal operation conditions (i.e. without failure). The proportion between this kind of data and data corresponding to failure states commonly is unbalanced. This is an issue for a classification model, as it will recognize observations corresponding to a normal operation state more easily than those corresponding to failures. By using an anomaly detection algorithm, this issue can be tackled, as the expected behavior may be defined as that corresponding to the normal operation of the asset. In (FIGUEROA BARRAZA et al., 2020), authors use an autoencoder to predict levels of CO_2 concentration in natural gas after passing through a treatment plant. In this case, the dataset contains less

than 5000 observations, with 27.3% of the data corresponding to CO_2 levels above the normal. In this case, an anomaly detection algorithm is more suitable than a typical classification algorithm.

2.3.2. Performance

Performance refers to a measure of the quality with which the defined task is performed. Thus, it is directly related to the task to be performed. In the following subsections, some performance metrics are described.

2.3.2.1. Confusion Matrix

A confusion matrix is not a performance metric. However, it is a useful tool for visualization, evaluation, and definition of other performance metrics in the context of classification or anomaly detection. Figure 1 shows a representation of a confusion matrix for a model with two classes. Each letter indicates the number of observations belonging to each of the possible situations indicated in the matrix. Thus,

- A: number of observations correctly classified as “Class 1”. Also referred to as true positives (TP)
- B: number of observations classified as “Class 2” that belong to class 1. Also referred to as false positives (FP)
- C: number of observations classified as “Class 1” that belong to class 2. Also referred to as false negatives (FN)
- D: number of observations correctly classified as “Class 2”. Also referred to as true negatives (TN)

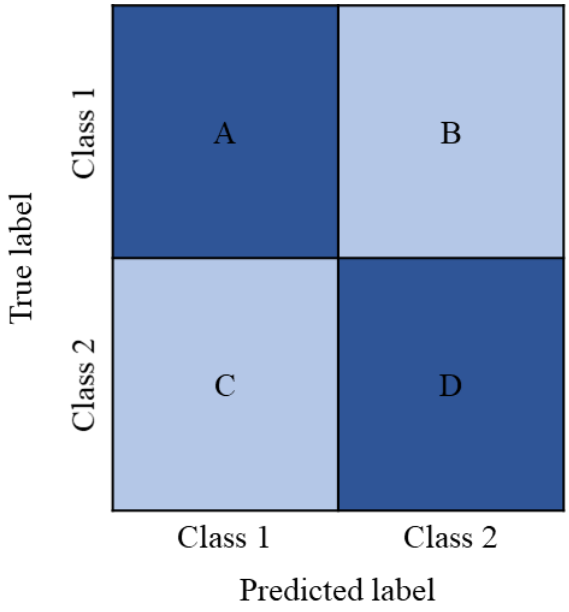

Figure 1 - Confusion matrix representation for a binary classification problem.

A confusion matrix is suitable for binary classification as well as for multi-class classification. The difference lies in the size of the matrix.

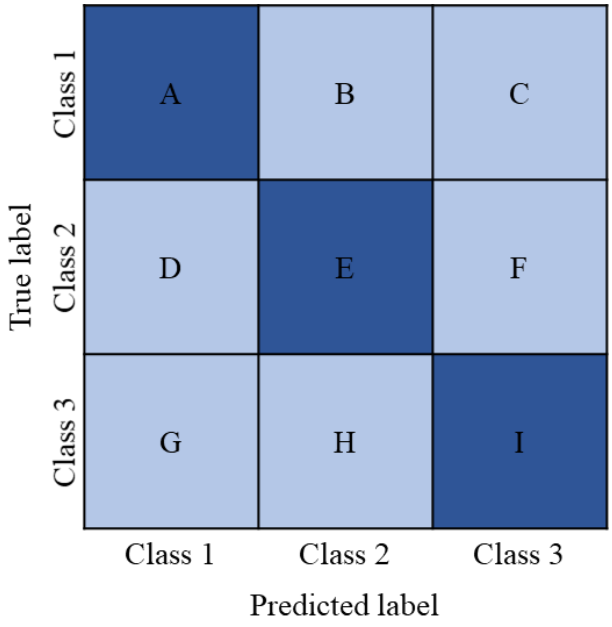

Figure 2 - Confusion matrix representation for a multi-class (k=3) classification problem.

In the context of PHM, classes typically represent health states. In this sense, a confusion matrix shows important information about the model's performance. In the case depicted in Figure 1, it is shown that there are different types of mistakes. Assuming Class 1 represents a normal operation state and Class 2 represents failure state, B would indicate the number of

observations that belong to a normal operation condition being wrongly classified as in failure state. In turn, C would indicate the number of observations that belong to a failure state being wrongly classified as in normal operation condition. While the former would result in a false alarm, the latter would result in a failure being unnoticed, which could lead to accidents. In PHM, models should try to avoid these mistakes, with the latter being prioritized.

2.3.2.2. Accuracy

Accuracy is a performance metric used in classification tasks. It is defined as

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \quad (1)$$

Following the confusion matrix example shown in Figure 2, accuracy corresponds to:

$$Accuracy = \frac{A + E + I}{A + B + C + D + E + F + G + H + I} \quad (2)$$

Accuracy is useful as a performance metric when classes are balanced. If this is not the case, accuracy may give misleading information. For example, in a binary classification task, if 80% of the data is concentrated in one class, the algorithm could tag every observation with that class and achieve an accuracy of 80%, giving a false sense of high performance.

2.3.2.3. Precision

Precision is a performance metric for each class. It indicates the fraction of correctly labeled observations among all the observations labeled to the analyzed class by the model. Following the example of Figure 2, precision for class 1 is defined as:

$$precision_1 = \frac{A}{A + D + G} \quad (3)$$

2.3.2.4. Recall

Recall is also a performance metric for each class. It indicates the fraction of correctly labeled observations among all the observations truly belonging to the analyzed class. Following the example of Figure 2, recall for class 1 is defined as:

$$recall_1 = \frac{A}{A + B + C} \tag{4}$$

2.3.2.5. F1-score

F1-score is a metric for classification tasks defined as the harmonic mean of the precision and recall metrics defined previously. Following the example of Figure 2, the F1-score for class 1 is defined as:

$$F1 - score_1 = 2 \cdot \frac{precision_1 \cdot recall_1}{precision_1 + recall_1} \tag{5}$$

2.3.2.6. Mean Squared Error

Unlike the previously described performance metrics, the mean squared error (MSE) is a metric for regression tasks. It is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \tag{6}$$

, where y_i is the i-th component of the model's predicted output, and $\tilde{y}_i$ is the i-th component of the real output.

2.3.3. Experience

Experience is what ML models use in order to learn. Experience is presented to a model in the form of a dataset. The way the dataset is structured affects the type of model to be chosen. There is a division of ML models in terms of dataset characteristics, which is presented in the next subsections.

2.3.3.1. Supervised Learning

In supervised learning, each observation in the dataset has an output value to which observations relate to. In this way, the model has the objective of generating a mapping function. Depending on the kind of output, supervised learning algorithms are divided in two: classification and regression, which are explained above.

2.3.3.2. Unsupervised Learning

In unsupervised learning, as opposed to supervised learning, there is no output value to which observations are mapped. Thus, there is no mapping function to be generated. In this sense, unsupervised learning algorithms are mainly used for more complex tasks than supervised learning algorithms. Some examples are:

- Clustering
- Denoising
- Anomaly Detection
- Dimensionality Reduction

Regarding anomaly detection, depending on the algorithm, it may be classified as unsupervised learning or a hybrid between unsupervised and supervised learning, referred to as semi-supervised learning. While unsupervised learning algorithms do not require the use of tags, semi-supervised learning algorithms uses tags to determine which kind of data is used to train a model. This is further explained in the next section.

In the context of PHM, unsupervised learning algorithms demand less from the dataset, compared to supervised learning algorithms. In the latter, there is a tagging process

required that can be time demanding and sometimes even unfeasible. However, the information given by a supervised learning model is more valuable regarding predictive maintenance than that given by an unsupervised learning model. Even though unsupervised learning models are useful for finding complex relations within the dataset, supervised learning models are needed for diagnostics and prognostics.

2.3.4. Training process

In the majority of ML models, the same process is used for defining an adequate mapping function, which is also used in this thesis. As stated in this chapter, ML models learn from experience, presented as a dataset. This dataset is typically divided into train and test set. While the train set is used for adjusting the model's parameters to define the mapping function, the test set is used to evaluate the performance of the resulting model. The objective of this division is to measure the model's performance on unseen data. Another objective is to overcome overfitting, which will be explained in the following subsection. Some algorithms also divide the train set, generating a validation set. This is used for evaluation criteria during the training process, which in turn can help to define a stopping criterion.

For the many of the ML algorithms mentioned in this thesis, the training process itself consists of the minimization of a loss function, which must lead to a maximization of performance. Some loss functions typically used in ML are the following:

- Mean Squared Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \tag{7}$$

, where y_i is the i-th component of the model's predicted output, and $\tilde{y}_i$ is the i-th component of the real output.

- Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \tilde{y}_i| \tag{8}$$

, where y_i is the i-th component of the model's predicted output, and $\tilde{y}_i$ is the i-th component of the real output.

- Cross-Entropy:

$$Cross - Entropy = -\sum_{i=1}^{C} y_i \log \tilde{y}_i \quad (9)$$

, where y_i is the i-th component of the model's predicted output, and $\tilde{y}_i$ is the i-th component of the real output.

2.3.5. Limitations: Overfitting

Despite the fact of ML algorithms being promising, they have some limitations. The three main one relates to overfitting. Overfitting is a situation in which the model has great performance in the train set but poor performance in unseen datapoints. In the context of PHM, this is a relevant issue. A change in operating conditions may change the kind of data being fed to the model. If this change was not present during the training process, an overfitted model will present a decrease in performance when evaluating these observations, increasing risks. Figure 3 shows an example of overfitting. In it, it can be seen that, at first, both the training error (error associated to the training set) and the validation error decrease at a fast rate as the training process advances. This relates to the model finding basic relations within the data. As training advances, both errors decrease but with a gap. This gap indicates that the training process is losing generalization capacity, since the learned dependencies are more effective in the training data than in the unobserved datapoints. The red line indicates a situation where the generalization error no longer decreases, but starts to increase. After the red line, the model is considered to be overfitted, as the dependencies encoded by the model are only applicable to the training set and do not relate other observations outside the training dataset.

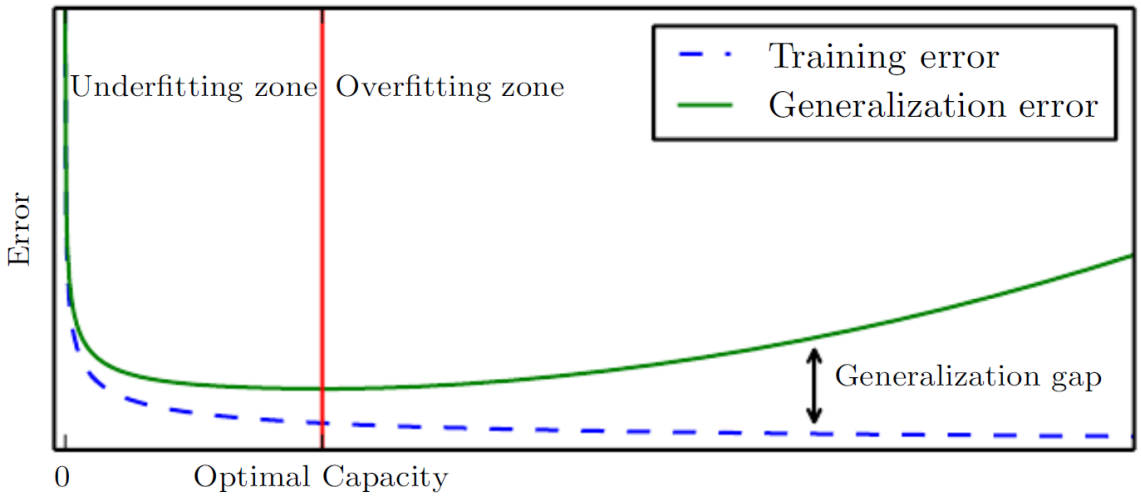

Figure 3 – Example of a model training process where overfitting occurs. Source: (GOODFELLOW et al., 2016)

2.3.6. Selected Algorithms

This subsection describes two ML algorithms used in this work, namely random forests (RF) and artificial neural networks (ANN).

2.3.6.1. Random Forests

A random forest is an algorithm that can be used for classification, regression and anomaly detection tasks. It is a composition of several decision trees. A decision tree is an algorithm that iteratively splits the data space into different portions according to different conditions to associate an output to each of them.

The structure of a decision tree is depicted in Figure 4. It consists of a root (or root node), nodes and leaves. At first, the algorithm decides which input feature will serve as root node. This is done by analyzing how good the split is for every feature and every possible cutoff value within each feature. For classification and anomaly detection tasks, this is done by analyzing a concept called “Gini impurity” (BREIMAN et al., 1984). Gini impurity is a measure of the likelihood of doing a random incorrect classification. It is defined as

$$G = \sum_{i=1}^{C} \Pr(i) \cdot (1 - \Pr(i)) \quad (10)$$

, where $\Pr(i)$ is the probability of choosing an observation corresponding to class i (according the distribution of classes within the dataset), and C corresponds to the total number of classes. As the impurity value gets higher, the more likely it is to pick a random observation and randomly classify it wrongly. For regression tasks, the MSE (as presented in equation 7) between the predicted values by the tree and the actual output values is analyzed in order to determine the quality of the possible split. Regardless of the task, the tree analyzes each possible split, and the one with the least sum of Gini impurity for each resulting division (for classification tasks) or least MSE value (for regression tasks) is the best split.

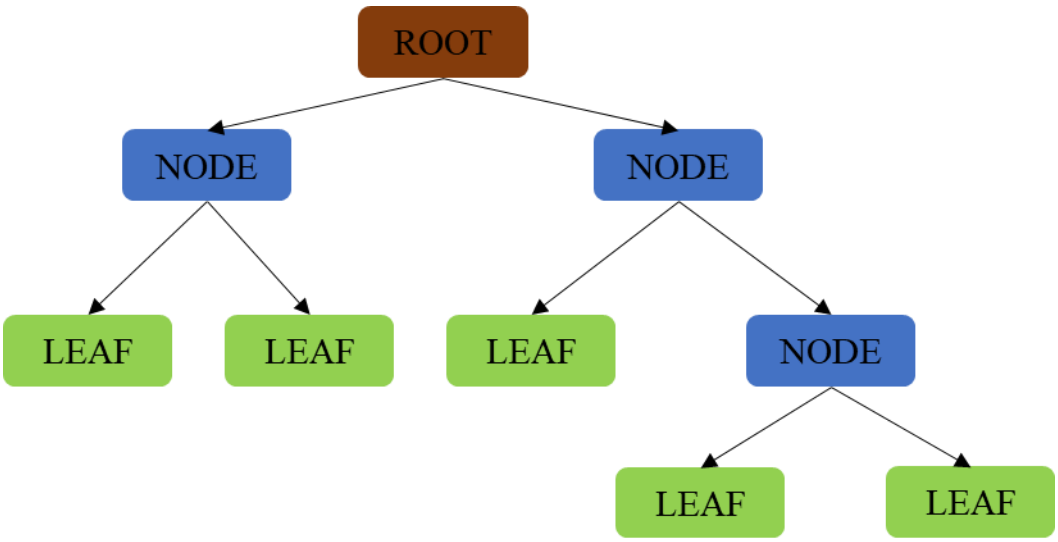

Figure 4 – Representation of a decision tree.

The same process described for the root occurs for the nodes. A node becomes a leaf (meaning data is no longer split) if a split is no longer beneficial (there is no decrease in gini impurity) or is not permitted due to constraints. For example, in regression tasks it is common to stop the splitting of the data when the amount of data to be split is less than a predefined threshold. This is done to prevent overfitting.

Despite decision trees being a useful tool, they are prone to overfitting. According to (HASTIE; TIBSHIRANI; FRIEDMAN, 2009), decision trees are not flexible when analyzing new samples. To solve this issue, random forests use several decision trees. Each

tree in the forest is trained with a dataset created from a bootstrap sample of the original dataset. The training process differs from a typical decision tree only in the fact that each node in each tree analyzes a random subset of input features, instead of the whole set of features. This, along with the use of bootstrap datasets for each tree, leads to a model with better generalization capabilities. Finally, results are averaged through all trees in the forest to deliver a prediction.

2.3.6.2. Artificial Neural Networks

Within ML algorithms, Artificial Neural Networks (ANN) have stood out in recent years due to their state-of-the-art results in many domains and to their processing capabilities. An ANN is a data-driven model used to encode a phenomenon. The structure of an ANN is presented in Figure 5. It consists of neurons organized into layers, and is inspired in the way neurons interact in the brain to process information from the environment. In the figure, each node in the input layer represents one feature. They are linearly combined using trainable weights (represented by the connections between nodes) to output the values of the neurons into the next layer. To add nonlinearity to the model, these values are passed through an activation function. Finally, a trainable bias term is added. Thus, the values in each layer (except the input layer) are defined as:

$$a_i^j = f\left(\sum_k W_{k,i}^j \cdot a_k^{j-1} + b^j\right) \tag{11}$$

, where a_i^j is the value (also called activation) of the i-th neuron in the j-th layer, $w_{k,i}^j$ is the trainable weight connecting the k-th neuron in the previous layer with the i-th neuron in the current layer, a_k^{j-1} is the activation of the k-th neuron in the previous layer, b^j is the bias term of the previous layer and f corresponds to the activation function of the current layer. According to (LECUN; BENGIO; HINTON, 2015), linear models can work with the inputs in simple regions, hindering their modelling capacity. Thus, activation functions are used in the neural networks to overcome this issue. There are several activation functions typically used for ANN. Some of them are:

- Sigmoid:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{12}$$

- Softmax:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \tag{13}$$

- ReLU:

$$f(x) = \begin{cases} 0 \; if \; x \leq 0 \\ x \; if \; x > 0 \end{cases} \tag{14}$$

- Tanh:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{15}$$

- Linear:

$$f(x) = x \tag{16}$$

ANNs can be divided by the type of task: classification or regression. In classification tasks, the output layer has one neuron per class. The one with the highest activation value indicates the class to which the model associates the corresponding input. In regression tasks, the output layer has one neuron per output dimension, which corresponds to the continuous value the model calculates.

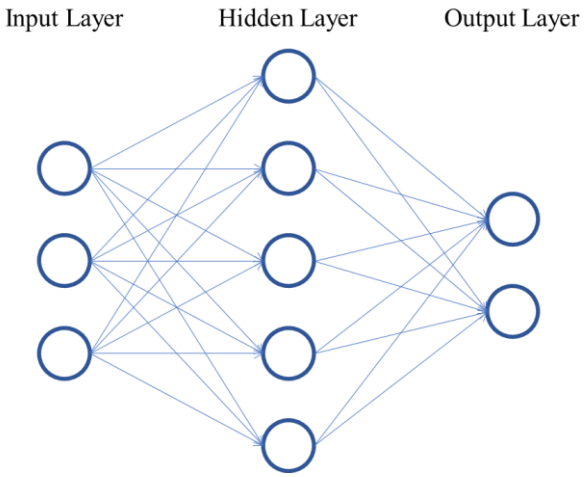

Figure 5 – Artificial neural network representation.

Regarding interpretability, ANNs lack interpretability as the number of neurons grows. For a simple model with a low-dimensional input, few neurons in the hidden layer (<5) and low-dimensional output, the inputs can be tracked down after training in a way that a human with knowledge in calculus and algebra can understand how the combination of the input values generates the output. Applying equation 11, the output is calculated as:

$$y_i = f^1\left(\sum_j W^1_{j,i} \cdot \left[f^0\left(\sum_k W^0_{k,j} x_k + b^0\right)\right] + b^1\right) \tag{17}$$

, where x refers to the input, y is the output, and superscripts 0 and 1 indicate the input layer and the hidden layer, respectively. Although possible, it is difficult to understand the equation shown above. To quantify the effect of every input is a task that gets more arduous as the number of neurons increases. Thus, though interpretability is possible in theory, it is seldom achieved in practice.

2.3.6.2.1. Backpropagation and learning process

An ANN is optimized through an algorithm called *backpropagation* (RUMELHART; HINTON; WILLIAMS, 1986), in which the weights and biases in the network are modified to minimize a loss function. The basic principle of backpropagation is to use the different

partial derivatives of the cost function (also referred to as error function) to propagate the error from the output to the inputs to update weights and biases. This is done using the chain rule. As an example, for a weight between the hidden layer and the output layer:

$$\frac{\partial C}{\partial w_{i,j}} = \frac{\partial C}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_{i,j}} = \frac{\partial C}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{i,j}} \tag{18}$$

, where C is the cost function, $w_{i,j}$ is a weight between the hidden and the output layer, o_j is a value in the output layer, and net_j corresponds to the output value before passing through the activation function. The three terms in the equation can be further developed using the definition of neural networks and the derivatives of the chosen loss function and activation functions. After obtaining the partial derivative of the cost function with respect to the desired weight, the new weight value is calculated as:

$$w_{i,j}^{new} = w_{i,j}^{old} + -\eta \frac{\partial C}{\partial w_{i,j}} \tag{19}$$

, where η is an hyperparameter called learning rate. The process of using the calculated derivatives to iteratively update weight values is called *stochastic gradient descent* (SGD) (BOTTOU, 2010). This is the basic optimization algorithm used to train neural networks. However, there are other algorithms that improve the basic idea under SGD, such as:

- Momentum (RUMELHART; HINTON; WILLIAMS, 1986): This modification accelerates SGD by keeping a fraction of the previous value of $-\eta\, \partial C/\partial w_{i,j}$ each time a weight is updated.
- Adagrad (DUCHI; SINGER, 2011): This extension of SGD makes the learning rate a varying parameter, which updates its value according to the sparsity of the parameters being updated.
- RMSProp (TIELEMAN; HINTON, 2012): As with Adagrad, RMSProp adds an adaptive learning rate, which is updated according to recent gradients for the weight being analyzed.
- ADAM (KINGMA; BA, 2015): Consists of an extension of RMSProp, in which not only the gradients are used but also their second moments.

2.4. Deep Learning

As stated before in this work, ML algorithms have proven to be a useful tool in many areas. They can find relations within data to build mapping functions with great performance levels. However, they present some limitations. According to the analysis in (NAJAFABADI et al., 2015), ML algorithms struggle in some situations such as large amount of streaming data, high dimensionality, noisy data, class imbalance, among others. According to (ZHOU et al., 2017) big data challenges ML algorithms in terms of scalability, adaptability and usability. In the context of PHM, the authors in (REZAEIANJOUYBARI; SHANG, 2020) recognize that, despite the success of some algorithms including support vector machines (SVM), random forests (RF) and particle filtering, they depend on experience and knowledge from experts to extract relevant features to feed the models. This is referred to as feature engineering.

Deep learning (DL) refers to a specific kind of ML algorithms inspired by ANNs that aim to tackle the issues mentioned above. According to (ALOM et al., 2019), an important difference between ML and DL is the process of feature engineering. While in ML algorithms features must be manually extracted, in DL features are automatically extracted from the raw data. This is an important improvement, since the need of experts for manually extracting features hindered the process of defining a model from being automated, while generating accessibility issues. Another advantage of DL over ML is the use of large amounts of data. According to (ALOM et al., 2019), ML algorithms are intrinsically limited with regard of the amount of data being fed. As more data is used for training, it is expected for models to find more complex relations within the data, have better generalization, and better performance overall. However, this does not occur with ML algorithms and massive amounts of data, reaching a point where there is no significant improvement. In the context of PHM, this is a relevant issue, since technological improvements have led to the possibility of monitoring multiple assets with multiple sensors, each with higher acquisition rates. According to (REZAEIANJOUYBARI; SHANG, 2020), by learning hierarchical representations of large amounts of data, DL algorithms can integrate feature extraction, feature selection and the task itself into an end-to-end process. This implies a great advance towards the objectives regarding Industry 4.0.

Deep learning algorithms receive that name based on the fact that ANNs are extended to deeper architectures. Due to this, they are referred to as deep neural networks (DNN). In the following subsection, DNNs are described.

2.4.1. Deep Neural Networks

Intrinsically, deep neural networks (DNN) are ANNs with more than one hidden layer. A representation of a DNN is shown in Figure 6. As with ANNs, DNNs are composed of neurons, layers and weights. Each neuron value (except for those in the input layer) represents a characteristic of the values in the previous layer. Thus, each layer extracts useful information and/or representations of the previous layer. This results in a network capable of extracted more complex information from the input data than neural networks with only one hidden layer.

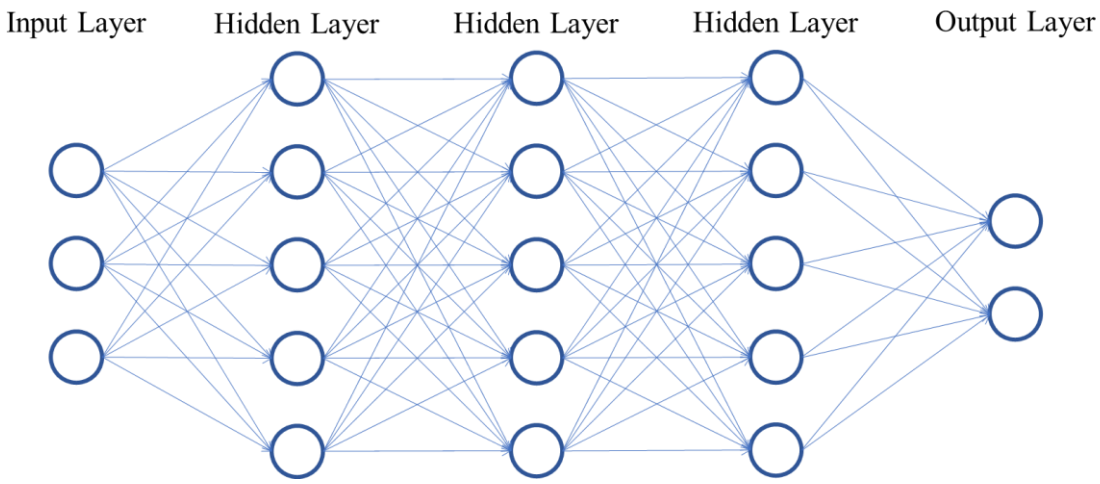

Figure 6 – Graphical representation of a deep neural network with three hidden layers.

Despite the fact of shallow ANNs being universal approximators (HORNIK; STINCHCOMBE; WHITE, 1989), meaning they can approximate any function to any level of precision, the amount of neurons needed in the hidden layer would make the training of the network unfeasible in terms of time and observations needed (BENGIO; LECUN, 2007). Instead, authors in (LAROCHELLE et al., 2009) state that neurons can be organized into a deep architecture with not one but many hidden layers. Thus, less neurons are needed and through the composition of several non-linearities, better modelling capacity is achieved.

Despite the theory behind DNNs exists before the 2010s, it is only after that year that their use began to gain popularity. This is mainly because of technological reasons. Before the 2010s, it was unfeasible to apply a DNN due to time consumption restrictions. With the possibility of using graphical processing units (GPU) to train DNNs (BERGSTRA et al., 2011), training time was reduced drastically, allowing the exploration of this and other kinds of related algorithms.

Due to the stacking of several hidden layers, DNNs are able to model complex and rare dependencies in the training data. However, sometimes this ability can lead to overfitting, as the model finds relations within the data that cannot be extrapolated to new observations. There are several ways to address this issue, which are encompassed within the concept of regularization. Some of the techniques used for regularization are detailed in the next subsection.

2.4.1.1. Regularization

According to (LECUN; BENGIO; HINTON, 2015), regularization refers to every modification of an algorithm with the objective of improving generalization and thus, preventing overfitting. In the context of DL, these techniques include parameter penalty terms, dataset augmentation, multi-task learning, early stopping, dropout (SRIVASTAVA et al., 2014), among others.

Regarding techniques based on penalty terms, two of the most used ones are the L^1 (TIBSHIRANI, 1996) and L^2 (HOERL; KENNARD, 1970a, 1970b) regularization. In DL, the L^1 regularization refers to the addition of a penalizing term to the loss function as

$$L^1(k) = \lambda \cdot \sum_i \left| w_i^k \right| \tag{20}$$

, where λ determines the strength of the regularization and w_i^k is the i-th weight in the k-th layer. Thus, when an L^1 regularization is applied to a certain layer of the network, the model, along with the primary task, tries to minimize the L^1 norm of the layer's weights. On the other hand, the L^2 regularization applies the same principle but with the L^2 norm:

$$L^2(k) = \lambda \cdot \sqrt{\sum_i \left|w_i^k\right|^2} \tag{21}$$

Though these two techniques are similar, their effect on the model and the learning process is different. According to (GOODFELLOW; BENGIO; COURVILLE, 2016), due to their derivatives, both encourage small weight values, but weights regularized by L^2 seldom reach the value zero, whereas L^1 induces solutions where a few number of weights have values greater than zero. This is explained in (TIBSHIRANI, 1996). According to the author, this is due to the shape of the constrain region of the regularizers. This is shown in Figure 7. This sparsity property of the L^1 regularization has been used for feature selection tasks (GOODFELLOW et al., 2016; VENKATESH; ANURADHA, 2019). For example, in (NG, 2004), Ng claims that in the context of logistic regression, a model of less affected by irrelevant features when it is regularized using L^1 regularization than when using L^2 regularization. When L^1 is used, the number of samples needed for an acceptable level of performance grows with the number of irrelevant features at a logarithmic rate, whereas with L^2 regularization, this rate is linear.

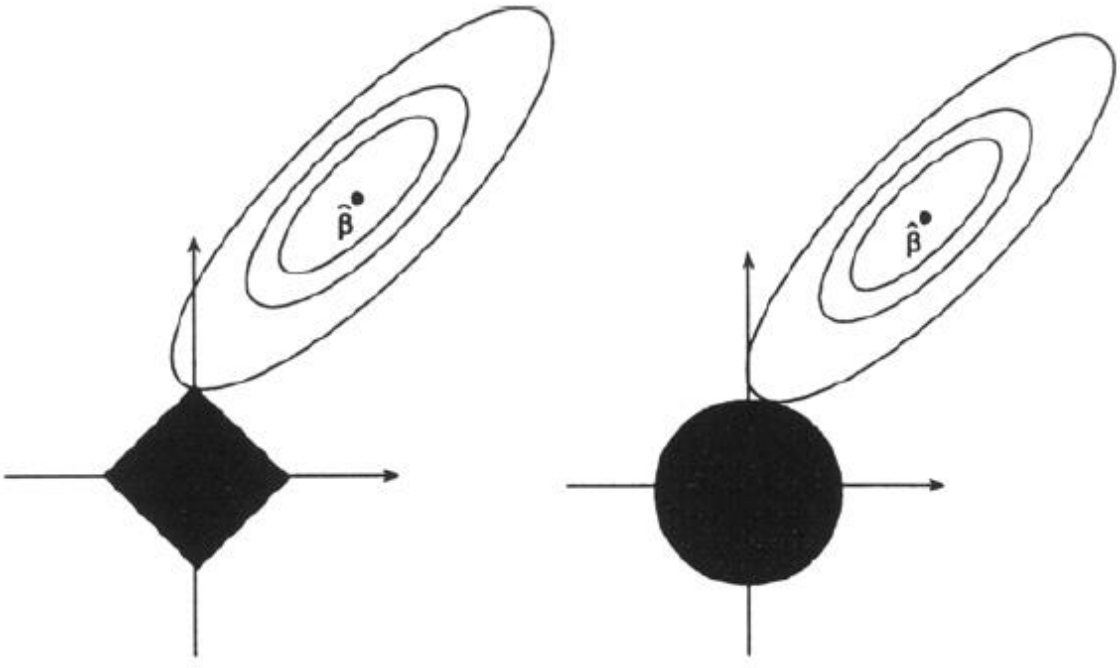

Figure 7 – Least squares solution contour for a linear model with $\boldsymbol{L^1}$ (left) and $\boldsymbol{L^2}$ (right) regularizations. Source: (TIBSHIRANI, 1996)

2.4.1.2. Multi-tasking

Neural networks are a kind of algorithm in which a loss function is minimized in order to maximize performance over a determined task. However, they have proven to be useful for multi-tasking. Multi-task learning (MTL) refers to the sharing of representations in order to learn two similar tasks simultaneously. In the context of neural networks, this can be achieved by creating separate layers for different tasks, while a portion of the layers is shared between the two tasks. An example is shown in Figure 8. In this example, the two first hidden layers are shared between the two tasks. This is useful when the two tasks are similar but not the same, and both of them can be benefited from the representations learned in those layers. Then, two branches are created in which differentiated representations are learned corresponding to each task. The kind of MTL shown in the image corresponds to hard parameter sharing. The alternative is soft parameter sharing, in which each task has its own separate layers, but the distance between the layers' parameters is regularized with distance measures such as L_2. This forces the parameters to have similar values.

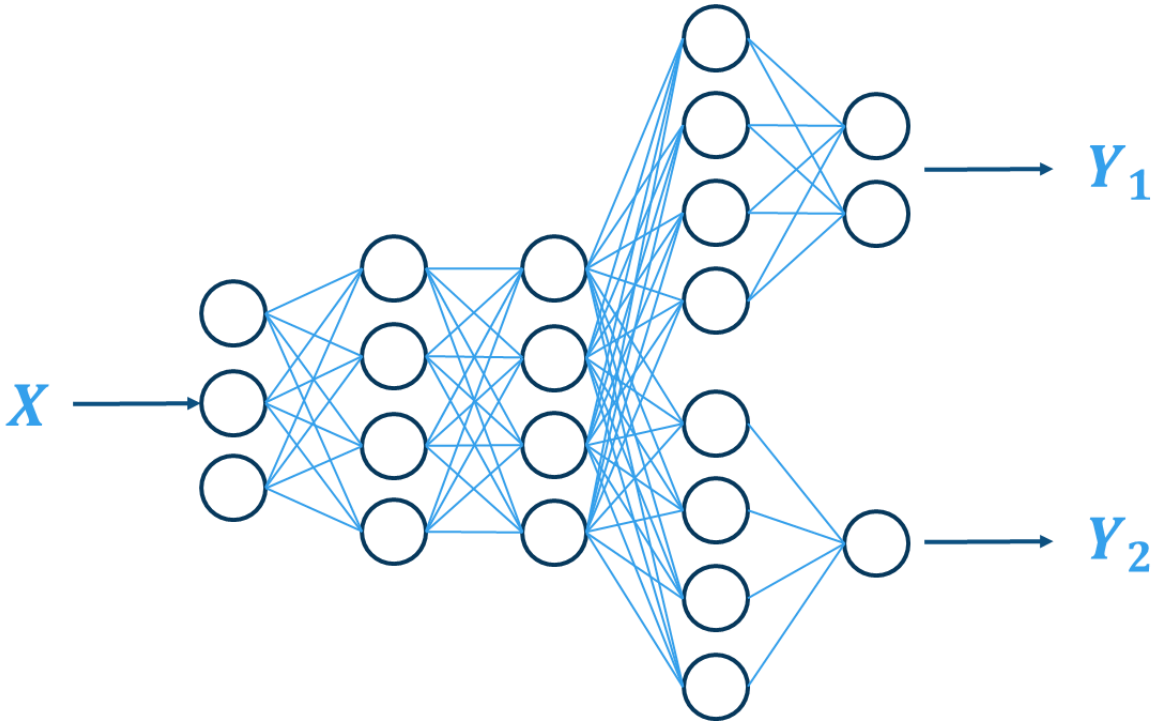

Figure 8 – Example of multi-task neural network with two tasks, one of them is a binary classification task and the other a regression task.

MTL is a useful tool against overfitting. In (BAXTER, 1997), the author showed that the risk of overfitting decreases with the number of simultaneous tasks being learned simultaneously in hard parameter sharing scheme, like the one shown in Figure 8. Overfitting occurs when the network learns a representation of the data that is excessively dependent on the training data. Thus, the trained model will have an optimal performance on the training dataset but poor performance on previously unseen data. Intuitively, MTL is useful for avoiding overfitting because the network has to learn a representation of the data able to perform well simultaneously in two different tasks. Thus, the shared layers intrinsically have a greater capacity for generalization than if they were not shared among different tasks. This is also achieved through a soft weight sharing scheme. As an indirect consequence, through MTL a model could require less data reach an adequate level of performance.

When the purpose of MTL is only to reduce overfitting of a single task, an auxiliary task is used that is not necessary for the nature of the problem, but nevertheless is used to enhance generalization. Alternatively, MTL can be used in order for two tasks to benefit from one another (PAN; YANG, 2010). Another reason to use MTL is due to requirements of the problem itself.

In the context of PHM, multi-tasking is used in order to enhance failure diagnosis and prognosis. In (KIM; SOHN, 2021), authors use a CNN-based multi-task network for simultaneous RUL prediction and health condition identification in order to improve the RUL prediction performance. While RUL prediction is a regression task, health condition identification is a classification task. According to the authors, the labeling of the data according to their health condition is done before training, thus, it is a supervised task. In (LIN et al., 2021), the authors propose to enhance RUL prediction with the aid of a classification task for fault detection. In a two-stage training process, they use the features from the fault detection task as initial values to a main network for RUL prediction, based on CNN and LSTM.

2.5. Interpretability of Deep Learning Models

This subsection describes interpretability, with emphasis in DL models. Definitions, limitations, and kinds of techniques are described.

2.5.1. Definition

Though interpretability is intuitively associated with the ability of understanding and/or interpreting a model, there is no clear definition of interpretability in practice. According to (LIPTON, 2018), model interpretability is composed of transparency (i.e. how does the model work) and explanations (i.e. what additional information can the model give). Transparency is a means to solve the "black-box" nature issue with ML models. The author states there are three levels of transparency:

- Simulatability: This level of transparency is achieved when a human can take the model inputs and parameters and produce a prediction in a reasonable amount of time. The amount of time which defines whether a model is transparent or not is subjective, but the definition still allows comparisons between models.
- Decomposability: Refers to the ability of understanding inputs, parameters and calculations within the model.
- Algorithmic transparency: This level of transparency refers to the understanding of the algorithm's learning process. DL algorithms have not reached this level of transparency yet. While the optimization processes of DL algorithms have proven to be powerful in terms of results, there is no consensus to why they work. Furthermore, there is no guarantee if they will work on new problems.

Explanations represent the other component of interpretability. They relate to additional information given by the model after training that, even though they do not help understand how the model works, they do help understand the phenomenon being modelled.

2.5.2. Goals

According to (BARREDO ARRIETA et al., 2020), interpretability of DL models has become a matter of interest in different areas mainly because of two reasons. The first one

relates to the existing gap between the research community the different business sectors, mainly due to risk issues. This applies to PHM, as companies are reluctant to rely on models they do not clearly understand to help them optimize operations while reducing downtimes and probability of failure. The other one relates to knowledge. With the potential of DL models, relations can be inferred that could help humanity understand different phenomena which could not be understood by humans alone, due to cognitive and computation limitations. In this sense, interpretability of DL models could define new paths in many research areas, including PHM.

The increasing interest towards interpretability has determined the objectives to be achieved with interpretability:

- Trustworthiness: As mentioned before in this work, the "black-box" nature of DL algorithms generates resistance from companies to adopt these algorithms in their processes. With the achievement of interpretability in deep learning, this issue will be solved by generating trust. As mentioned before in this work, this is a relevant issue in PHM.
- Causality: DL models are exceptionally good at finding patterns within data. However, they struggle to deal with causality, as recently discussed in (SCHOLKOPF et al., 2021). It is believed that interpretability can help in the task of finding causal relations between data.
- Transferability: When achieved, interpretability will help to find the boundaries and constraints of the model. This, in turn, will help to determine whether the same model can be applied in other situations. In the context of PHM, models are typically trained using historical data from one component or system, with its own operating conditions. Interpretability may help to determine if the same model can be applied to another component/system with different operating conditions.
- Informativeness: Though the main goal of ML and DL models is to support decision making, the model's output after training is not the only information needed for decision making. In this sense, interpretability will further help in this process. As an example, in the context of PHM, knowing the RUL of an asset gives an indication of when to perform maintenance. However, knowing

why and/or how the model associates the input features to a certain RUL might give insights to what kind of maintenance action must be taken.

- Confidence: It is believed that by achieving interpretability, the robustness and stability of a model could be measured through the additional information it gives.
- Fairness: As a tool to be used to better understand a model, Other kinds of analysis could be made, besides performance. One of them is fairness. It is believed that interpretability could help analyze models and results in terms of fairness and ethics.
- Accessibility: It is often believed that only experts and/or highly-technical personnel is needed when a deep learning model is being deployed, mainly because the aforementioned "black-box" nature issue. By achieving interpretability, non-experts and end users will have the possibility to be more involved in process of developing and/or evaluating a deep learning model.
- Interactivity: Similar to the previous objective, it is believed that only experts can successfully interact with a deep learning model. In situations where the user of the model is relevant, interpretability could help with the interaction of the user with the model.
- Privacy awareness: One objective of interpretability in DL models is privacy assessment. As mentioned in (MIRESHGHALLAH et al., 2020), there are threats to deep learning which include the exposure of private information, whether it is through training data, inferred relations or model parameters and hyperparameters. In this sense, interpretability of DL models could help in the development of privacy-preserving mechanisms.

2.5.3. Current limitations towards application

Despite interpretability being crucial, there is no widely accepted technique for achieving it. According to (FAN; XIONG; WANG, 2020), there are four main issues why model interpretability is not easily achievable:

- Human limitation: With machines having much more computational power than humans, DL models are being used for more complex tasks, where humans do not have a vast previous knowledge. This hinders the process of understanding results.
- Commercial barrier: Companies benefit from models that are not interpretable. They usually want to hide their models in order to prevent reverse engineering from competing companies.
- Data wildness: To obtain high quality data is unusual. This interferes with interpretability. Also, high dimensionality of data makes interpretability a difficult task. According to (HALFORD et al., 2005), humans can process up to four variables simultaneously in a single problem. This means that a problem with more than three input variables cannot be easily understood by a human being. Applying this to PHM, where problems have a number of input variables much bigger than three, interpretability without any help is almost impossible.
- Algorithmic complexity: As described in the previous chapter, DL algorithms involve a large number of complex operations, many of them involving nonlinearities in order to capture intricate relations between variables. This large amount of operations prevents humans from understanding the inner dynamics of DL algorithms.

2.5.4. Taxonomy

As mentioned before, there is no clear definition of interpretability. Consequently, there is no consensus towards a taxonomy of the existing techniques. Researchers have proposed different taxonomies for grouping techniques according to different criteria. For example, (CARVALHO; PEREIRA; CARDOSO, 2019) state that ML interpretability techniques can be categorized according to four criteria:

- Pre-model vs. In-model vs. Post-model: This criterion refers to when the techniques are applied, with respect to the construction of the model.

- Intrinsic vs. post-hoc: This refers to whether interpretability is a product of model constraints (intrinsic) or occurs after training due to an external technique (post-hoc).
- Model-specific vs. model-agnostic: According to this criterion, interpretability techniques can be specific to a type of model (model-specific) or applicable to any ML model.
- Global vs local: According to this criterion, interpretability techniques can be divided into those who seek interpretability through explanations of single predictions or a group of predictions (local) or those who try to give information about the model, its inner dynamics, and/or how predictions are affected by its components (global).

Other taxonomy for interpretability, in this case applied to neural networks, is proposed by (FAN; XIONG; WANG, 2020). It is summarized in Figure 9. As it can be seen, interpretability techniques are primarily divided into post-hoc interpretability analysis and ad-hoc interpretable modeling. While the first one refers to interpretable analysis after the model is trained, the second refers to models that are inherently interpretable. Post-hoc techniques are further classified into:

- Feature analysis: This kind of techniques analyzes neuron values in different layers to understand what features are being learned by the network.
- Model inspection: These techniques try to obtain useful information from the inner dynamics of neural networks and how they interact with the inputs.
- Saliency: These techniques build saliency maps to identify how predictions are affected by each input variable.
- Proxy: Results obtained by the model are used to create a proxy model, which is much simpler and more interpretable than a neural network. The interpretation of the proxy model is used as interpretation of the original more complex model.
- Advanced mathematical/physical analysis: In this kind of techniques, neural networks are analyzed through a mathematical/physical framework in order to obtain interpretations.
- Explaining-by-case: These techniques use representative examples to explain the general dynamics of the model.

- Explaining-by-text: This kind of techniques is restricted to problems involving text and images at the same time. They generate text descriptions of images.

On the other hand, ad-hoc techniques are classified into:

- Interpretable representation: These techniques use regularization techniques in neural networks to enhance interpretability.
- Model renovation: These techniques aim to increase interpretability by adjusting the network's components, such as neurons, activations, layers, etc.

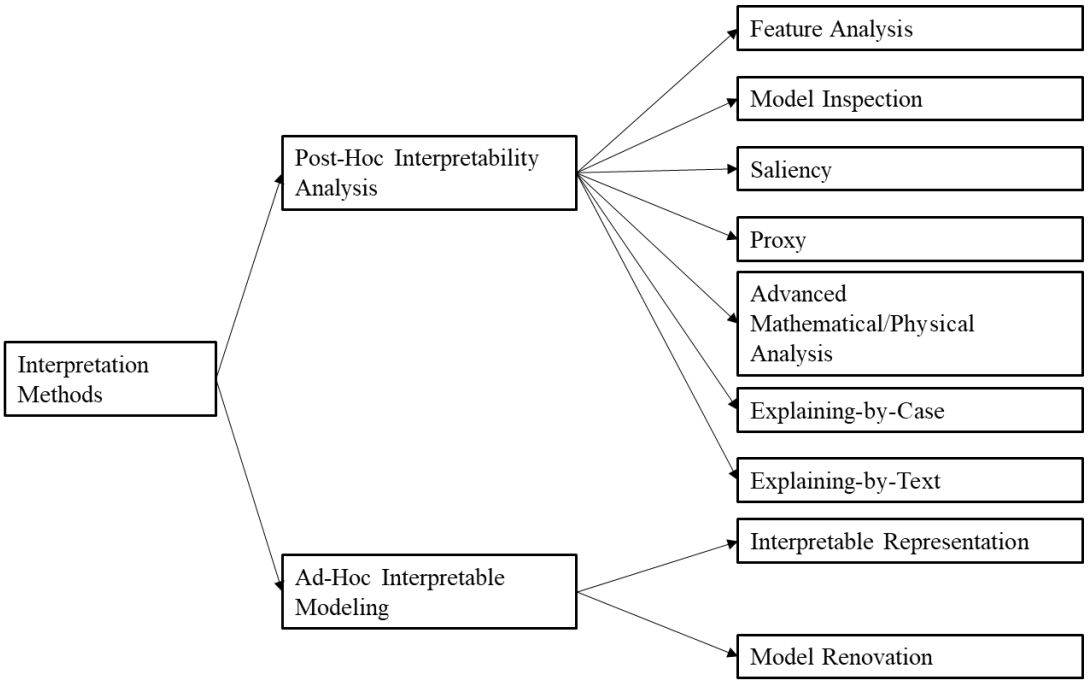

Figure 9 – Taxonomy for interpretability categorization. Source: (FAN; XIONG; WANG, 2020).

2.5.5. Performance-interpretability trade-off

Regarding interpretability in machine learning and deep learning, there is an issue commonly mentioned by researchers ((ALVAREZ-MELIS; JAAKKOLA, 2018; BARREDO ARRIETA et al., 2020; CARVALHO; PEREIRA; CARDOSO, 2019; LUNDBERG; LEE, 2017)), which states that as models get more complex and achieve better results in terms of performance, interpretability is harder to achieve. Figure 10 shows a graphical representation of this issue. In it, machine learning algorithms are distributed

according their interpretability and performance, also indicating future research guidelines. It is noteworthy that simpler models such as rule-based learning and linear/logistic regression have high interpretability but low performance compared to the rest. Consequently, future research guidelines refer to the improvement of these algorithms' performance. On the other hand, DL models have the highest performance levels but the lowest interpretability levels. Guidelines refer to an increase in interpretability while increasing performance.

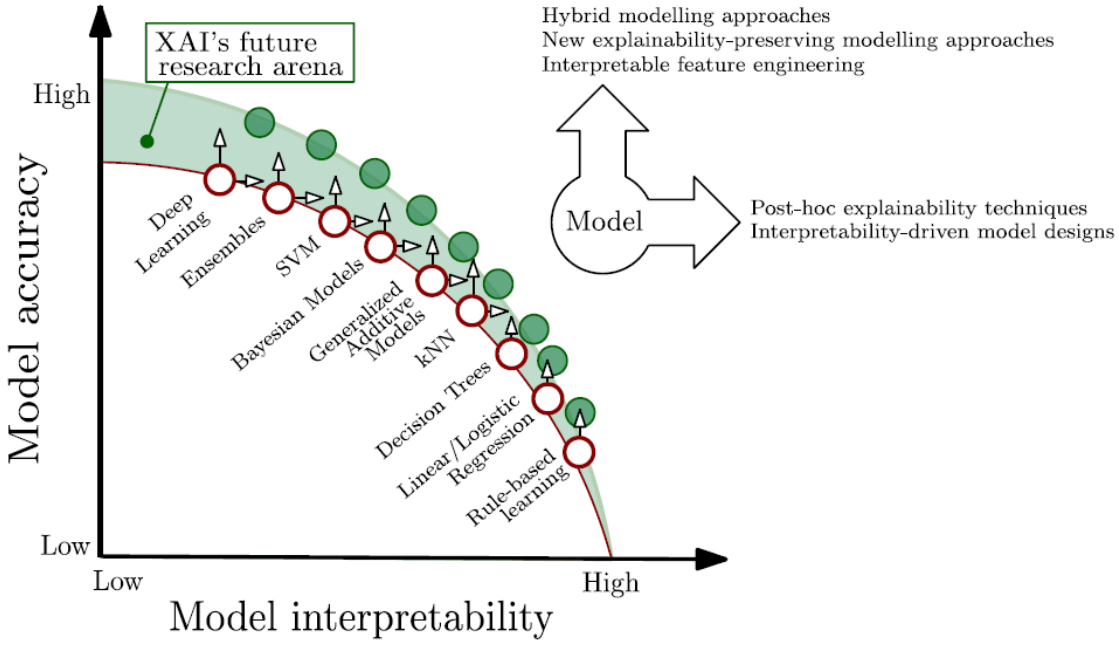

Figure 10 – Graphical representation of the trade-off between performance and interpretability. Here, accuracy is treated as a synonym for performance. Source: (BARREDO ARRIETA et al., 2020)

2.6. Feature Importance

As discussed in the previous section, there is no unique definition of interpretability. This means there are many ways to interpret a DL model, each of them giving information about different aspects. One of these aspects relates to the importance each input feature has within the model. Among the input features, there are those that have a bigger influence on the objective than others. There may also be features that do not have any influence on the objective or add noise to the model. Because of this, it is important to calculate feature importance and discard features with negative effects over the model.

Another reason why calculating feature importance is important relates to the "curse of dimensionality". Concept first coined by Richard Bellman, it refers to diverse problems that arise from problems involving high-dimensional data. In machine learning and deep learning, the curse of dimensionality relates to generalization capacity. For a model to be able to generalize and have acceptable levels of performance when evaluating a previously unseen observation, the train set must contain examples from many possible combinations of feature values. Thus, the input feature space is well represented by the train dataset. When the problem is low-dimensional, this is not an important issue, since the possible combinations are limited by a low number of dimensions. When a problem includes a large number of input features, the observations needed to cover all the possible combinations increases exponentially. In practice, the more input features a problem has, the more difficult it is for the model to generalize to unseen observations. Actually, as the number of input features increases, the number of observations needed for the model to have an adequate capacity of generalization grows exponentially (BELLMAN, 1966). In this sense, discarding irrelevant input features is important to prevent overfitting issues that lower performance.

Within interpretability, techniques for calculating feature importance[1] are very versatile, and may correspond to many of the types of techniques listed in the taxonomies described above. Regardless of how are they classified within interpretability techniques, techniques for calculating feature importance have a more consolidated taxonomy for classification. According to (JOVIĆ; BRKIĆ; BOGUNOVIĆ, 2015), techniques for calculating feature importance are divided in three:

- Filter methods: Filter methods calculate the importance of each input feature before training, regardless of the algorithm being used for training. Thus, input features are filtered and the most important ones are used for training.
- Wrapper methods: Wrapper methods evaluate different feature subsets to find the one that maximizes performance. This is done by training the model with the selected feature subset and evaluate its performance. Since the number of possible subsets to be evaluated increases rapidly with the number of features, wrapper methods use different search algorithms to find the optimal subsets.

[1] In the literature, techniques for calculating feature importance are also found as "feature selection" techniques. In this work, they are used as synonyms. This is because, despite calculating feature importance and selecting features being different processes, the two are executed sequentially in the context of neural networks.

- Embedded methods: Embedded methods are algorithms that calculate feature importance during training, not needing any external technique.
- Hybrid methods: These methods combine two or more of the methods described above. An example is presented in (BEN BRAHIM; LIMAM, 2016), where the authors use a hybrid filter wrapper approach for feature selection. They use a filter method to obtain an initial subset of features. Then, the optimal subset is obtained through a cooperative subset search and a classifier algorithm.

All of the three kinds of techniques (excluding hybrid techniques) have advantages and disadvantages. These are summarized in Table 1. In the case of filter methods, they use statistical analysis prior to the training of the model to rank features according to their importance. Some examples include the use of correlation criteria (GUYON; DE, 2003), mutual information (BATTITI, 1994), χ^2 statistics (DASGUPTA et al., 2007), among others. These techniques evaluate features or subsets of features according to their correlation with the output. Since they are techniques used before training and do not relate to the training algorithm, they can be used in diverse algorithms. Their restrictions arise mainly from the type of task the model is used to (classification, regression, anomaly detection, etc.), but not the algorithm the model is built upon. This is a positive aspect, since it allows algorithms with low interpretability to be analyzed in terms of feature importance. Since training is not needed for analyzing features and importance is obtained through statistical metrics, implementation is typically not time intensive. One disadvantage of filter methods arises from the fact that the training process is not used for calculating importance values. Thus, despite features having different relevance values within the model's training process, this information is not considered in filter methods. Another disadvantage comes from the fact that some techniques analyze single features only, which means that interactions between features are not considered. This may generate misleading results, as the importance of a feature may be diminished by the absence of another feature (ROTH, 1988). On the other hand, techniques that analyze subsets of features need to create the subsets through a search algorithm, which increases the computation time.

Table 1 – Advantages and disadvantages of feature selection techniques.

Technique	Advantages	Disadvantages
Filter	- Fast implementation. - Generally applicable to many kinds of algorithms.	- Calculations do not consider the inner dynamics of the algorithm involved. - Techniques that analyze single features do not include the interaction with other features. - Techniques that analyze subsets of features require a search algorithm for analyzing all possible subsets.
Wrapper	- Model performance indicates which is the best subset.	- Computationally expensive.
Embedded	- Implementation as the model is being trained.	- Not applicable to other kinds of algorithms.

Wrapper methods train the model with different subsets to find the one that delivers the best performance value. Although these techniques solve the issue with filter methods of not involving the training process, there is a problem of time consumption. As the number of input features grows, the number elements in the power set (set that contains all the possible subsets of input features) grows at a faster rate. For an input feature set containing n features, the number of possible subsets is $2^n - 1$. For a dataset with 10 input features, there are 1023 possible subsets. These means the algorithm should be trained 1023 times to find the optimal subset, which is unfeasible. In this sense, different wrapper methods use different approaches to define the subsets to be analyzed. Some of them are:

- Forward feature selection: Starts from analyzing single features and adds more features according to their performance.
- Backward feature elimination: Starts from analyzing the whole set of features and eliminates features according to their performance.

- Exhaustive feature selection: Analyzes all possible combinations of input features. To limit the search process, the maximum number of features allowed may be limited.
- Bidirectional search: Applies forward and backward feature analysis simultaneously.

Though some of the techniques described above try to optimize the search process, the only method that guarantees the achievement of the optimal combination of features is the exhaustive method.

Embedded methods are training algorithms that, due to their inner dynamics, there is a way of obtaining the importance of each feature without the use of any external technique. Since the feature importance values are obtained through the training of the model, there is no computational time being added, like in the other two kinds of methods. However, the main drawback of embedded methods is that they are restricted to the algorithm they belong to. In the context of DL and neural networks, the use of an embedded method requires the training of an algorithm that can calculate feature importance values, and use the results to train the desired neural network-based model. Examples of algorithms with embedded techniques for feature selection include random forests (GENUER; POGGI; TULEAU-MALOT, 2010), lasso regression and ridge regression (MUTHUKRISHNAN; ROHINI, 2017). As it can be seen, these are techniques which, when looking at Figure 10, would correspond to high interpretability and low performance techniques. As mentioned before, it would be necessary to train two separate models in order to apply these techniques in neural networks. In terms of automation and time consumption, this is not an ideal scenario. Indeed, an ideal scenario would be for neural networks to have an embedded technique for feature importance calculation. In this way, alongside training, neural network-based models would inform which are the more relevant features and which should be discarded.

In the next subsections, feature selection techniques used in this work are described.

2.6.1. Mutual Information

Mutual information (MI), also referred to as information gain (IG), measures the dependency of two variables using the concept of entropy. According to (SHANNON, 1948), the entropy of a discrete random variable X is calculated as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \tag{22}$$

, where $p(x_i)$ is the probability of every possible outcome. This is also referred to as Shannon entropy, and represents a measure of uncertainty or information inherent to the random variable's possible outcomes. Consequently, mutual information uses this concept to define the information shared by two variables X and Y:

$$MI(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y)\, log \frac{p(x,y)}{p(x)p(y)} \tag{23}$$

, where $p(x)$ represents the probability of the possible outcomes of X, $p(y)$ represents the probability of the possible outcomes of Y, and $p(x,y)$ is the joint distribution of X and Y. Mutual information indicates how much information about X can be accessed through Y and vice versa. It relates to Shannon entropy by the following relation:

$$MI\ (X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{24}$$

, where $H(X|Y)$ and $H(Y|X)$ are the Shannon entropies of the conditioned random variables.

The technique calculates the mutual information of every input variable with the output variable. Since the definition of mutual information involves random variables, it is estimated using the available dataset according to approach detailed in (ROSS, 2014). The obtained values are interpreted as importance values, and are used to create a ranking of values. It is a filter method that analyzes each feature separately. Therefore, it does not consider interactions between input features, which can generate misleading results.

2.6.2. Relief, ReliefF and RReliefF

Relief (KIRA; RENDELL, 1992) is an algorithm for feature selection that uses the variation of feature values in near observations to determine feature importance. The original algorithm, created for binary classification tasks, introduces the concepts of *nearest hit* and *nearest miss*. A nearest hit with respect to an observation p_i is the nearest neighbor with the same class as p_i, whereas the nearest miss is the nearest neighbor with the other class. The algorithm is described in Figure 11. In it, a $diff()$ function is used. For numerical values, it is defined as:

$$diff\left(x_i, x_j\right) = \frac{|x_i - x_j|}{\max(x) - \min(x)} \tag{25}$$

, where x_i and x_j are the values of the same feature for different observations Thus, $\max(x)$ and $\min(x)$ are the maximum and minimum observed values for the analyzed feature. For non-numerical values, $diff()$ is defined as:

$$diff\left(x_i, x_j\right) = \begin{cases} 1 \; if \; x_i = x_j \\ 0 \; if \; x_i \neq x_j \end{cases} \tag{26}$$

```
Set all feature values to W(f) = 0
For i = 1 to m:
      Select a random observation p_i
      Find nearest hit H_i and nearest miss M_i
      For f in F:
            W(f) = W(f) − diff(p_{i,f}, H_{i,f})/m + diff(p_{i,f}, M_{i,f})/m
```

Figure 11 – Relief pseudocode

The original contribution described (KIRA; RENDELL, 1992) works only for binary classification. However, in (KONONENKO, 1994), Kononenko extends the use of the algorithm to multi-class classification (ReliefF). Instead of finding one near miss, the algorithm finds one near miss for every other class in the model. Thus, the weight update step is modified:

$$W(f) = W(f) - \frac{diff(p_{i,f}, H_{i,f})}{m} + \sum_{C \neq class(p_i)} \frac{Pr(C) \cdot diff(p_{i,f}, M_{i,f}^{C})}{m} \quad (27)$$

, where $Pr(C)$ is the prior probability of class C (calculated according to the distribution of classes within the dataset) and $M_{i,f}^{C}$ corresponds to the f feature value for the nearest miss of p_i corresponding to class C.

In 1997, Robnik-Šikonja and Kononenko extend the ReliefF algorithm for it to be used with regression tasks (ROBNIK-ŠIKONJA; KONONENKO, 1997). The algorithm, referred to as RReliefF is detailed in Figure 12. Here, they modify the concepts of nearest hit and nearest miss, since there are no classes to determine them. Instead, they introduce three different intermediate weights which are used to update the weights associated to each feature: a weight for different prediction N_{dC}, a weight for different feature value (also referred to as "attribute") N_{dA}, and a weight for different prediction and different feature value $N_{dC\&dA}$. Also, a distance function is used, defined as

$$d(i,j) = \frac{d_1(i,j)}{\sum_{l=1}^{k} d_1(i,l)} \quad (28)$$

$$d_1(i,j) = \exp\left(-\left(\frac{rank(R_i, I_j)}{\sigma}\right)^2\right) \quad (29)$$

, where $rank(R_i, I_j)$ is the position in a ranking built using the distances of each of the selected k nearest instances I_j to R_i.

1. set all $N_{dC}, N_{dA}[A], N_{dC\&dA}[A], W[A]$ to 0;
2. **for** i := 1 **to** m **do begin**
3. randomly select instance R_i;
4. select k instances I_j nearest to R_i;
5. **for** j := 1 **to** k **do begin**
6. $N_{dC} := N_{dC} + |f(R_i) - f(I_j)| \cdot d(i,j)$;
7. **for** A := 1 **to** #all_attributes **do begin**
8. $N_{dA}[A] := N_{dA}[A] + diff(A, R_i, I_j) \cdot d(i,j)$;
9. $N_{dC\&dA}[A] := N_{dC\&dA}[A] + |f(R_i) - f(I_j)| \cdot$
10. $diff(A, R_i, I_j) \cdot d(i,j)$;
11. **end**;
12. **end**;
13. **end**;
14. **for** A := 1 **to** #all_attributes **do**
15. $W[A] := N_{dC\&dA}[A]/N_{dC}$ - $(N_{dA}[A] - N_{dC\&dA}[A])/(m - N_{dC})$;

Figure 12 – Pseudocode for the RReliefF algorithm. Here, the word “attribute” is used as a synonym for “feature”. Additionally, $\boldsymbol{f(x)}$ indicates the output value for observation $\boldsymbol{x}$.

In this way, Relief-based algorithms are filter methods that iteratively analyze features through observations and neighbors to obtain a weight value for each input feature. These weights are used to rank features to determine which of them are going to be used for training.

2.6.3. Random Forest

As described before, random forests are used for classification and regression. Due to the way they are trained, information can be obtained about the importance of each input feature, thus, making it an embedded method. When a defining a node, the possible splits are analyzed in terms of Gini impurity or MSE. Therefore, a split is characterized by the input feature used for splitting and the decrease in Gini impurity or MSE. This is used to obtain the relevance of each feature in the model. For each node of each tree, the decrease of impurity or MSE is calculated for all input features. The mean decrease in impurity or MSE represents the importance value of each feature. These values are used to rank features and select which of them are going to be used in deployment.

2.6.4. "AFS: An Attention-based mechanism for Supervised Feature Selection"

As was mentioned in the Introduction section, there is interest from researchers to develop techniques for DNN interpretability. One of them is the work by Gui et. al. (GUI; GE; HU, 2019a). In it, researchers propose the use of attention mechanisms (BAHDANAU; CHO; BENGIO, 2015) to build a module for feature selection within DNNs. A representation is shown in Figure 13. As seen in the image, the input values enter the attention module through a dense layer used to compress the input feature space and keep important information. This layer has a $tanh$ activation function. The compressed information is then passed through a number of attention nets equal to the initial number of input features. These attention nets have the objective of determining the importance of each input feature. The architecture of each attention net is the same for each input. However, weights are not shared, meaning each input feature has its own corresponding attention net with its own different weights. While the number of hidden layers, neurons per layer, and activation functions are model hyperparameters, the last layer must have two neurons and a $softmax$ activation function. The reason of this is for the output layer to represent the probabilities of selection and non-selection of the input feature. Each of the probabilities of selection (one for each input feature) represents the importance value of its corresponding feature. These values are rearranged into an array and used to multiply the input feature vector, resulting in weighted feature values that enter the rest of the network, referred to in the original work as *learning module*. According to the authors, the learning module can be a DNN, a convolutional neural network (CNN) or a recurrent neural network (RNN).

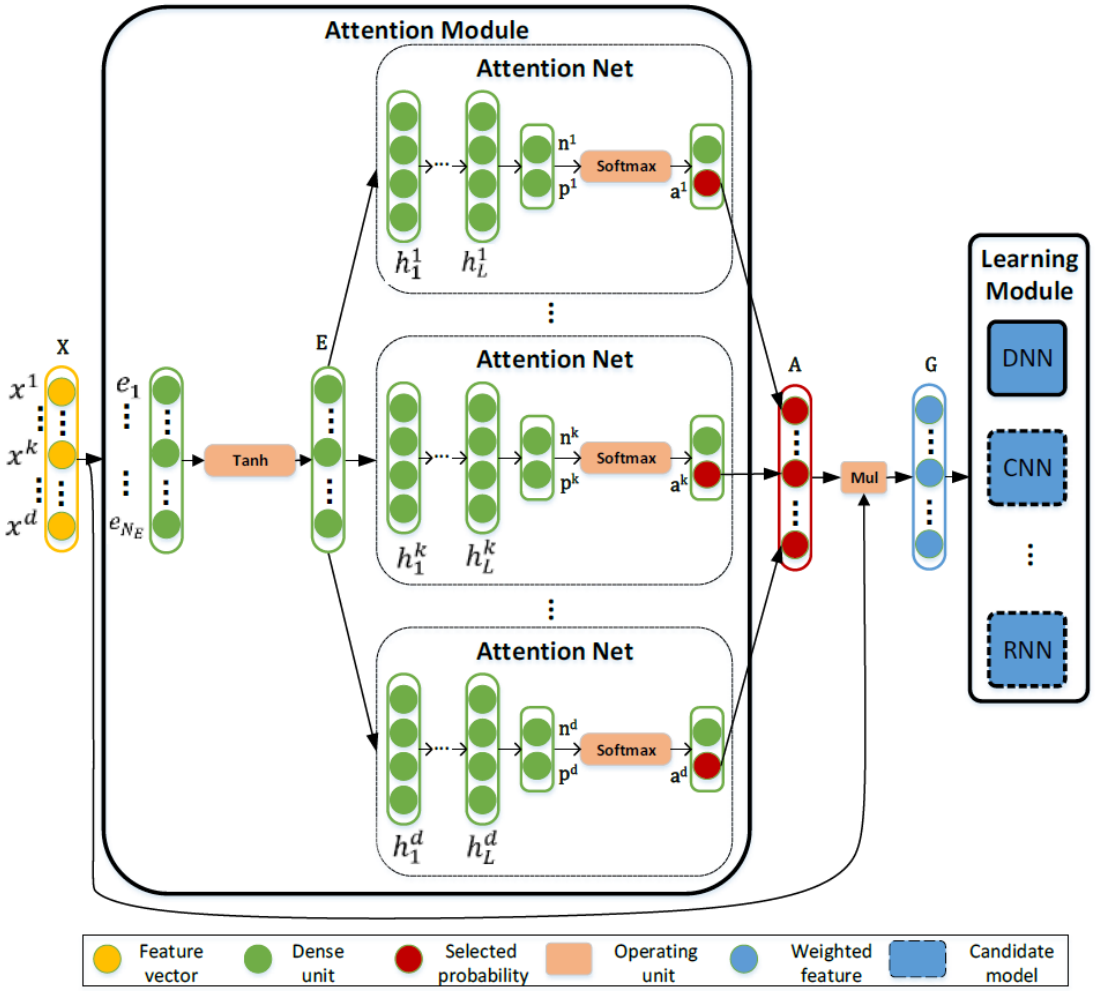

Figure 13 – Attention-based architecture for feature selection. Source: (GUI; GE; HU, 2019a)

In the paper, authors evaluate their technique using six datasets:

- MNIST: A widely used dataset for recognizing images hand-written digits (LECUN; CORTES; BURGES, 1998).
- n-MNIST-AWGN: MNIST variant with additive white gaussian noise (BASU et al., 2015).
- n-MNIST-MB: MNIST variant with motion blur (BASU et al., 2015).
- n-MNIST-RCAWGN: MNIST variant with reduced contrast and additive white gaussian noise (BASU et al., 2015).
- Lung_discrete: A dataset for lung cancer detection (LI et al., 2017).
- Isolet: A dataset for recognition of spoken letters (LI et al., 2017).

Results are compared with filter and embedded techniques, including Fisher score (HE; CAI; NIYOGI, 2005), ReliefF (KONONENKO, 1994), $l_{2,1} - norm$ minimization

through linear models (LIU; JI; YE, 2009), Random Forests and a DNN-based technique based on the concept of activation potential, reproduced according to the work in (ROY; MURTY; MOHAN, 2015). One way to evaluate and compare the different techniques is to obtain the feature importance values, sort them in descending order, create subsets using the top n features and obtain the performance value for each subset. These values are plotted to see how performance evolves as less important features are added to the model. However, the comparison is done visually, and it is shown in Figure 14. This is a qualitative comparison, meaning it is prone to interpretations. In the image below, performance is evaluated with the addition of more features. When compared to other techniques, the authors claim their proposed technique achieves better results with respect to accuracy and feature selection stability. It can be seen in the images that the curve related to the AFS technique presents high performance in relation to the other techniques, with the same number of features being evaluated. Intuitively and visually it could be assumed that the AFS technique is better than the rest. However, the values are close together and for some numbers of features the proposed technique does not reach the highest performance, being surpassed by random forest. Considering this, there is no way to determine if the proposed technique is better than the rest overall. Even if performance was better for every number of features, there is no way of knowing how much better one technique is when compared to another.

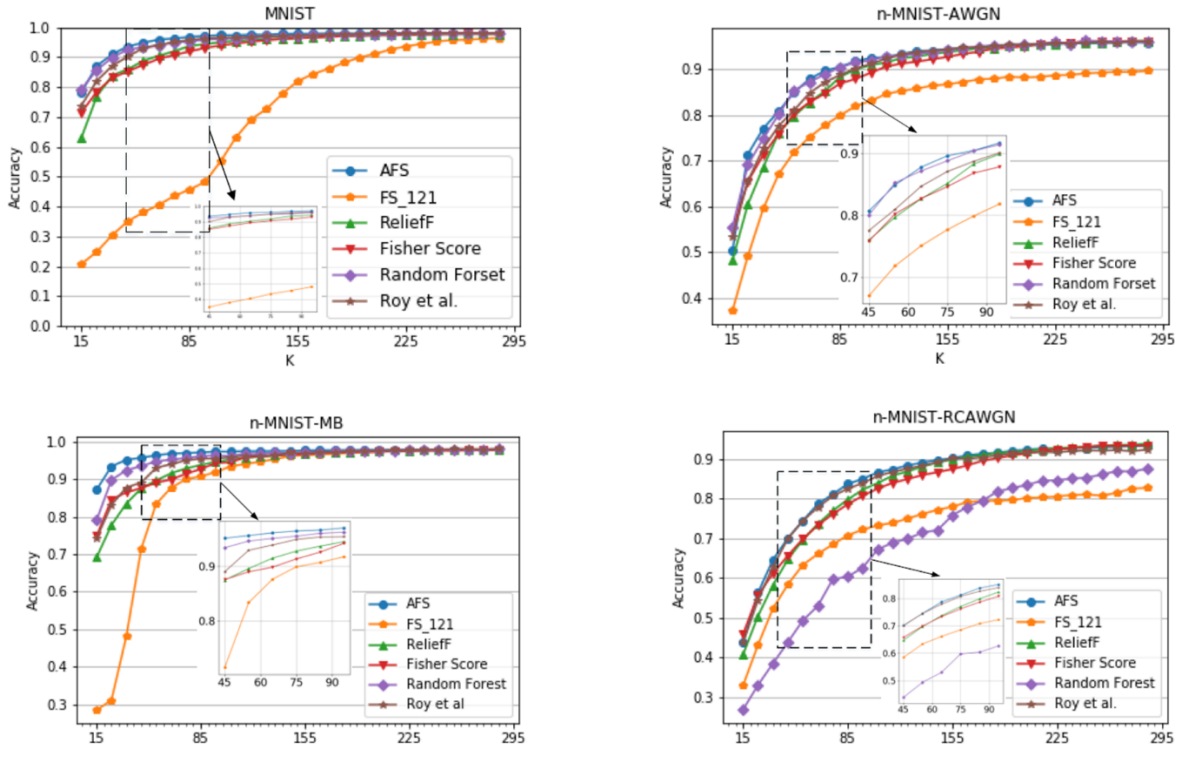

Figure 14 – Accuracy evolution for MNIST datasets. Source: (GUI; GE; HU, 2019a)

Despite the visual analysis of the results not being accurate, it is commonly used by researchers (CAI et al., 2018; CHANG; RAMPASEK; GOLDENBERG, 2017; NARDONE; CIARAMELLA; STAIANO, 2019; ŠKRLJ et al., 2020). This shows there is a need for an evaluation protocol in which feature selection techniques can be evaluated according to quantitative criteria.

Another aspect of the technique is the fact that its implementation comes with the addition of several hyperparameters. Some of them are

- Size of the first hidden layer
- Number of layers in the attention networks
- Size of each layer in the attention networks
- Activation function for each layer in the attention networks

Other hyperparameters regarding the attention module include dropout rates and regularization terms. Though is not strictly necessary for the attention modules to include dropout and/or regularization, these are techniques commonly used to avoid overfitting and, thus, increase performance. Their inclusion implies the fine-tuning of hyperparameters. Along with the rest of the hyperparameters mentioned before, the use of the AFS technique has the drawback of the construction of the attention module. The large amount of hyperparameters that must be tuned in order for the technique to be successful makes the process more time-demanding and arduous.

Another drawback of the AFS technique relates to the size of the model. The attention module includes an attention network for each input feature. Despite there is a hidden layer before the attention networks which compresses the information, the number of attention networks is the same as the original number of input features. For a large number of input features, a large number of attention networks must be added, each of them with a number of hidden layers and neurons per layer. This may increase the training time considerably, and could even limit the number of neurons and layers of the attention modules. As mentioned before, the inclusion of the attention module could affect the learning module. This is particularly notorious when the learning module has much fewer parameters than the attention module.

2.6.5. Drawbacks of Feature Importance Approaches

Though feature importance approaches to interpretability are relevant to unveil the dynamics of a model and explain its behavior, there are drawbacks. The main problem relates to the fact that the information given by these kind of approaches does not give much insights as to how the model generates an output. Thus, there is ambiguity towards the definition of importance.

2.7. Counterfactuals

The original concept of counterfactuals comes from the fields of philosophy and psychology, and refers to conditional statements in which the antecedent is false (GOODMAN, 1947). Thus, through the use of counterfactuals, an alternative scenario is imagined in which something is different from reality. This imaginative exercise is useful as an explanation. A common example used for explaining the concept of counterfactuals relates to bank loans aided by black box models (GRATH et al., 2018). An explanation for a rejected loan application presented as a counterfactual could be the following: “If the annual income had been $10,000 more than the actual income, the loan application would have been accepted”. While this kind of explanation does not give information about the dynamics of the model used for making the decision, it does relate inputs with outputs, which is useful for the end-user. Indeed, researchers have studied how do humans interact with counterfactuals, with positive results being achieved (BYRNE, 2007, 2019; KAHNEMAN; MILLER, 1986; WOODWARD, 2005). In (KAHNEMAN; MILLER, 1986), authors argue that humans constantly analyze events of reality by imagining counterfactual alternatives, and that we even create a notion of distance between counterfactuals and reality. Furthermore, these alternatives influence on our emotional response. In the example above, should the difference in annual income be $500 instead of $10,000, the end-user would naturally feel much more upset than in the original case, despite the outcome being the same. In (BYRNE, 2007), the author states that counterfactual thought are triggered constantly by humans, and that they follow similar rules to rational thoughts, meaning that humans use imagination to understand the rationality of the environment. Thus, the idea of counterfactuals is familiar to humans, and using them for explaining black-box models is a suitable option.

While there is a variety of scenarios that may serve as counterfactuals, the ones that create a better explanation are those closer to reality. Following the example above, a difference of

$1,000,000 in income would be a valid counterfactual, as the loan would be accepted. However, that information is not very useful to the client, since that is not the minimal distance from a loan rejection scenario to an accepted loan scenario. The original counterfactual, being the minimal value for the loan to be accepted, gives him/her a better notion of the decision boundary, and thus, creates a better sense of understanding of the reasons why the loan was not accepted.

In the context of machine learning, counterfactual explanations are alternative input values that are close to the original but change the output's class. In the work by Verma et. al. (VERMA; DICKERSON; HINES, 2020), authors describe several other desirable properties of counterfactuals, such as:

- Actionability: There are features that due to problem constraints should not be modified. A typical example is a person's age when requesting a loan to a bank. In this case, a counterfactual in which the age of the person is lower than the actual value is not actionable.
- Sparsity: A counterfactual should not only be close to the original value, but also should alter the value of a few number of features. This is directly related to interpretability, as humans are only capable of understanding phenomena involving only a limited number of variables (HALFORD et al., 2005).
- Data manifold closeness: The proposed counterfactual should be close to the training data manifold in order to be more realistic and trustworthy.
- Treatment of causal relation between input features: Input features may be correlated through causality with each other. Thus, the variation of one feature may alter the value of a correlated feature. In this sense, a counterfactual should acknowledge and maintain these causal relations.

2.7.1. Counterfactual Generation as a Minimization Problem

In the work by Wachter et. al. (WACHTER; MITTELSTADT; RUSSELL, 2017), counterfactuals are introduced as an optimization problem in which the distance between the original input value and the counterfactual is minimized, given that the counterfactual

generates the desired output variation. Thus, the counterfactual value is found by minimizing the following expression:

$$arg\min_{x'}\max_{\lambda}\lambda(f(x')-y')^2+d(x,x') \tag{30}$$

, where x' is the counterfactual value, x is the original input value, y' is the desired output class by the counterfactual, $f()$ is the trained model and $d(\cdot,\cdot)$ is a distance measure. λ is a value iteratively maximized to find a valid counterfactual value that generates an output value close enough to the desired output. Thus, according to this definition, a valid counterfactual is an input that not only changes the class output, but also is a value close to the original input value. Counterfactuals are limited to classification tasks, since they need class-like outputs to evaluate the validity of counterfactuals.

To enhance certain properties of counterfactuals, the expression to be minimized can be altered. For example, to enhance actionability, sparsity and data manifold closeness, the minimization problem is modified to the following expression:

$$arg\min_{x'\in\mathcal{A}}\lambda(f(x')-y')^2+d(x,x')+g(x-x')+l(x';\mathcal{X}) \tag{31}$$

, where $\mathcal{A}$ is the set of actionable features, $g()$ is a penalty function that encourages sparsity, and $l(x';\mathcal{X})$ is a function that measures the closeness of the counterfactual to the training data manifold.

Regarding the distance measure $d(\cdot,\cdot)$, several works have used the following metric (DANDL et al., 2020; MOTHILAL; SHARMA; TAN, 2020; RUSSELL, 2019; WACHTER; MITTELSTADT; RUSSELL, 2017):

$$d(x,x')=\sum_{j=1}^{F}\frac{\left|x_j-x_j'\right|}{MAD_j} \tag{32}$$

$$MAD_j=median_{i=1}^{N}\left(\left|x_{i,j}-median_{k=1}^{N}(x_{k,j})\right|\right) \tag{33}$$

, where F is the number of features and N is the number of training samples. MAD is the median absolute deviation. Regarding the penalty function for sparsity $g()$, typically L1 and L2 norms are used. However, it has been well analyzed that L1 norm is prone to produce more sparsity in the results than the L2 norm (NG, 2004). Although not formally a norm, the L0 norm has been also considered (VERMA; DICKERSON; HINES, 2020). Finally, $l(x'; \mathcal{X})$ is the term related to the data manifold closeness. However, this could hard to implement directly in a neural network ad-hoc approach. Some authors have considered other properties for counterfactuals besides the four described in this section, such as diversity (MOTHILAL; SHARMA; TAN, 2020; RUSSELL, 2019). This relates to the generation of not only one but a set of plausible counterfactuals. This is useful to give the end user a variety of actions he or she can take in order to achieve the desired goal. From an interpretability perspective, this is useful to differentiate the data manifold of each class.

Regarding the optimization of equation (30), the authors in (WACHTER; MITTELSTADT; RUSSELL, 2017) use the ADAM optimizer (KINGMA; BA, 2015). Since the machine learning model represented by $f()$ is already trained, this is a post-hoc approach. Research has shown that post-hoc explanations have inconsistency issues, as different techniques may yield different explanations for the same situation (BORDT et al., 2022; KOMMIYA MOTHILAL et al., 2021). Thus, the importance of creating interpretable models rather than using post-hoc methods to explain black-box models is highlighted.

2.7.2. Counterfactuals Guided by Prototypes

In (VAN LOOVEREN; KLAISE, 2021), the authors present an approach to creating counterfactuals with the help of prototypes for each class. They argue that this helps creating more interpretable counterfactuals in less time when compared to the previous technique. To create the class prototypes, there are two alternatives: the use of k-d trees (BENTLEY, 1975) or autoencoders. Furthermore, they use elastic net regularization (ZOU; HASTIE, 2005) to enhance sparsity. When using k-d trees, the counterfactual generation process is the following:

1. Extract a sample from the training set and label each datapoint according to the prediction function.

2. Build one k-d tree per class using the extracted sample.
3. Find the nearest point to the one to be explained but different from the original class, by minimizing the Euclidean distance. This search determines the prototype to be used.
4. Optimize the objective function:

$$L_{pred} + L_1 + L_2 + L_{proto} \quad (34)$$

, where L_{pred} is the prediction loss, L_1 and L_2 are regularization terms and L_{proto} is the distance from the counterfactual to its prototype. The objective function is optimized using a fast iterative shrinkage-thresholding algorithm (FISTA) (BECK; TEBOULLE, 2009).

With this methodology, the authors show that counterfactuals are more interpretable, both quantitatively and visually (for image-type data). Furthermore, they find that the use of a prototype reduces the number of gradients needed to reach a solution by at least 75%. Also, they can induce the generation of actionable counterfactuals by fixing some of the feature values according to the problem's restrictions.

2.8. Counterfactuals, Causality and Machine Learning

While we as humans have a notion of what causality is, it is not simple to put that notion into a formal definition. Causality (or causation) broadly refers to the influence of one event in another event. It has been studied thousands of years ago. An example is the concept of karma, born before c. 1500 BCE (DONIGER; O'FLAHERTY, 1980). This shows that the concept of causality has been present in our way of viewing the world.

As mentioned in the Introduction section, counterfactuals are closely related to causation. In humans, counterfactual explanations lead to causal understanding (VERMA; DICKERSON; HINES, 2020). In the example above, the counterfactual explanation creates a causal understanding in the client. By knowing that the income should be $10,000 higher in order for the loan to be approved, the end-user understands that the cause of the rejection is in the income variable, and not in any other variables used for evaluation.

Among the different approaches to causation, David Lewis elaborates a theory related to counterfactuals (LEWIS, 1974). In it, he states that two events are causally dependent of each other if the two occur and if one of them does not occur, the other also does not occur. Thus, he defines causation in terms of counterfactuals.

Other theories of causation use counterfactuals as a relevant concept as well. Judea Pearl describes the logic of causal reasoning through three hierarchical levels, referred to as "ladder of causation" (PEARL; MACKENZIE, 2018):

1. Association: Refers to the act of observing the environment without any interaction. The observation of the environment is used to find correlations between variables. Pearl argues that humans and many other animals have this ability.
2. Intervention: Is the analysis of the environment when a variable is deliberately modified. Only few species have the ability to understand the consequences on their actions on the environment. While some events do not rely on an intervention to occur, the difference between this and the previous level is, through intervention, there is certainty of the variables being altered. In association, variables may be altered due to other variables.
3. Counterfactuals: This is the higher level in the ladder. It is the analysis of events through the imagination of alternative scenarios. Pearl argues that conclusions about causality can be achieved through the answering of these "what if" queries.

According to Judea Pearl, machine learning algorithms belong in the first rung. They are algorithms that observe the environment through data and draw conclusions without interacting with it. Though association is necessary for drawing causal conclusions, it is not sufficient. Interacting with the environment through interventions and imagining alternative scenarios is necessary for drawing causal conclusions. Thus, typical machine learning algorithms (such as the ones described previously) do not capture causal relations, but they do give information about correlations. In (PEARL, 2019), Pearl argues that machine learning algorithms cannot formulate nor answer causal queries since they are based only in statistical tools. However, he proposed the use of seven tools that can overcome this issue in machine learning by climbing to levels 2 and 3 in the ladder of causation:

1. Encoding causal assumptions: Transparency and testability: Algorithms should encode causal assumptions between variables in a way that results can be analyzed and understood. Pearl argues that graphical models address this.
2. Do-calculus and the control of confounding: Confounders are hidden causes of two or more variables. This is one of the main reasons why correlation does not imply causation. To address this, do-calculus is an operator that estimates the effects of interventions whenever possible. Thus, it is known that the effect comes from the intervention and not a confounder. However, intervention data is needed for do-calculus to be available.
3. Algorithmization of counterfactuals: Tools for systematic analysis of counterfactuals are essential for understanding causation.
4. Mediation analysis and the assessment of direct and indirect effects: Some cause and effects relations are indirect, in that other variables act as mediators. The assessment of these situations is important to deal with causality.
5. Adaptability, external validity and sample selection bias: Through the level of association, machine learning algorithms are not adaptable to different domains or when environment conditions change. Through causal inference, machine learning algorithms can become more adaptable and robust.
6. Recovering from missing data: By knowing cause and effect relations between variables, it will be possible to do better estimates for missing data than, for example, interpolation-based approaches.
7. Causal discovery: While encoding known causal relations is crucial for causal inference, discovery of unknown causal relations is important for modifying the original causal model, improve performance, an even learn from the model's discoveries.

Machine learning algorithms such as neural networks do not include these tools. However, interpretability is a useful approach for users to understand the dynamics of a machine learning model and to take the most information out of it in order to draw more conclusions in the association level of Pearl's ladder, which in turn will be useful for the next two levels. While machine learning algorithms cannot be used to draw causal conclusions, there is a potential through interpretability to get insights that could lead to information about causality, thus, closing the gap between the first level of the ladder and the other two.

2.8.1. Necessity and Sufficiency

Necessity and sufficiency are two concepts used to better understand causality. A necessary cause is an event that, should it not occur, the effect would never occur. On the other hand, a sufficient cause is an event that its occurrence will always lead to the occurrence of the effect (SEAWRIGHT, 2002). Thus, an event can have a combination of these two causal attributes to determine its connection to the alleged effect. For example, an event can be a necessary cause but not sufficient, if another event must occur jointly in order to the effect event to occur. On the other hand, an event can be a sufficient cause but not necessary if other events also determine the occurrence of the effect.

In (PEARL, 1999), Pearl defines the probabilities of causation of a variable X with relation to a variable Y as a set of three probabilities: probability of necessity ($PN(X_Y)$), probability of sufficiency ($PS(X_Y)$) and probability of necessity and sufficiency ($PNS(X_Y)$):

$$PN(X_Y) = P\left(y'_{x'} \mid x, y\right) \tag{35}$$

$$PS(X_Y) = P(y_x \mid y', x') \tag{36}$$

$$PNS(X_Y) = P(y_x, y'_{x'}) \tag{37}$$

, where x and y are specific values for binary variables X and Y respectively. Here, x stands for $X = true$ and y for $Y = true$, and x' and y' are their complements. Furthermore, the notation $y'_{x'}$ stands for "Y would have been false if X would had been false", and y_x stands for "Y would have been true if X would had been true". Thus, PN is the probability that Y would have been false if X would have been false, given that X and Y are actually true. PS is the probability that Y would had been true if X would had been true, given that actually X and Y are false. PNS is the probability of Y being true if X is true and being false otherwise. According to the author, the three values are related through the following equation:

$$PNS = P(x, y) \cdot PN + P(x', y') \cdot PS \tag{38}$$

To calculate the values mentioned above, the seven tools presented by Pearl are necessary. Particularly, the algorithmization of counterfactuals. Without an appropriate way to generate counterfactuals, probabilities of causation cannot be calculated. In (KOMMIYA MOTHILAL et al., 2021), authors operationalize the calculation of necessity and sufficiency in the context of machine learning for determining necessity and sufficiency of each input feature. By using existing counterfactual generation techniques, they calculate each of the input features' necessity and sufficiency for explaining an output value. In the case of necessity, they generate counterfactuals by only modifying the value of the feature of interest and measure the proportion of valid counterfactuals, and present it as a percentage value. In the case of sufficiency, they use an opposite approach and generate counterfactuals that modify the values of all input features but the one of interest. Sufficiency is then quantified as the fraction of non-valid counterfactuals generated. By evaluating the whole test set and for all input features, they quantify necessity and sufficiency for each of the model's input features. Furthermore, they argue that more necessary and/or sufficient features are more important to the model. Thus, this allows the creation of feature rankings based on necessity and sufficiency.

3. PROPOSED FRAMEWORK

In this section, the proposed frameworks for interpretability of neural networks for PHM are presented. The section is divided according to the three frameworks: Feature selection using deep neural networks, counterfactual generation for interpretable fault diagnosis, and combining feature selection and counterfactual generation.

3.1. Feature Selection Using Deep Neural Networks

In this subsection, the framework for feature selection in deep neural networks is presented. It is subdivided in four parts: feature selection layer, ranking quality score, model selection and comparison with other techniques.

3.1.1. Feature Selection Layer

The feature importance analysis in DNNs is addressed by adding a feature selection layer (FS) after the input layer, in which each feature is multiplied by a trainable weight with values constrained to the range of [0,1]. These weights are trained jointly with the rest of the network's weights and represent each feature's importance. A representation of this approach and how it is embedded into the neural network is shown in Figure 15. Since the objective is to build a framework in which no preprocessing is needed, it was assumed there is no prior information about the features' importance values. Thus, the weights in the FS layer are initialized as $1/n$, where n is the number of features. Also, a regularization term is added to ensure the interaction between features, which is calculated as:

$$r(W^{FS}) = \lambda \cdot \left| \sum_{i=1}^{n} w_i^{FS} - 1 \right| \tag{39}$$

, where W^{FID} represents the weights vector in the FS layer and λ is a value between 0 and 1 that determines the strength of the regularization. The addition of this regularization term to the loss function enforces the weights to sum 1. Thus, it is assured that the importance of each feature is influenced by the interaction with other features, which does not occur in methods like random forest or mutual information. This also avoids situations in which

weights have the same values. Furthermore, the use of the l_1 norm, instead of the l_2 norm is based on the effects of the L^1 and L^2 regularizations in the model related to feature selection, as detailed in the previous section. This configuration enforces the model to reach sparse solutions, in which irrelevant features are tied to a weight value of zero. The use of the l_2 would result in irrelevant features having values close to zero.

The inclusion of an FS layer to a DNN adds two hyperparameters to the network: the layer's activation function and the value of the λ parameter presented above. Both of them must be tuned to reach the highest performance possible, and in unison with the hyperparameters in the network.

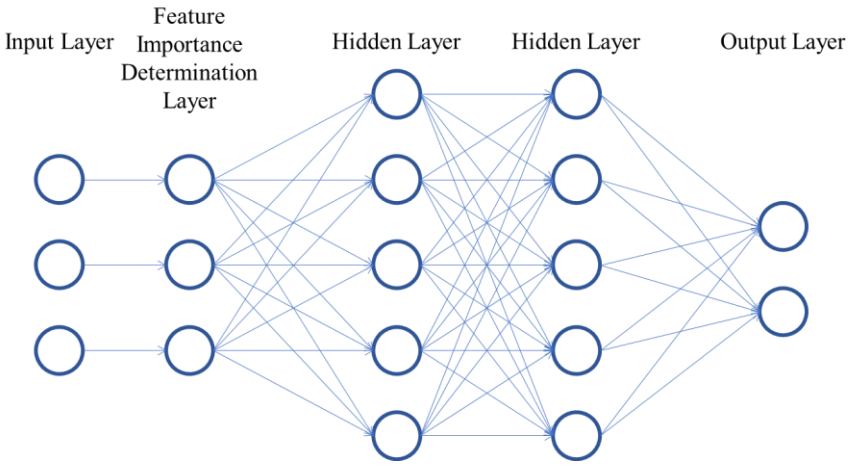

Figure 15 – An example of a deep neural network with the proposed FS layer.

According to the taxonomy shown in Figure 9, the presented technique corresponds to an ad-hoc interpretability modelling technique. Furthermore, it has elements of interpretable representation techniques, such as regularization, and model renovation techniques, such as the addition of a layer with the purpose of calculating feature importance. Within feature importance, this is an embedded technique, as it is a product of the network's configuration. Its application implies the modification of the network's architecture. However, the number of parameters added the model is equal to the number of input features. In most DL applications, this number is small in comparison to the total number of trainable parameters. This means that the application of the FS layer technique has a low probability of altering the results obtained through the original architecture. However, this must be demonstrated.

3.1.2. Ranking Quality Score (RQS) for Ranking Evaluation

The addition of an FS layer to the network enables the creation of a ranking of input features in terms of how relevant they are. This is useful for visualizing the tradeoff between number of input features and performance, which is key to determining the number of features used in deployment. Here, it is proposed a framework for a quantitative evaluation of the obtained ranking, which helps on deciding which model is the most suitable, and what features of the input feature space shall be used in the model deployment. This framework is presented in Figure 16. It consists of the following steps.

1. The raw dataset is preprocessed to make it suitable for usage. Since feature selection will be done during training, it is not necessary to do any analysis for removing variables, besides discarding categorical variables.
2. The dataset is divided into train and test sets and are normalized. To do this, the scaler chosen to normalize the data is fitted to the train set and used in both the training and testing set. For example, if the data is standardized, the mean and variance used for normalizing both the training and test sets are obtained only with the train set. This is done in order not to have any interference in the training process from the test set.
3. The training set is used for training the network.
4. When the training process is finished, the FS layer's weights are used to rank the features in decreasing order. The model's performance is then evaluated in the test set using a subset of the top n features, with $n = \{1,2, \dots, N\}$.At first, the subset only contains the most relevant input feature. After evaluation, the second most relevant feature is added to the subset and performance is evaluated. This process is repeated until all the input features available are added to the subset.

The performance results obtained from these iterative process are presented in a curve in order to compare with other methods, as in other works (CHANG; RAMPASEK; GOLDENBERG, 2017; GUI; GE; HU, 2019a; ŠKRLJ et al., 2020). However, a visual comparison is not as accurate and reliable as a quantitative comparison. To address this issue, it is proposed the ranking quality score (RQS), defined as:

$$RQS \equiv \frac{\sum_{n=1}^{N} PM_n \cdot n}{\sum_{n=1}^{N} \max(PM) \cdot n} = \frac{2 \cdot \sum_{n=1}^{N} PM_n \cdot n}{\max(PM) \cdot N(N+1)} \quad (40)$$

, where PM_n is a performance metric of the choice evaluated using the n most relevant features and $\max(PM)$ is the maximum value reached by the model. The range of values of this metric is [0,1] (the proof is presented in Appendix A) and it compares the two scenarios exemplified in Figure 17. In the figure, the scenario shown in the blue line represents an example case where the performance metric value increases alongside the number of features. Then, the numerator of equation 40 is calculated by multiplying each performance metric (calculated through step 4 mentioned above) by the number of features used to reach that performance value and summing up all of the results. This value is an indicator of the quality of the ranking obtained through step 4. However, it is highly dependent on the model. Within one model, this value can be used successfully to compare different rankings. Nonetheless, this cannot be done when comparing models. To address this issue, the RQS metric has a normalization term shown in the denominator of equation (3), represented by a red dotted line in Figure 6. It is the same calculation of the numerator, but assuming an ideal case where the maximum performance value is reached with the most relevant feature and is unaltered by the addition of less relevant features. By using the denominator mentioned above, the RQS metric presents values between 0 and 1. To illustrate its behavior, Figure 18 shows six different curves with their corresponding RQS value.

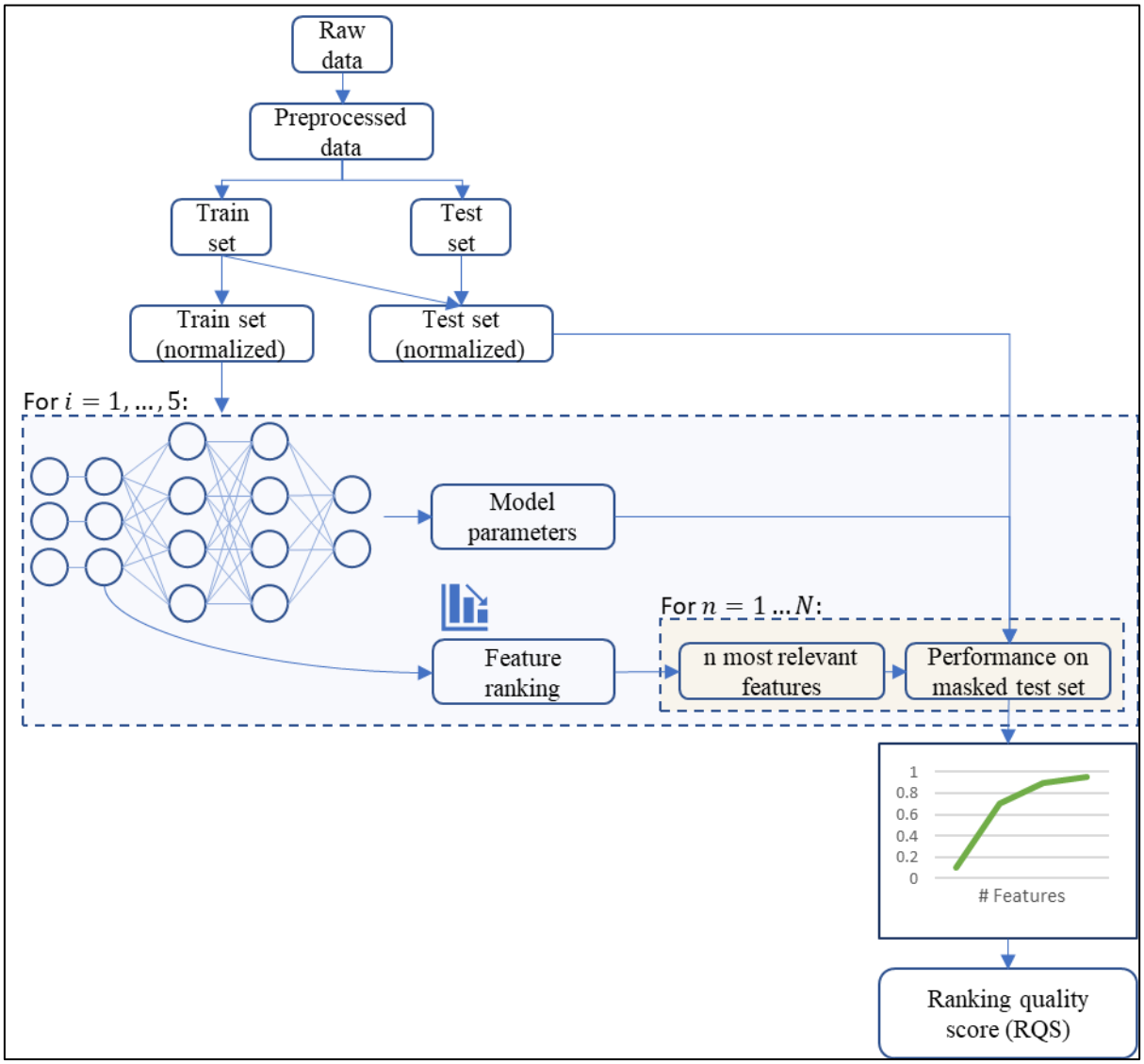

Figure 16 - Proposed methodology for ranking quality evaluation.

The RQS metric measures the quality of the ranking obtained through the model using an FS layer. It is a normalized weighted sum of the chosen performance metric. It gives more weight as the number of variables increases. When using one feature, performance typically shows a low value. As features are being added, performance increases at a fast rate. After a certain number of features, performance reaches a plateau. The RQS metric gives a high value to a model in which the plateau is reached with few variables and this plateau is not affected by the addition of more features. On the other hand, it gives a low value to a model in which more features are needed to reach the plateau, or the addition of features leads to an important decrease in performance.

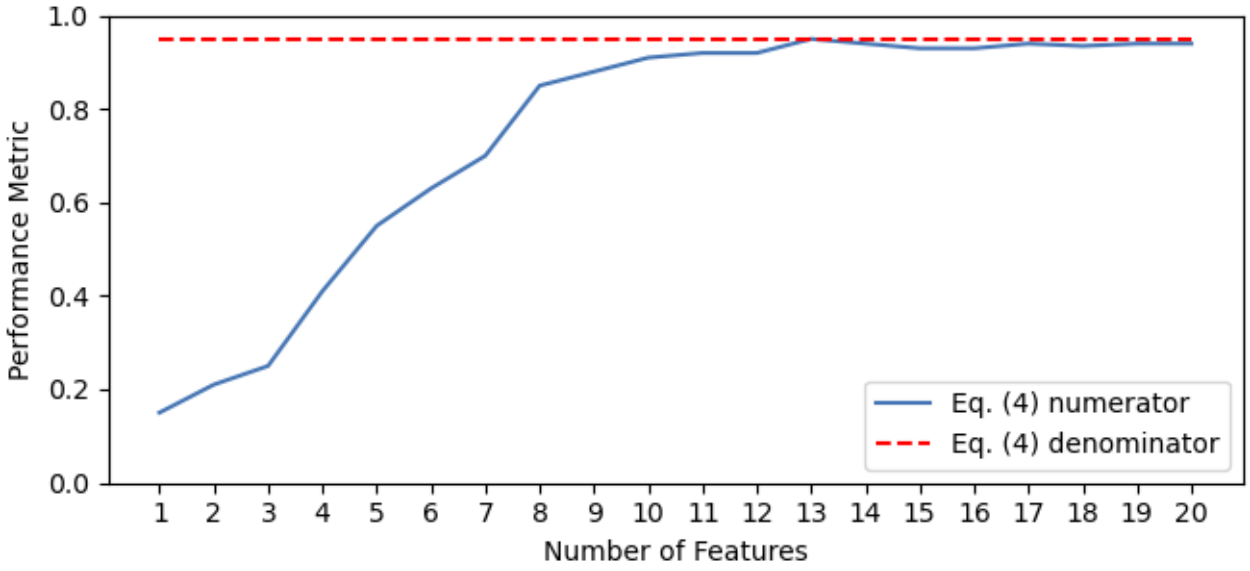

Figure 17 - Graphical representation of equation (40).

Besides achieving a high RQS, the proposed technique must, at least, maintain the same level of performance compared to the same network without an FS layer. Otherwise, there would be a tradeoff between performance and interpretability, which is not desirable. As mentioned before, it is expected that the use of the FS layer technique does not affect greatly the inner dynamics of the rest of the network, and thus, alter the results. To evaluate the influence of the FS layer in the model's performance, and based in the work presented in (ŠKRLJ et al., 2020), relative performance metrics are used, as defined in:

$$PM_n = \begin{cases} \rho_n / \rho_0, & for\ classification\ tasks\ or\ PM_n = R^2 \\ \rho_0 / \rho_n, & for\ regression\ tasks\ with\ PM_n \neq R^2 \end{cases} \quad (41)$$

, where ρ_n is the performance metric of the model trained with the FS layer and tested with the n more relevant features, and ρ_0 is the performance metric of the same model but without the FS layer and tested with all the set of features. Since typical regression performance metrics (except for the R^2 coefficient) show the same behavior as the prediction error, the difference in definition is necessary for having the same kind of analysis regardless the task. Performance metrics are used as in equation (41) to easily illustrate in this work if the inclusion of an FS layer is beneficial or detrimental to the model's performance. A value >1 indicates the use of an FS leads to an increase in the model's performance, whereas a value

≤1 indicates the contrary. Despite the fact that this work uses relative performance metrics, the proposed framework allows the use of any type of performance metric, relative or absolute. The objective of using relative performance metrics in this work is to show how the FS layer technique affects performance, compared to a model where no FS layer is used.

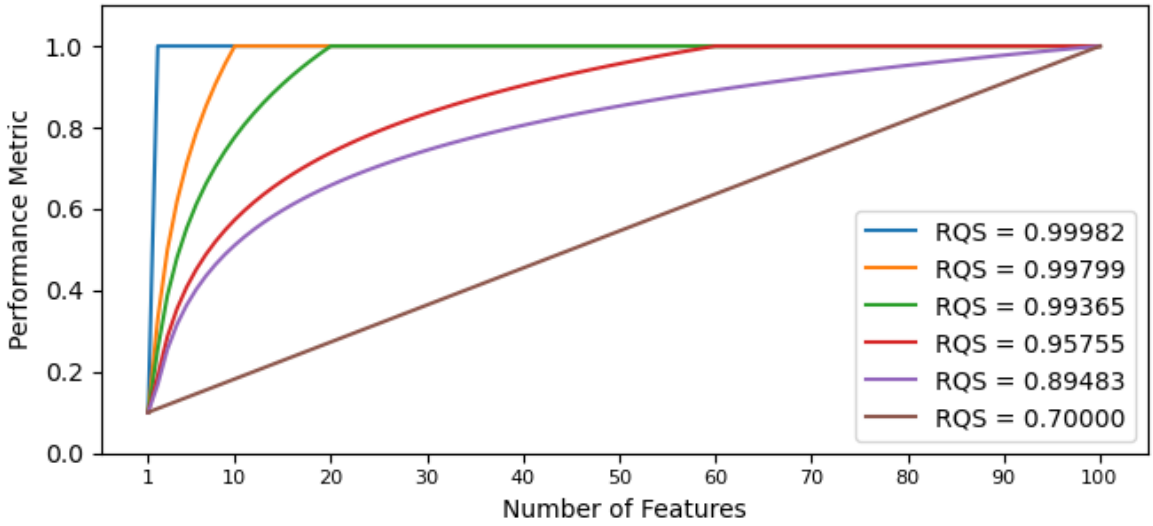

Figure 18 - Different ranking situations with their corresponding RQS values.

The proposed methodology involves the evaluation of models in terms of task performance (through a chosen performance metric) and their feature ranking quality (through the proposed RQS metric) simultaneously. Since the RQS metric quantifies how the model reaches its maximum performance and not the performance itself, this simultaneous analysis is not only possible but necessary. This is more clearly explained by observing that a model may have a good ranking quality (reflected in a high RQS value) but a low task performance. Thus, the selection of an appropriate model is determined by its task performance and its RQS.

3.1.3. Model Selection

The objective of using an FS layer and the RQS metric is to improve the model selection process for DNNs. The RQS metric compares different approaches for feature selection. Thus, a model with a higher RQS value should be preferred over one with a lower RQS value when performance is similar between them. After choosing the appropriate

technique, the number of features used in deployment must be determined. To do this, a mathematical-based criterion would be the number of features that yields the maximum performance value. While this is a reasonable criterion, it does not consider the fact that some features may have little impact on performance. In a real-life application, maximum performance may not be the ultimate objective, but rather a performance level above a predefined threshold, subject to limitations such as costs or time, among others. Moreover, models with a high RQS might benefit from other selection criteria besides maximum performance since they achieve high levels of performance with fewer features.

In this work, two criteria for model selection are analyzed. The first one is maximum performance, and the second one is defined as the number of features needed to reach a 95% of the maximum performance value. This is done in order to show the performance and applicability of the techniques in a scenario where maximum performance is required and another one where a 5% decrease in performance is allowed, emulating a real-case scenario. However, the application of this framework is not restricted to these two criteria. Other criteria can be used for model selection depending on particular objectives and limitations. With this, model selection is done not only evaluating a performance value but also the RQS value simultaneously.

3.1.4. Comparison with other Techniques

To compare the proposed technique against other approached in terms of RQS and task performance, the following techniques were tested:

- Mutual Information
- ReliefF (KONONENKO, 1994)
- Random Forest
- “AFS: An Attention-based mechanism for Supervised Feature Selection” (GUI; GE; HU, 2019a) (AFS)

In the first method (mutual information), the feature importance is determined according to the dependency of each feature with the output feature, as calculated in equation 23. Thus, features that are more dependent with the output are more relevant. For entropy

estimation in this thesis, the number of nearest neighbors is set to three, as it presents a balance between variance and bias (ROSS, 2014). In ReliefF, features are analyzed in terms of how well they can distinguish classes in instances close to each other. The feature weights are calculated by measuring how the feature value varies with respect to a *nearest hit* (the nearest instance with the same class) and with respect to *nearest misses* (nearest instances with different classes). This is repeated for a set of instances. For regression tasks, ReliefF algorithm is extended to RReliefF (ROBNIK; KONENKO, 2003). Like in the mutual information technique, the number of nearest neighbors used is set to three. In the case of Random Forest, feature importance is calculated during training based on how much each feature contributes to the decrease in Gini impurity for classification tasks and the variance of the result for regression tasks. When using random forests, the number of trees is set to 200, as it reaches a balance between computation time and performance. The AFS technique, as described in the Theoretical Background section, presents a detachable module for feature selection based on attention mechanism. An attention net for each feature is trained jointly with the rest of the network to determine whether the feature is relevant or not. The output of each attention net after training is used to rank features.

These four techniques cover filter and embedded methods. Wrapper methods are not used since it becomes unfeasible for some datasets. Mutual Information and ReliefF techniques are filter methods, thus, they are model agnostic. On the other hand, Random Forest is a popular ML technique which has an embedded method for feature selection. The AFS method is used for comparison because it is a technique embedded in neural networks, achieving promising results when compared against other filter and embedded techniques, including one used for neural networks as well (ROY; MURTY; MOHAN, 2015). To the best of author's knowledge, it is the most promising technique amongst those embedded in neural networks. In this thesis, two different configurations of this model are used to compare with the proposed framework, namely AFS and AFS 2. In the first one (AFS), the first hidden layer after the input layer (referred to in the paper as E) has an $N/2$ neurons, N being the number of input features. A graphical representation of an example (with $N = 4$) is shown in Figure 19. In AFS 2, the same layer has N neurons. Regarding the attention nets, it was decided not to include any hidden layers (referred to in their work as $\{h_1^k \dots h_L^k\}$) for three reasons: First, the authors in (GUI; GE; HU, 2019a) do not specify the number of hidden layers and neurons used in their experiments; second, the implementation available in (GUI; GE; HU, 2019b) does not include hidden layers in their attention nets; and last, depending

on the number of features, the inclusion of hidden layers would increase the model size and, thus, its computational cost. Therefore, the E layer is connected to N softmax-activated layers with two neurons, which is used to determine the importance of each feature before entering the learning module.

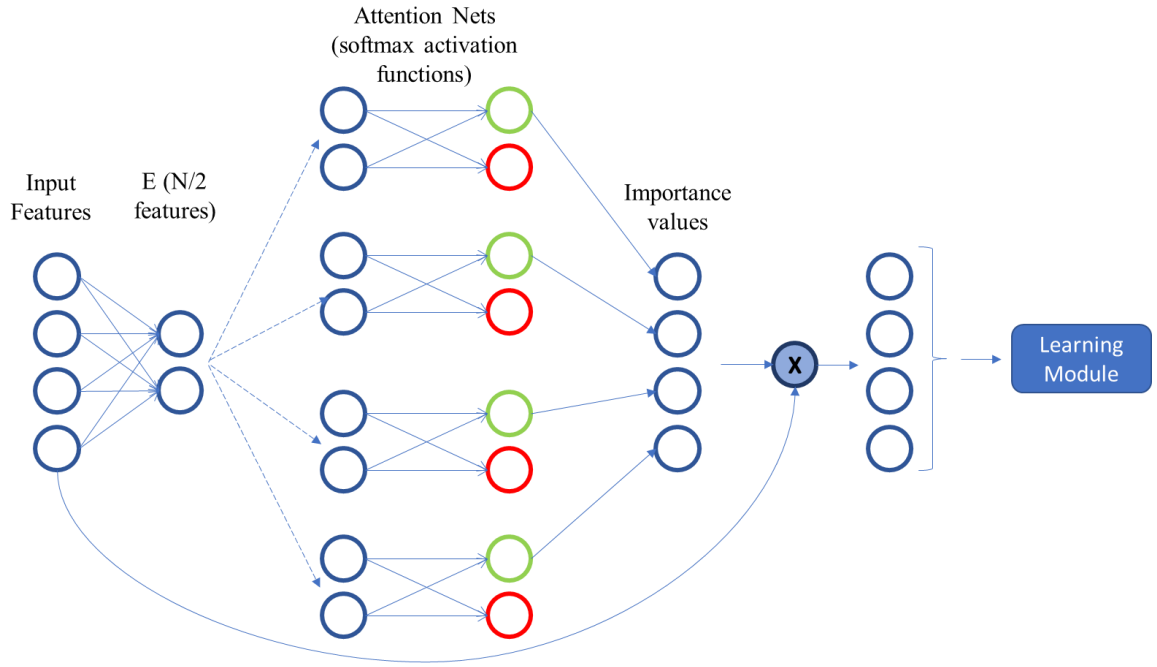

Figure 19 - Example representation of the AFS network. Note that the E layer has $\boldsymbol{N/2}$ neurons. In the AFS2 configuration, the E layer has $\boldsymbol{N}$ features. As described in the original work, the E layer has a $\boldsymbol{tanh}$ activation function. It serves as input to $\boldsymbol{N}$ attention nets, each of them used to determine the importance value of an input feature. Adapted from (GUI; GE; HU, 2019a)

Regarding the two widely used techniques LIME and SHAP discussed in the Introduction section, we do not use them for comparison because their mechanism for interpretability uses the model's predictions. They are post-hoc techniques in which explanations depend entirely on the model's outputs, and therefore, their inputs. In the case of LIME, it is a method for local interpretability, while in the case of the SHAP algorithm, feature importance for the model is obtained through the mean Shapley values for each feature across the dataset. This means that they may eventually vary when evaluating them with other datasets (such as using the test set). In contrast, all of the approaches above estimate feature importance values that do not change when varying the test set.

To evaluate the ranking quality of the aforementioned methods, the first step is to obtain the features' importance values using each of the aforementioned techniques. Then, a neural network with the same configuration as the main model but without the FS layer is trained. Then, similarly to the process depicted in Figure 16, the n most relevant features are

used to evaluate the network's performance on the masked test set, with n varying from 1 to the total number of features N. The network's performance evolution across the number of features is represented in a curve, and RQS is calculated, as discussed, the in the following section.

3.2. SCF-Net: A Sparse Counterfactual Generation Network for Interpretable Fault Diagnosis

To achieve interpretability of neural networks-based fault diagnosis using counterfactuals, a multi-task architecture is proposed. Thus, a trained model will classify the input variables according to their health state and simultaneously generate a counterfactual according to a desired class. The architecture for SCF-Net is presented in Figure 20. As shown in the figure, the SCF-Net receives three vectors as inputs: a vector X with all the input features' values, a random latent variable (RLV) vector which allows the generation of multiple counterfactual values, and a desired class (DC) vector that indicates the objective class of the generated counterfactual. Thus, the SCF-Net receives a $[X, RLV, DC]$ vector as input. This configuration is inspired in InfoGANs (CHEN et al., 2016), where the generator receives a random point in space and a class label as inputs. The proposed architecture uses these three inputs in two tasks: the first one is a classification network that determines which class the X vector belongs to. The second one is a counterfactual generator that uses the three inputs to generate a counterfactual for X according to the desired class DC. The generated counterfactual X' then uses the weights of the classification network to generate an output class Y'. This is done in order to enhance the generator network to generate a counterfactual value that creates the desired output change when passing through the classification network. The DC value is necessary in order to specify which class the counterfactual must belong to. While this might not be necessary in a binary classification problem, it is necessary in the context of multi-class classification. Furthermore, the RLV value addresses the need for diversity in counterfactual generation algorithms, as mentioned in the previous section. Thus, a different counterfactual will be generated for each forward pass on the same input value.

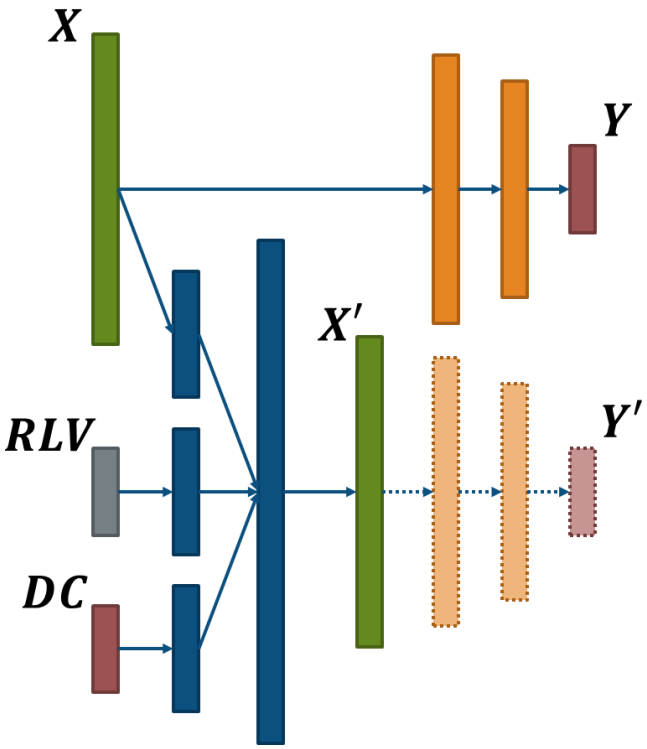

Figure 20 – SCF-Net schematic representation.

In order for a SCF-Net to succeed at both tasks simultaneously, appropriate loss functions must be defined. This work proposes the use of a loss function composed of four terms:

$$loss = -\sum_{j=1}^{C} y_j \, log \, \hat{y}_j + \sum_{i=1}^{F} \frac{|X_i - X'_i|}{MAD(X_i)} + \|X - X'\|_0 + -\sum_{j=1}^{C} y'_j \, log \, \hat{y}'_j \tag{42}$$

, where C is the number of classes, F is the number of features, y is the network's output value, $\hat{y}$ is the correct label, X is the input value, X' is the counterfactual value generated by the network, $MAD()$ is the median absolute deviation function described in equation 4, and $\hat{y}'$ is the correct counterfactual label. The first term corresponds to the categorical cross-entropy loss function, commonly used for classification problems. The second term is used in order to minimize the distance between the original input value and the generated counterfactual. According to Wachter et. al. (WACHTER; MITTELSTADT; RUSSELL, 2017), the feature-wise $MAD()$ function is used instead of the standard deviation in order to enhance robustness against outliers. Furthermore, they test the use of both L1 and L2 norms,

concluding that the former is more suitable as it generates sparser results. The third term's purpose is to enhance sparsity, as it gives penalty each time the feature-wise difference between X and X' is greater than zero. Finally, the fourth term is added in order to ensure that the generated counterfactual actually belongs to the desired class.

The architecture presented in this section is trained to simultaneously minimize the classification error of the original input values and to generate a counterfactual value that is close to the original value, that changes as few variables as possible (sparsity) and that satisfies the condition of changing the output class. Due to the fact that neural networks deal with continuous values for inputs and outputs, it is virtually impossible for the proposed architecture to generate a counterfactual value with any number of features remaining unchanged. Thus, the neural network will generate counterfactuals with all of their features having a variation in their values. This hinders sparsity and also interpretability, as minor changes in all of the features may be difficult for the end user to understand. To circumvent this issue, we follow the procedure described in (MOTHILAL; SHARMA; TAN, 2020). After generating the counterfactuals, all features whose difference with the original value is below a predefined threshold will be restored to their original value. In their work, they compare the MAD function value with the 10th percentile value of the absolute distance from the median, and use the minimum of the two as the threshold. In this work, it is suggested that the percentile value should be treated as an hyperparameter (referred to as φ), for three situations may arise. In the first one, no features are restored, thus the procedure is useless at enhancing sparsity. The opposite may also occur and all features are restored, thus the counterfactual being nullified. Finally, it may occur that due to the procedure, the counterfactual does not generate a change in the output class, thus decreasing the counterfactual performance. In this work, the percentile value is chosen as the maximum value in which no counterfactuals are nullified due to all features being restored. Thus, sparsity is enhanced without losing counterfactuals. This restoration process is illustrated in Figure 21.

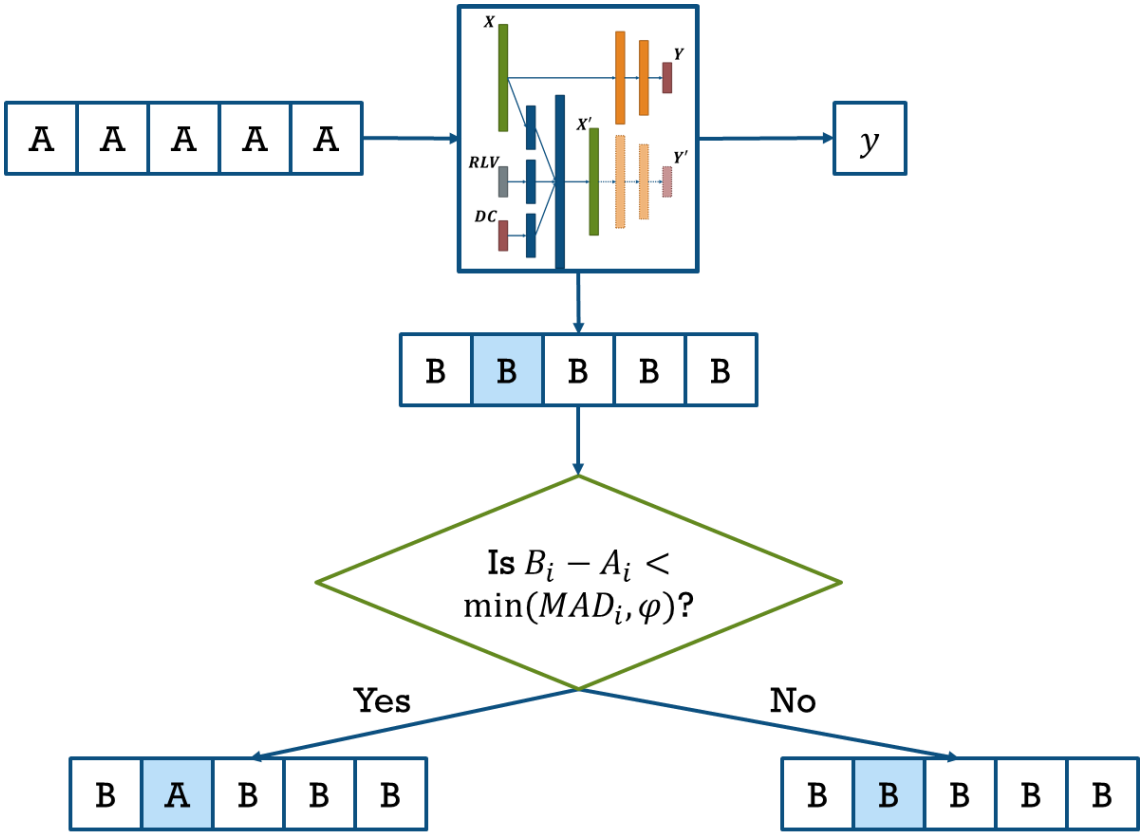

Figure 21 – Restoration process for sparsity enhancing. Example for the second feature.

After training, the model's performance must be evaluated considering the input values' classification performance, the counterfactual values' classification performance, and the quality of the counterfactuals according to the desirable properties described above. For both the original and counterfactual values' classification performance, accuracy is used. For the evaluation of the quality of the generated counterfactuals, there is no consensus for standard metrics. However, authors have proposed different metrics for evaluation. In (VAN LOOVEREN; KLAISE, 2021), the authors introduce the IM1 and IM2 metrics:

$$IM1 = \frac{\|X' - AE_t(X')\|_2^2}{\|X' - AE_o(X')\|_2^2 + \epsilon} \tag{43}$$

$$IM2 = \frac{\|AE_t(X') - AE(X')\|_2^2}{\|X'\|_1 + \epsilon} \tag{44}$$

, where X' is the counterfactual value, AE_t is an autoencoder trained only using data belonging to the counterfactual class, AE_o is an autoencoder trained using data from the

original class, and AE is the autoencoder trained with all of the classes. In this work, we use $\epsilon = 1.0 \times 10^{-7}$. The IM1 metric uses autoencoders trained on different classes to create representations of these classes. Thus, if a counterfactual is better reconstructed by AE_t than by AE_o, it means the counterfactual value is closer to the data manifold of the counterfactual class than the original class. On the other hand, the IM2 metric measures the closeness of the counterfactuals' data manifold to the general data manifold. In this sense, both metrics measure the interpretability of the achieved counterfactuals, with lower values implying more interpretability. However, in this thesis only IM1 is used, as IM2 has shown some issues with interpretability of its values (MAHAJAN; TAN; SHARMA, 2019). Also, in (SCHUT et al., 2021), authors compare the values of IM2 with in-distribution and out-of-distribution data, achieving no significant difference in their values.

Besides IM1 and IM2, we measure realism (Re) and sparsity (Sp) through the following metrics:

$$Re = \|AE(X') - X'\|_2^2 \tag{45}$$

$$Sp = \|X - X'\|_0 \tag{46}$$

The Re metric is based on the work by Nemirovsky et. al. (NEMIROVSKY et al., 2020). It measures how close the counterfactual is from the training set data manifold. The closer it is, the more realistic it is assumed to be. In the mentioned work, authors measure sparsity through the L1 norm instead of the L0 norm. Though this is a useful measure of sparsity, we prefer to use the L0 norm. The main reason is because it is used in the loss function for training the model, thus, the metric measures directly the quality of the optimization. Also, as some features will most likely be restored to their original value, the L0 norm will directly quantify how many features remained unchanged with respect to their original value after applying restoration procedure. Furthermore, the L1 norm is affected by how much each feature varies. While the L1 norm is a useful metric for evaluating proximity, we believe that the L0 norm is more suitable for evaluating sparsity. The L0 norm is also a measure of how interpretable the counterfactual is, as interpretability is directly related to the number of features altered from the original value.

The methodology followed for this framework is presented in Figure 22. Its steps are the following:

1. Preprocess raw data.
2. Split into train and test data. Train data is used to adjust the networks parameters during training and test data is used to measure its performance after training.
3. Train the SCF-Net using backpropagation.
4. With the trained network, classification performance is measured using the test set and counterfactuals are generated for each point in the test set. Restoration process is included in this step.
5. The generated counterfactuals are evaluated using the IM1, sparsity and realism metrics.
6. The set of counterfactuals generated using the test set is fed to the trained network. This is done in order to evaluate the validity of the generated counterfactuals.

To calculate the IM1 and realism metrics, one autoencoder must be trained for each available class. Also, one autoencoder must be trained with all the classes. In this work, each of these autoencoders is trained three times, and results are presented as the mean of the three predictions each time the SCF-Net is trained. Furthermore, the proposed classification and counterfactual generation model is trained five times. Thus, all performance metrics results are calculated as the mean of these five training processes.

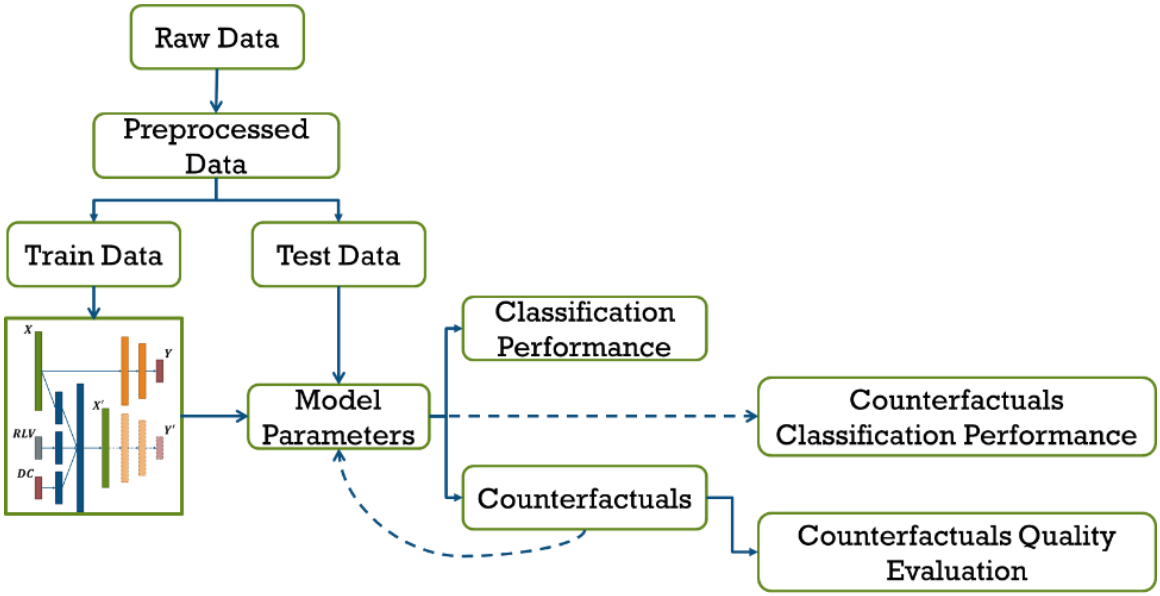

Figure 22 – Evaluation methodology for the proposed architecture.

3.2.1. Training process

Due to the fact that there are additional values besides X that are used as inputs to the SCF-Net, the training process is slightly modified. The two main modifications are the following:

1. Data augmentation: The SCF-Net must be able to generate counterfactuals for each of the available classes. Thus, it must respond correctly to the DC value. To achieve this, the train set is augmented according to the number of classes. For each datapoint in the training dataset, one vector is created for each class. Thus, during the training process the network will have to use the DC value properly to create a counterfactual belonging to that class. If this augmentation process was not made, the network would not see enough values to properly learn the representations needed for each class. Furthermore, the data is augmented for each possible class, including that to which the X value actually belongs. This is done in order not to give information to the network that should not be given. For example, in a binary classification case, not including the class that X belongs to and only the one that X does not belong to will give information to the network about the belonging class, and it will probably use the DC value to classify X in the upper part of the network. In

contrast, by including all of the classes in the data augmentation process, the network will not get information that can be used in the classification task.

2. Random latent variable: The RLV value is used to enhance diversity. Without its presence, the network will always generate the same counterfactual for each value. To use it during training, the random values are recalculated after each epoch. This is done to avoid the network to learn any bias from the value of the RLV itself, and only use it to generate a different counterfactual each time the same value is fed to the network.

3.2.2. Comparison with other techniques

Besides the evaluation of the SCF-Net, a comparison is made with two other techniques. The first one (referred to as (1) in this thesis) is based on the work by Wachter et. al. (WACHTER; MITTELSTADT; RUSSELL, 2017). The calculation of each counterfactual is treated as an optimization problem:

$$\min_{X-X'} \max_{\lambda} (f(X') - y')^2 + \lambda \|X' - X\|_1 \quad \text{subject to } |f(X') - y'| \leq \epsilon' \tag{47}$$

, where ϵ' is a tolerance value. To find the counterfactual values, the ADAM optimizer (KINGMA; BA, 2015) is used. The second one (referred to as (2) in this thesis) is based on the work by Van Looveren and Klaise (VAN LOOVEREN; KLAISE, 2021) in which class prototypes are used as guides for counterfactuals generation. The minimization problem used to generate counterfactuals is the following:

$$\min_{X-X'} \left(\max \left([f(X')]_{t_0} - \max_{i \neq t_0} [f(X')]_i, -\kappa \right) + \beta \cdot \|X - X'\|_1 + \|X - X'\|_2^2 + \left\| X' - X_{prototype} \right\|_2^2 \right) \tag{48}$$

, where t_0 is the original class, i corresponds to each class different than t_0, κ establishes the limit for the difference between the prediction values, β is a parameter to define the predominance of the L1 regularization, and $X_{prototype}$ is a value that represents a desired

class. This value is obtained by using k-d trees (BENTLEY, 1975) for each class and it is used to speed up the search for counterfactuals and to avoid out-of-distribution results.

3.3. FS-SCF: Combining Feature Selection and Counterfactual Generation for Prognostics and Health Management

To combine both feature selection and counterfactual generation in one algorithm, the Feature Selection and Sparse Counterfactual Generation network (FS-SCF) is proposed. It incorporates the use of the FS layer into the SCF-Net. Its representation is shown in Figure 23. As with the SCF-Net architecture, the inputs are a X vector containing sensor data information, a random latent variable (RLV) for enhancing diversity and a one-hot-encoded desired class (DC) vector that determines to what class does the generated counterfactual must belong to. The input vector X is used in the upper part of Figure 23 as a neural network classifier for fault diagnosis. However, before the typical hidden layers there is a feature selection layer, whose weights are used to determine the importance of each input feature and generate a feature ranking according to these values. In the lower part of the network, the $[X, RLV, DC]$ vector is used to generate a counterfactual close to X and belonging to the DC class. Like in the SCF-Net, the RLV value is used in order to enhance diversity. With the generated counterfactual, the weights used in the classifier above (including the feature selection layer weights) are copied for a forward pass, with the objective of determining the validity of the generated counterfactual. Like in the SCF-Net, the whole network is trained simultaneously, without pre-training of any layers. Thus, every time the weights are copied for a forward pass, their values are the ones obtained in the previous backpropagation step.

The training process of the FS-SCF network consists on the minimization of the following loss function:

$$loss = -\sum_{j=1}^{C} y_j \, log \, \hat{y}_j + \sum_{i=1}^{F} \frac{|X_i - X'_i|}{MAD(X_i)} \cdot \theta_i + \|X - X'\|_0 + -\sum_{j=1}^{C} y'_j \, log \, \hat{y}'_j \tag{49}$$

This loss function is very similar to the one for the SCF-Net. The only difference is the θ_i term, which corresponds to the feature importance value of the i-th input feature, obtained from the FS-layer's corresponding weight. Each time the loss is calculated, the θ_i values correspond to layer's weights in that specific epoch. Thus, these values vary with each epoch. The intuition behind the inclusion of the θ_i value in the loss function is that more important features should not have their values altered. Thus, there is a higher penalization in the loss function for important features. This forces the network to keep the values of those features closer to their original values. This was proposed in (GRATH et al., 2018)

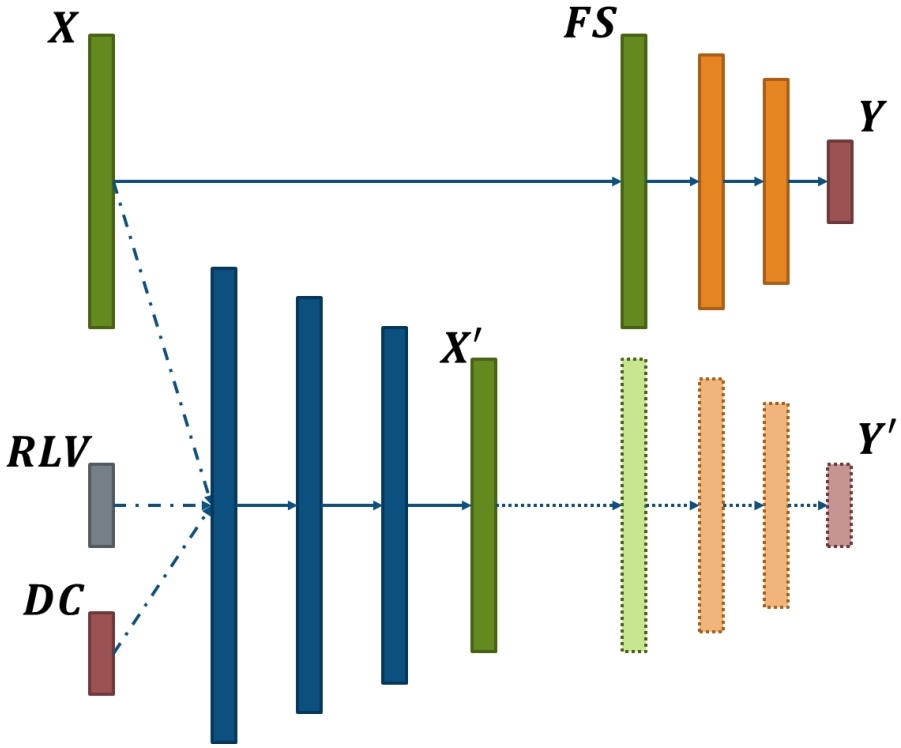

Figure 23 – FS-SCF network representation.

With the FS-SCF network, three kinds of information are obtained: task performance, counterfactuals, and feature importance values. While the first one relates to the original objective of neural networks (to do predictions with the highest performance possible), the two latter are used for interpretability. Thus, no external techniques are needed to analyze the trained model, as it is inheritably interpretable.

3.3.1. Necessity and Sufficiency Quantification

With the obtained counterfactuals for each value in the test set, this thesis proposes a method for necessity and sufficiency quantification. The algorithm proposed in (KOMMIYA MOTHILAL et al., 2021) for quantifying necessity and sufficiency relies on the ability to restrict what features can be altered and what cannot in the process of counterfactual generation. This is not possible in the proposed FS-SCF network, due to the nature of neural networks. To circumvent this, a restoration process similar to the one from (MOTHILAL; SHARMA; TAN, 2020) is used. The process is straightforward: For necessity quantification, all features' values except the analyzed one are restored to their original value, and the analyzed one is kept to the counterfactual value. The resulting vector is then fed to the FS-SCF network, to evaluate if it is a valid counterfactual or not. If it is, then the analyzed feature is necessary for explaining the original output, as a change in its value determines a change in the output's class. For sufficiency, the process is the opposite: The analyzed feature's value is restored to its original value, while all of the other values are kept as the ones in the counterfactual vector. If this resulting vector is not a valid counterfactual, then the analyzed feature's value is a sufficient explanation to the original output. This is because even though all of the other features' values where altered, it was not possible to create a valid counterfactual. An example of this is shown in Figure 24, for a feature that is both necessary and sufficient.

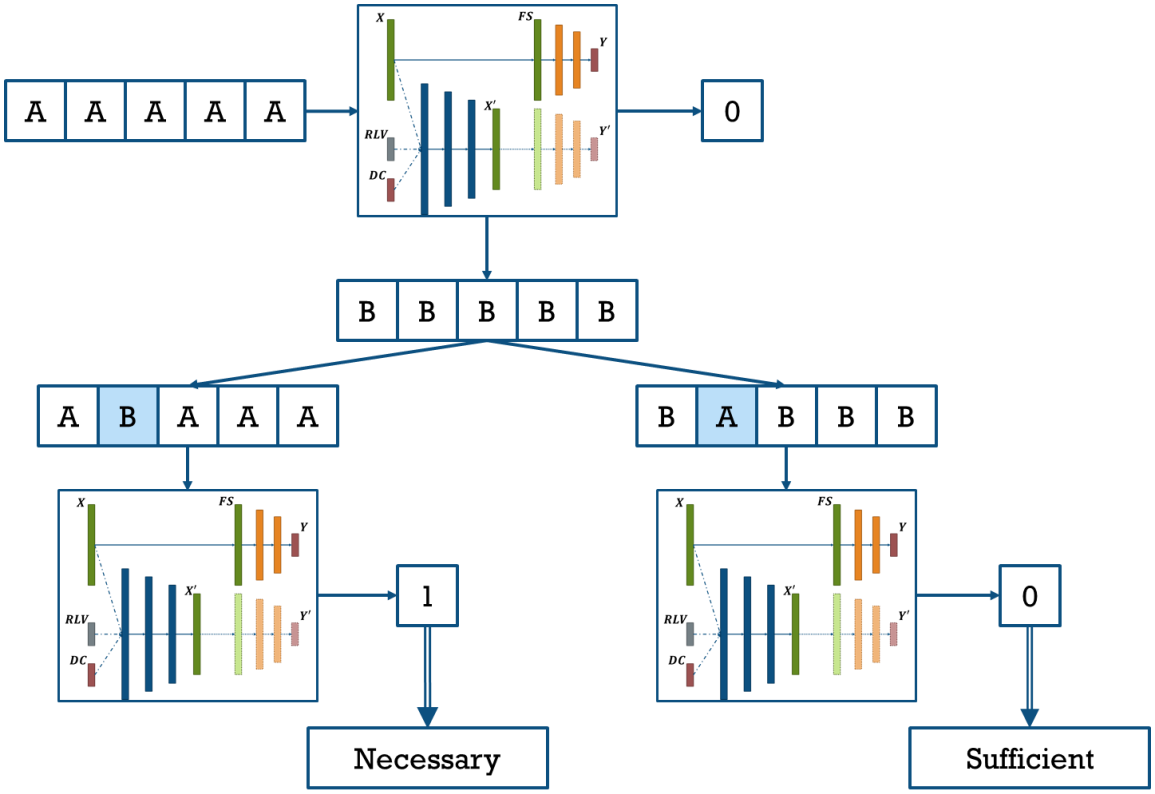

Figure 24 – Necessity and sufficiency quantification example.

By applying the necessity and sufficiency quantification method described above, users can know what features are necessary and/or sufficient for each generated counterfactual. Taking the mean value for all datapoints in the test set, each feature has a corresponding value between 0 and 1 indicating its necessity. The same process is followed for sufficiency. Besides necessity and sufficiency, two more properties are quantified: (necessity AND sufficiency) (NaS), and (necessity OR sufficiency) (NoS). The process of quantification is direct: if a feature is necessary and sufficient, the (necessity AND sufficiency) value will be 1. Otherwise, its value will be 0. The same process is followed for the (necessity OR sufficiency).

3.3.2. Correlation Between FS-based and Causality-based rankings

With the quantification for necessity, sufficiency, NaS and NoS, users can know which feature are more relevant regarding all of these properties. This information is used to create feature rankings and to compare them with the FS-based ranking obtained through the

feature importance values from the FS layer. This is done in order to try to understand why the FS-layer technique gives more relevance score to some features rather than others. To do this, the correlation between each of the four causality-based rankings and the FS layer is measured using the Spearman's rank correlation coefficient (WINTER; GOSLING; POTTER, 2016):

$$R_s = \frac{\sum_{i=1}^{N} R(x_i) \cdot R(y_i)}{\sqrt{\sum_{i=1}^{N} R(x_i)^2 \sum_{i=1}^{N} R(y_i)^2}} \tag{50}$$

, where $R()$ denotes the conversion of raw values to ranks. The range of its values is from -1 to 1. A negative value will indicate a negative correlation (the rankings are similar but in opposite order) and a positive value indicates a positive correlation (rakings are similar in the same order). Table 2 indicates the level of correlation that must be assumed according to the R_s value (XIAO et al., 2016).

The Spearman correlation is a special case of the Pearson correlation in which samples are converted into ranks. According to (XIAO et al., 2016), the Spearman correlation is able to capture the monotonic relationship between the variables. The Pearson correlation focuses in the values rather than the ranks. This may lead to misleading results, as exemplified in the work by Xiao et. al. There, they show a case where despite two vectors presenting the same order in their values, the Pearson correlation value is equals to 0.88, whereas the Spearman correlation value is of 1.0. Thus, the advantage of using the Spearman correlation when rankings are being compared.

Table 2 – Strength of correlation according to the R_s value. Adapted from (XIAO et al., 2016).

R_s	Correlation strength
[-1.0, -0.5]	Strong negative correlation
[-0.5, -0.3]	Moderate negative correlation
[-0.3, -0.1]	Weak negative correlation
[-0.1, 0.0]	None or very weak negative correlation
[0.0, 0.1]	None or very weak positive correlation
[0.1, 0.3]	Weak positive correlation
[0.3, 0.5]	Moderate positive correlation
[0.5, 1.0]	Strong positive correlation

3.3.3. FS-SCF Methodology

The FS-SCF network unifies the use of the feature selection layer, the multi-tasking SCF network and the proposed necessity and sufficiency quantification methodology. The methodology to evaluate this proposed network through different case studies is as follows:

1. Preprocess raw data. This includes handling missing data and noisy data. Feature selection is done in the feature selection layer, thus, any feature analysis with the intention of selecting the most appropriate ones is not necessary. However, features can be deleted if the number of missing values is excessive and hinders the analysis by either deleting too much datapoints or using interpolation excessively. That trade-off analysis must be done by the user.
2. Split the dataset into train and test data. Train data is used to adjust the network's parameters during training (including those related to feature selection), while test data is used for evaluation of the network's performance and counterfactual generation.
3. After training, the model's parameters are obtained. Amongst these, the feature selection layer's weights are used to create a feature ranking according to their importance. The trained model is used to evaluate its performance in the test set, and to generate counterfactuals for each datapoint in the test set and for every class except the original one.

4. The generated counterfactuals are evaluated according to quality measures, such as $IM1$, realism and sparsity. Furthermore, they are re-fed to the trained network in order to evaluate whether they achieve the objective of altering the output's class or not. Also, they are used to quantify the necessity and sufficiency of each input feature when explaining the original output value. This is done according to the process explained above.
5. With the quantification of necessity and sufficiency, features are ranked according to four criteria: necessity, sufficiency, necessity and sufficiency, and necessity or sufficiency.
6. The feature ranking obtained through the feature selection layer is compared to the rankings obtained through necessity and sufficiency quantification by using the Spearman correlation.

With this process being followed, the FS-SCF is an inherently interpretable neural network-based classifier that ranks features according to their importance for the model and is able to generate counterfactuals. Furthermore, through the generated counterfactuals, necessity and sufficiency are quantified in order to better describe the ranking obtained through the model's feature selection layer. The methodology described above is represented graphically in Figure 25.

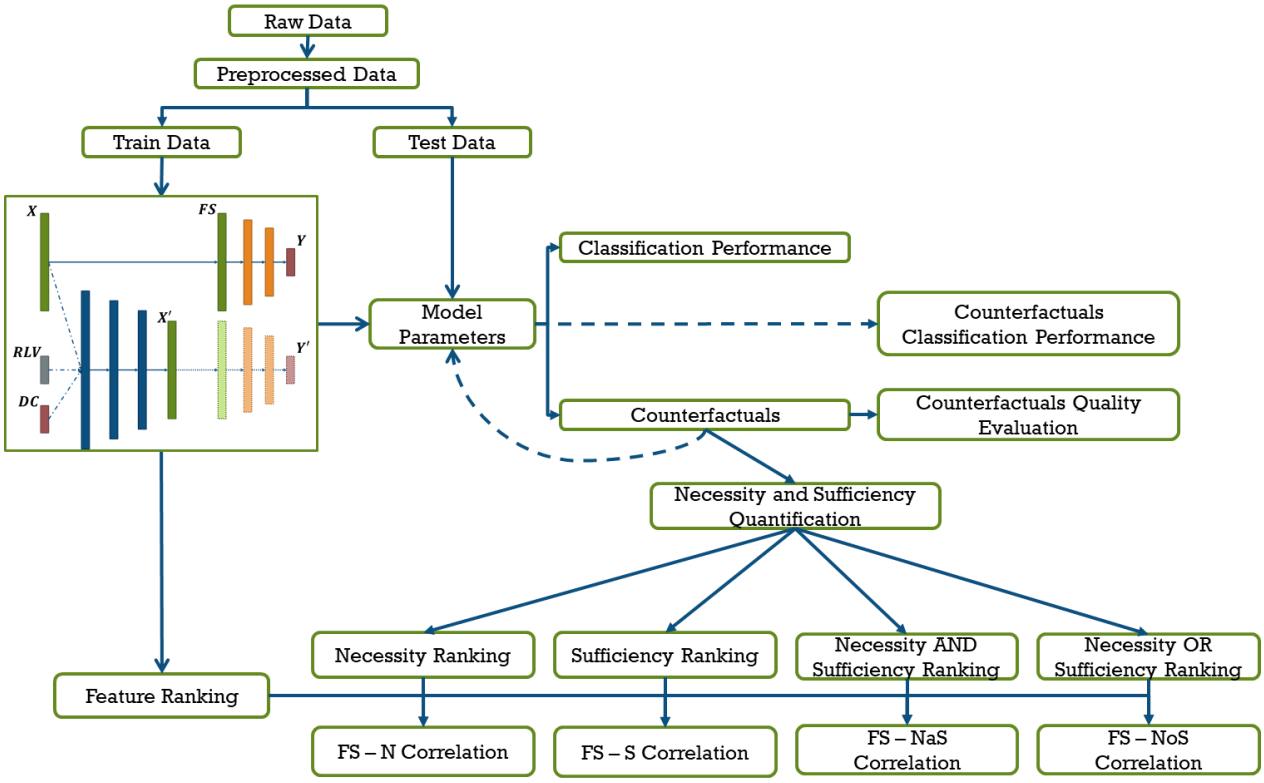

Figure 25 – FS-SCF network methodology.

4. CASE STUDIES

This section presents the case studies used to evaluate the performance of the proposed framework for fault diagnosis and prognosis and RUL prediction.

4.1. Case Western Reserve University Bearing Data Center (CWR)

The dataset is provided by the Case Western Reserve University Bearing Data Center, and consists of vibrational data collected with accelerometers on an electric motor's drive-end ball bearing (LOPARO, 2013). The description of the experiment is detailed in Table 3. Twelve health conditions are measured, which determine the bearings' health classes. Similarly to the work presented in (SAN MARTIN et al., 2019), 100 features are manually extracted from the data. These are presented in Table 4. The dataset has 8004 samples with 100 features and 12 classes. The classes' description and distribution are detailed in Table 5. The experiment setup is shown in Figure 26.

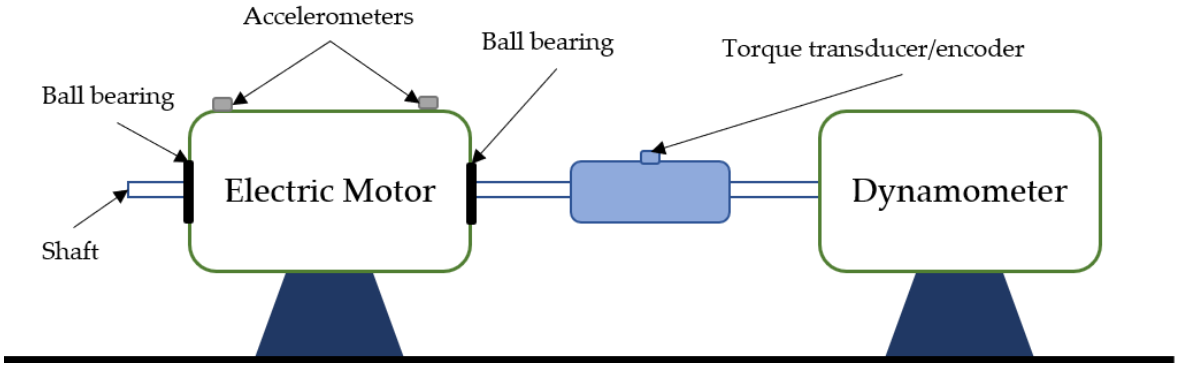

Figure 26 – CWR experiment setup. Based on (LOPARO, 2013)

Table 3 - CWR experiment settings.

Bearing Model	SKF 6205 2RSJEM
Sensors Location	Drive end bearing housing – 12 o'clock position
Faults Location	Balls, outer ring, or inner ring
Motor Loads	0-1-2-3 hp
Motor Speeds	[1720,1797] rpm
Sampling Rates	48 kHz (baseline) – 12 kHz (faults)

Table 4 - CWR features' description.

Type of signal	Feature	Feature Number
Original signal	Maximum amplitude	1
	Root Mean Square (RMS)	2
	Peak-to-peak amplitude	3
	Crest factor	4
	Arithmetic mean	5
	Variance	6
	Skewness	7
	Kurtosis	8
	Centered moments (k=5-11)	9-15
	Arithmetic mean of the Fourier amplitude, divided in 25 frequency bands	16-40
	RMS of the first five IMFs* (empirical mode decomposition)	41-45
	Percent energy of the first five IMFs (empirical mode decomposition)	46-50
	Shannon entropy of the first 5 IMFs (empirical mode decomposition)	51-55
	RMS of the first 5 PFs** (Local Mean Decomposition)	56-60
	Percent energy of the first 5 PFs (Local Mean Decomposition)	61-65
	Shannon entropy of the first 5 PFs (Local Mean Decomposition)	66-70
Derivative of the original signal	Maximum amplitude	71
	Root Mean Square (RMS)	72
	Peak-to-peak amplitude	73
	Crest factor	74
	Arithmetic mean	75
	Variance	76
	Skewness	77
	Kurtosis	78
	Centered moments (k=5-11)	79-85
Integral of the original signal	Maximum amplitude	86
	Root Mean Square (RMS)	87
	Peak-to-peak amplitude	88
	Crest factor	89
	Arithmetic mean	90
	Variance	91
	Skewness	92
	Kurtosis	93
	Centered moments (k=5-11)	94-100

*IMF: intrinsic mode functions; **PF: product functions.

Table 5 - CWR classes' description.

Class ID	Fault size [mm]	Fault location	Datapoints in the train set	Datapoints in the test set
Baseline	-	-	496	91
18BF	0.18	Balls	565	108
36BF	0.36	Balls	569	107
53BF	0.53	Balls	583	93
71BF	0.71	Balls	565	104
18IR	0.18	Inner race ring	572	104
36IR	0.36	Inner race ring	590	85
53IR	0.53	Inner race ring	576	99
71IR	0.71	Inner race ring	572	99
18OR	0.18	Outer race ring	565	110
36OR	0.36	Outer race ring	572	103
53OR	0.53	Outer race ring	578	98

To diagnose the bearing's health state, 100 extracted features are used. However, there are features that give more information about the health state than others. Furthermore, a situation may be presented in which the information of some features is obtained through other features, making them irrelevant to the network. Thus, it is necessary to determine the importance of each input feature fed to the network.

4.2. Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)

The second case study used in this work corresponds to the NASA Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) datasets (SAXENA et al., 2008), widely used as benchmark for RUL prediction methods (COFRE-MARTEL; DROGUETT; MODARRES, 2020; KONG et al., 2019; RUIZ-TAGLE PALAZUELOS; DROGUETT; PASCUAL, 2020; VERSTRAETE; DROGUETT; MODARRES, 2020). They are four datasets containing simulated data of turbofans degradation time series. The datasets description is shown in Table 6. As it can be noted in the table, data is already divided into train and test trajectories, each of them ending in failure. Also, operating conditions in the datasets alternate between 1 and 6 and fault modes alternate between 1 and 2, making each dataset different from the others. Each dataset is comprised of 21 sensor measurements, which are used to train models for RUL prediction. To generate the labels, the same procedure shown in (RUIZ-TAGLE PALAZUELOS; DROGUETT; PASCUAL, 2020) is

used. Thus, degradation is represented as a piecewise linear function of the time cycles, with a maximum RUL number of 125. A diagram of the engine used for creating the datasets is shown in Figure 27. The description of the features used for RUL estimation are listed in Table 7.

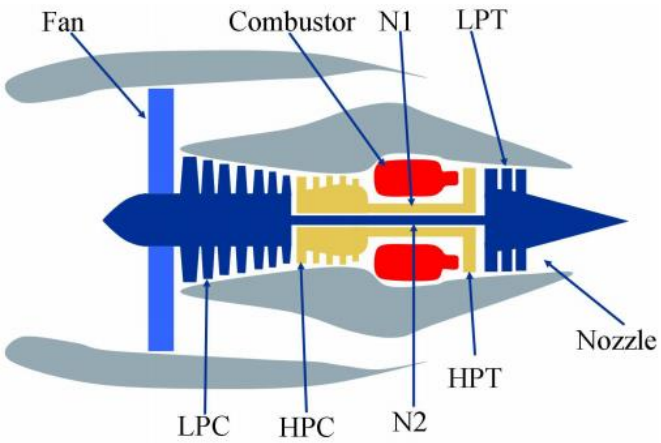

Figure 27 - Diagram of engine simulated in the C-MAPSS datasets. The components shown in the figure are the fan, the combustor, the nozzle, low-pressure compressor (LPC), high-pressure compressor (HPC), low-pressure turbine (LPT), high-pressure turbine (HPT). N1 and N2 indicate the low-pressure rotor and high-pressure rotor, respectively. Source: (SAXENA et al., 2008)

Table 6 - C-MAPSS datasets.

Dataset	Train Trajectories	Test Trajectories	Conditions	Fault Modes
FD001	100	100	1	1
FD002	260	259	6	1
FD003	100	100	1	2
FD004	249	248	6	2

Out of the four datasets, this work shows results on datasets FD001 and FD004, since this work's objective is to show the framework's performance in different scenarios, and these two datasets do not share any of the characteristics shown in Table 6. The sizes of the two datasets are shown in Table 8.

Table 7 - C-MAPSS features' description.

Feature Tag	Description	Feature Tag	Description
1	Total temperature at LP* compressor outlet	8	Ratio of fuel flow to HP compressor outlet
2	Total temperature at HP** compressor outlet	9	Corrected fan speed
3	Total temperature at LP turbine outlet	10	Corrected core speed
4	Total pressure at HP compressor outlet	11	Bypass Ratio
5	Physical fan speed	12	Bleed Enthalpy
6	Physical core speed	13	HP turbine coolant bleed
7	Static pressure at HP compressor outlet	14	LP turbine coolant bleed

*LP: Low Pressure; **HP: High Pressure

Table 8 - C-MAPSS train and test sets.

Dataset	# Training samples	# Testing samples
FD001	20,631	13,096
FD004	61,249	41,214

Among the input features, there are some more informative about the degradation of the turbofan than others. This means that some features (or combination of features) are more sensitive to a change in the health state than others. More specifically, there may be features that are related to a certain failure mode, that do not relate to another failure mode. Therefore, it is relevant to determine the importance of each input feature on both datasets.

4.3. Offshore Natural Gas Treatment Plant (NGTP)

In this case study, data comes from an offshore natural gas treatment plant used to remove CO_2 from the gas using amines. This process, also known as “gas sweetening” is done because CO_2 is corrosive and reduces the natural gas’s energetic value. It is also usually used to remove H_2S, which is toxic and corrosive. Figure 28 shows a schematic representation of the plant. The CO_2 removal is done in the amine contactor, where non-treated gas and the amines flow in countercurrent, so that the amines capture CO_2. The rest of the system has the function of removing the CO_2 from the amines, in order to it be used

again. To monitor the plant's performance, 10 variables are measured constantly. Figure 28 shows a description of the features, which are also identified in Table 9. The objective is to reduce the CO_2 levels in the treated gas to <2,500 ppm, which is the acceptable level. The dataset contains 4,885 points sampled every 20 minutes, divided in four time periods. It is used to train models for quantifying the CO_2 levels after gas treatment. Throughout this thesis, it is also used for classification, using the 2500 ppm threshold as a division between two classes describing the quality of the CO2 removal process. Class distribution is shown in Table 10.

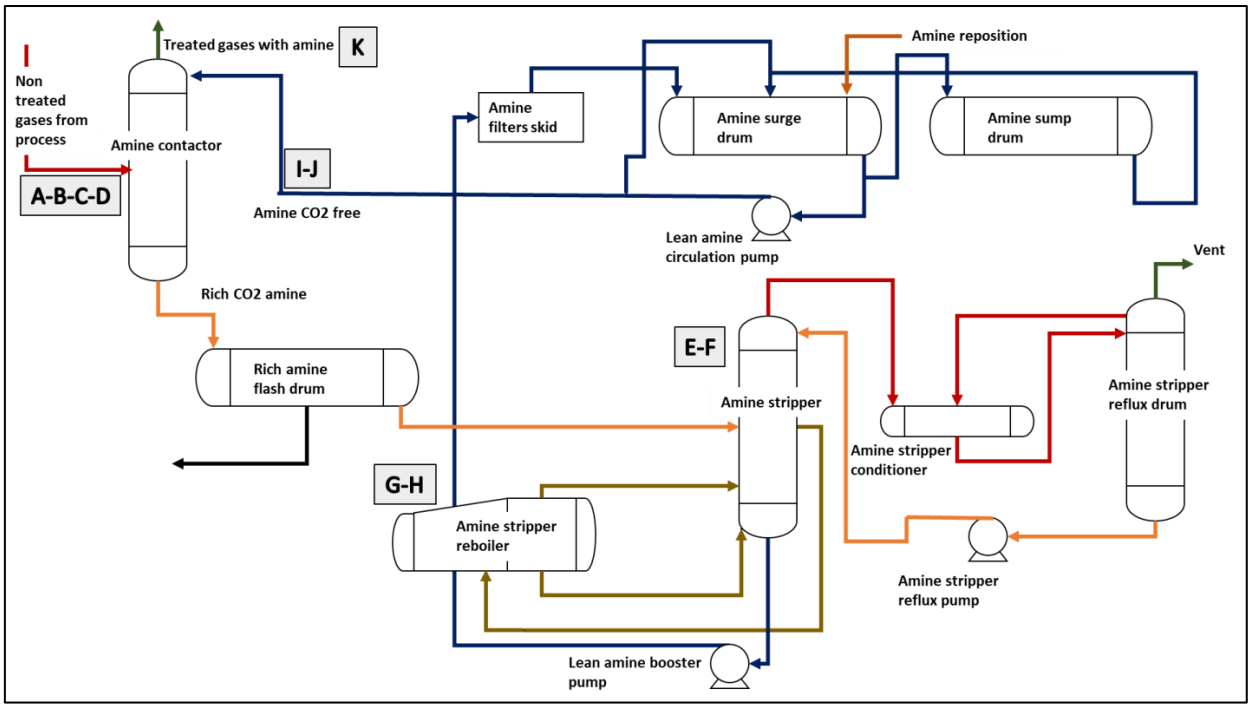

Figure 28 - CO_2 removal process illustration. Source: (FIGUEROA BARRAZA et al., 2020)

Table 9 - Monitored data for natural gas treatment plant. Source: (FIGUEROA BARRAZA et al., 2020)

Reference	Sensor Tag	Description	Units
A	Fluxo_Gas_A	Non-treated gas flow rate	kg/h
B	T_Gas	Non-treated gas temperature	ºC
C	P_Torre_Contactora	Amine contactor pressure	kPa
D	Delta_T	Temperature difference between non-treated gas and amine in the contactor	ºC
E	T_Torre_Stripper	Stripping tower temperature	ºC
F	P_Torre_Stripper	Stripping tower pressure	kPa
G	T_Amina_Reboiler	Amine temperature at the reboiler	ºC
H	P_Amina_Reboiler	Amine pressure at the reboiler	kPa
I	Fluxo_Amina	Amine flow rate	kg/h
J	T_Amina	Temperature of amine entering the contactor	ºC
K	CO2_gas	Particles per million of CO_2 in the treated gas	ppm

In this case study, input features are related to certain equipment acting on the process. To identify an irrelevant variable could lead to stop monitoring an equipment that does not need monitoring. On the other hand, identifying a feature with high relevance could lead to identify an equipment that has more impact on amount of CO_2 in the treated gas. This could help to improve the process.

Table 10 – Class distribution for the amine treatment plant database.

Dataset	Class 0 (CO_2 < 2500 ppm)	Class 1 (CO_2 > 2500 ppm)
Train	3674	1058
Test	906	277

4.4. Water Injection Pump for Production Stimulation in Offshore Wells

The dataset consists of different variables monitored from a water injection pump used for production stimulation when the oil well's pressure is decreasing. Like in most real-world datasets, this dataset presents some issues that must be addressed to feed a deep learning model. In this case, data being collected at different times and missing values are the two main issues. Regarding the first issue, values are grouped into 10-minute windows. Regarding missing values, a direct solution would be to interpolate. However, when there are

large windows of missing values, the achieved results could be misleading. To address this, each sensor data is analyzed independently, and values are interpolated if the missing values window is smaller than a threshold. Each threshold is determined by analyzing the behavior of each variable. After that, variables with a large number of missing values are discarded, in order to keep the dataset as large as possible. Finally, rows with missing values that where not interpolated are discarded. The remaining variables used in this study are described in Table 1.

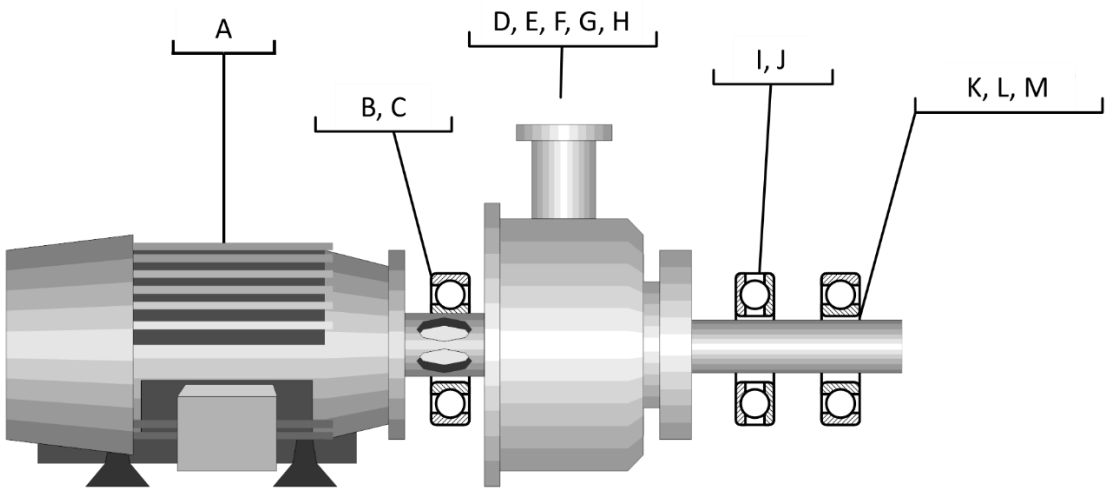

Figure 29 – Water injection pump representation.

Table 11 – Feature description for the water injection pump system

Reference	Sensor Tag	Description	Units
A	FIT-323D	Flow	m^3/h
B	IT-1031	Engine current	A
C	PT-305D	Suction pressure	kPa
D	PT-306D	Discharge pressure	kPa
E	TIT-305D	Lubrification circuit temperature	ºC
F	TIT-306D	Housing temperature	ºC
G	TT-308D	Axial (thrust) bearing side temperature	ºC
H	VT-301D	Non-drive end radial bearing vibration 1	µm
I	VT-302D	Non-drive end radial bearing vibration 2	µm
J	VT-303D	Drive end radial bearing vibration 1	µm
K	VT-304D	Drive end radial bearing vibration 2	µm
L	ZT-301D	Axis-bearing spacing 1	mm
M	ZT-302D	Axis-bearing spacing 1	mm

This dataset can be used for classification, as each datapoint is tagged according to its corresponding health state. Four kinds of health states are presented: healthy, incipient fault, degraded fault, and critical failure. The distribution of all classes is shown in Table 12.

Table 12 – Class distribution for the water injection pump dataset.

Dataset	Healthy	Incipient	Degraded	Critical
Train	53681	17070	2342	1887
Test	6024	1838	251	219

5. RESULTS AND DISCUSSION

In this section, the results are presented and discussed for the three proposed frameworks in this thesis for neural network interpretability. This section is divided in three subsections, one subsection for each proposed framework.

5.1. Feature Selection

In this subsection, results obtained for all case studies are presented, including the different models' configuration, RQS values, performance, and comparison with other techniques. All models are trained using Python v3.7, Tensorflow v2.1.0 and Keras v2.3.1. Hardware configuration is the following: Intel i5-9600K CPU, 16 GB DDR4 RAM and NVIDIA 11GB Geforce RTX 2080 Ti GPU. The configurations of the trained models are shown in Table 13. For this framework, three case studies are used: the CWR bearing data, the CMAPSS dataset, and the amine treatment plant data for CO_2 removal. The objective is to show the capabilities of the proposed framework for classification (CWR), regression (amine treatment plant) and RUL prediction (CMAPSS) tasks. That is why those three cases are used for this framework.

Table 13 - Configuration of the four trained models.

Case Study	**CWR**	**C-MAPSS**		**NGTP***
Dataset		**FD001**	**FD004**	
Hidden layers	2			
Neurons per hidden layer	64-32			
Activation functions	ReLU - ReLU			
Learning rate	1e-3			
FS layer activation function	Tanh	Sigmoid	Linear	Tanh
Regularization rate	1e-6	1e-5	1e-3	20,000
Epochs	2000	800	10,000	10,000

*NGTP: Natural Gas Treatment Plant

5.1.1. Case Western Reserve University Bearing Data Center (CWR)

Figure 30 shows the evolution of the relative F1-score as a function of the number of features. As detailed in section 3, features are arranged according to their importance in

descending order. Regarding the use of an FS layer, it may be observed that approximately 30 features are needed for the model to reach performance values close to the maximum. After that, performance remains at a high level, reaching its maximum value with 44 variables, requiring much less variables than the rest of the techniques. When comparing with other methods, it is clearly seen that they are outperformed by the proposed technique. The difference between the models suggests there is a large number of features with redundant information which cannot be easily recognized by traditional methods. However, they are identified through the use of an FS layer, achieving a low importance value. It is believed that this sparse behavior is encouraged by the regularization term in equation 39.

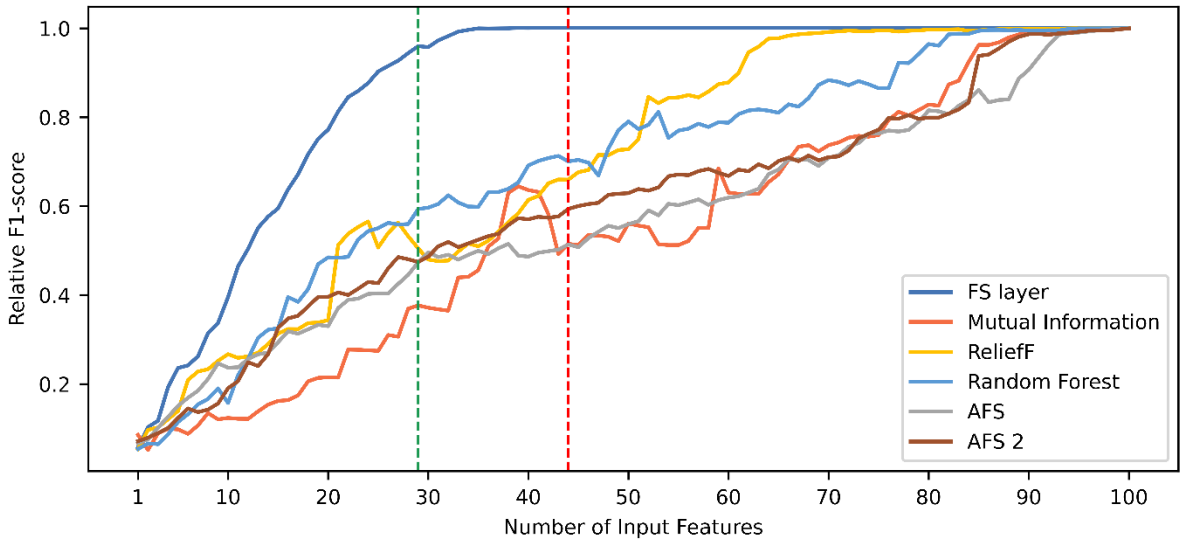

Figure 30 - Comparison of relative F1-score based on six different rankings. The red and green dotted lines indicate the number of features needed when using the FS layer technique to reach maximum performance and a 95% of this value, respectively.

Table 14 shows the performance of the different methods in terms of ranking quality and task performance. It also shows the number of variables needed to reach the maximum performance value and a 95% threshold. The RQS values presented confirm what is seen in Figure 30 regarding ranking quality. The FS layer technique achieves the highest RQS value, followed by ReliefF, Random Forest, AFS 2, Mutual Information and AFS. At some point, all techniques reach a very similar level of performance (shown in the table by max(PM)). However, the number of input features required for reaching the neighborhood of that value is different for each technique. The proposed technique requires 29 features to reach the 95% performance level threshold. In turn, the ReliefF, Random Forest, AFS 2, Mutual Information and AFS techniques need 63, 80, 87, 85 and 92 features, respectively. This example shows

how the RQS metric favors techniques that require less variables to achieve high levels of performance. As can be seen in Figure 30 and Table 14 , those techniques with the highest RQS need less input features to achieve a high level of performance. In particular, when comparing the AFS 2 technique with the Mutual Information technique, it can be seen that both reach high levels of performance needing approximately the same number of features (87 and 85, respectively). However, because the AFS 2 technique in most of the cases shows higher performance than the Mutual Information technique, it has a higher RQS.

Overall, it can be seen that those techniques where the RQS value is higher, less features are needed to reach the 95% performance threshold. In this case, a high RQS indicates that some variables can be discarded without an important performance loss with respect to the maximum value.

Table 14 - Task performance and ranking quality for CWR dataset.

Approach	**RQS**	$\boldsymbol{max(PM)}$	**Number of variables for** $\boldsymbol{max(PM)}$	**PM for 95% threshold**	**Number of features for 95% threshold**
FS layer	**0.9754**	**1.0014**	**44**	0.9599	**29**
Mutual Info.	0.7215	1.0000	100	0.9626	85
ReliefF	0.8630	1.0000	95	0.9587	63
Random Forest	0.8346	1.0000	98	0.9646	80
AFS	0.7127	0.9997	100	0.9629	92
AFS 2	0.7466	0.9997	100	0.9545	87

Regarding performance, Table 14 shows that feature selection improves performance regardless of the technique. Moreover, the highest improvement in performance is achieved when using FS layer, with a 0.14% F1-score improvement, reached with the 44 most relevant features. This suggests that the inclusion of an FS layer not only maintains the performance level, but also improves it. However, the difference in performance for this case study is not high enough as to be conclusive.

Regarding model selection, the FS layer technique is the most suitable for both criteria. In the first one, the highest performance value is reached with this technique, also with the least number of features. For the second criterion, the desired threshold is reached with only 29 features. For this criterion, the performance value is not as important as the number of features needed, because the value is above the determined threshold.

Table 15 shows the duration of the feature importance value generation process for each technique. It can be seen that the mutual information and random forest techniques achieve the best results, followed by the proposed FS layer technique. Furthermore, when the latter is compared to other embedded techniques (such as AFS and AFS 2), it can be noted that the proposed technique is much less time consuming. This is due to the fact that the FS layer technique only adds one parameter per feature. This is not the case for the AFS and AFS 2 techniques, as each feature requires an attention module. As the number of features grows, the time difference between the proposed technique and the AFS techniques should also grow.

Table 15 - Time duration for feature importance values generation (CWR case). For the embedded tech-niques (FS layer, AFS and AFS 2), this value is calculated as the difference between the total training time and the training time for a vanilla model without any embedded techniques. For the random forest technique, this value corresponds to the total training time, as there is no way to identify how much time it takes to obtain the desired values.

Approach	FS layer	Mutual Info.	ReliefF	Random Forest	AFS	AFS 2
Time [s]	89.3	5.2	595.3	10.2	2377.0	3026.5

5.1.2. Commercial Modular Aero-Propulsion System Simulation (C-MAPSS): FD001

Figure 31 shows the model's relative mean squared error (MSE) according to the number of features used for testing. The proposed technique is compared to the other five techniques. As seen in the figure, the FS layer technique presents better performance than the other techniques when using from three up to 12 input features. High performance level is reached when using the top nine input features, and the maximum value is reached with 12 features. In general, the performance evolution when using the FS layer technique appears to be more stable and smoother than the others, with a linear rate at the beginning, then stabilizing at a performance value close to 1.0. On the other hand, the Mutual Information technique shows a stable behavior with low performance until testing with the top 12 features. After that, there is a sudden increase in performance. This suggests that the two least important features got incorrect weights by this technique. Since it does not include relationships between the input features and only the relationship of each feature with the

output, it is highly possible that there are codependent features. In contrast, when looking back at the FS layer curve, the stability of the curve suggests that the technique successfully captures the dependences between input features.

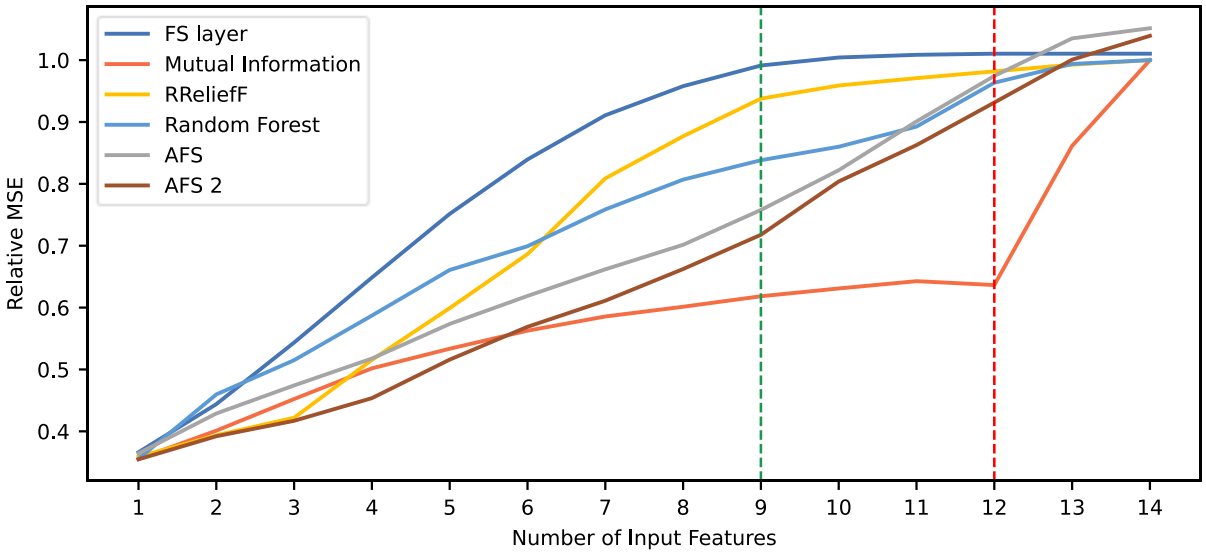

Figure 31 - Comparison of relative MSE according to six different rankings. The red and green dotted lines indicate the number of features needed, when using the FS layer technique, to reach maximum performance and a 95% of this value, respectively.

To do a more complete analysis about the aforementioned issue regarding the Mutual Information-based technique, Table 16 shows the two most relevant features and the two least relevant features for each technique. The feature located at the middle of the ranking is also shown. Indeed, note that mutual information-based technique gives the least relevance to variables 6 (physical core speed) and 10 (corrected core speed), which are among the two most relevant features in most of the other techniques. This shows that the core speed features alone are not enough for calculating the RUL of the turbine engine. However, alongside the rest of the variables, they are highly relevant. Because of this, the mutual information-based technique is not able to recognize their importance. It is also noteworthy that the FS layer technique is the only one that gives a high importance value to feature 8 (ratio of fuel flow to HP compressor outlet), while feature 10 goes to the seventh place. This helps understand why the FS layer technique presents a performance evolution curve with a better behavior than the others.

Table 16 - Features ranked 1st, 2nd, 7th, 13th and 14th in the ranking, for the six compared techniques.

Ranking	FS Layer	Mutual Info.	RReliefF	R. Forest	AFS	AFS 2
1st	8	7	**10**	7	**10**	**10**
2nd	**6**	3	**6**	**6**	7	**6**
7th	**10**	13	8	11	4	12
13th	5	**6**	12	5	12	2
14th	2	**10**	14	12	8	11

To compare the different approaches quantitatively, Table 17 shows their ranking quality and task performance in terms of relative MSE. The results show that the proposed model achieves higher RQS, as it can be also seen in Figure 31. Also, along with the AFS and AFS 2 techniques, the proposed FS layer technique reaches a maximum performance value $\max(PM) > 1$, meaning the use of this technique results in a performance improvement. However, as shown in Figure 31 and Table 17, the use of the AFS and AFS 2 techniques leads to better results in terms of performance. This case serves as an example of a situation where, despite having a high RQS, a feature selection technique may lead to a maximum performance value lower than other techniques. Thus, the RQS metric does not measure performance, but how fast the maximum level of performance is reached, in terms of number of input features needed. Indeed, the FS layer technique presents the higher RQS value and, consequently, Table 17 shows that out of the 12 features required to achieve maximum performance, 3 of those can be discarded in a restricted scenario and still present a performance value within above the 95% of the maximum value. The other techniques require more features to achieve this threshold. This opens the possibility for a two-fold analysis for model selection, by looking at performance and RQS simultaneously. This is useful when there is a limitation of input features to use in deployment, when the difference between performance is minimal between two models, or when performance level requirements are not so demanding. In this case study, despite the AFS techniques leading to a higher performance, the FS layer technique could be selected if there is a situation in which the number of input features is limited, as shown with the 95% threshold values. For example, when using the top 9 features, the FS layer technique leads to a better performance than the rest of the techniques, and it is a value close to its maximum. In the case of the AFS techniques, the performance value with the top 9 features is considerably lower than its maximum, far below the 95% threshold. This kind of information is summed up in the RQS metric, which is why it gives useful information for model selection.

Table 17 - Task performance and ranking quality for C-MAPSS FD001 dataset.

Approach	RQS	$max(PM)$	Number of features for $max(PM)$	PM for 95% threshold	Number of features for 95% threshold
FS layer	**0.9216**	1.0104	**12**	0.9913	**9**
Mutual Info.	0.6786	1.0000	14	1.0000	14
RReliefF	0.8730	1.0000	14	0.9589	10
Random Forest	0.8465	1.0000	14	0.9634	12
AFS	0.7821	**1.0515**	14	1.0351	13
AFS 2	0.7560	1.0392	14	1.0007	13

Table 18 shows the time consumption for each technique when calculating the feature importance values. As with the previous case, the mutual information and random forest techniques achieve the best results, followed by the proposed technique. When comparing with the AFS techniques, the difference is not as large as with the previous case. However, it is still considerable. The AFS techniques take an order of magnitude more time than the proposed FS layer technique. This shows that despite having less variables compared to the previous case (100 against 14), the FS layer is still a faster technique.

Table 18 - Time duration for feature importance values generation (C-MAPSS FD001 case).

Approach	FS layer	Mutual Info.	ReliefF	Random Forest	AFS	AFS 2
Time [s]	53.9	7.3	503.3	26.7	316.9	322.9

5.1.3. Commercial Modular Aero-Propulsion System Simulation (C-MAPSS): FD004

Figure 32 shows the relative MSE values depending on the number of variables used, based on each ranking technique. The behavior of the curves is similar and unusual, since the performance does not increase unless all 14 features are used. This indicates that the individual contribution of each feature is almost zero, and none of the evaluated techniques was able to identify relations between features that lead to an improved prediction of the

remaining useful life. This is opposite to the results shown in section 5.1.2 using the FD001 dataset, in which performance improves as more features are added. A plausible reason is the fact that the FD004 dataset includes six operation conditions and two failure modes, whereas the FD001 dataset only includes one of each. Therefore, the FD004 data is substantially more heterogeneous. Also, it is possible that features have different importance values depending on the operating condition and failure mode, and that there is not a ranking capable of integrating the information of all cases within the dataset. It can be concluded that a single model for the whole dataset is not an optimal solution, and the division in subsets (and, therefore, more homogeneous data) could be a better approach.

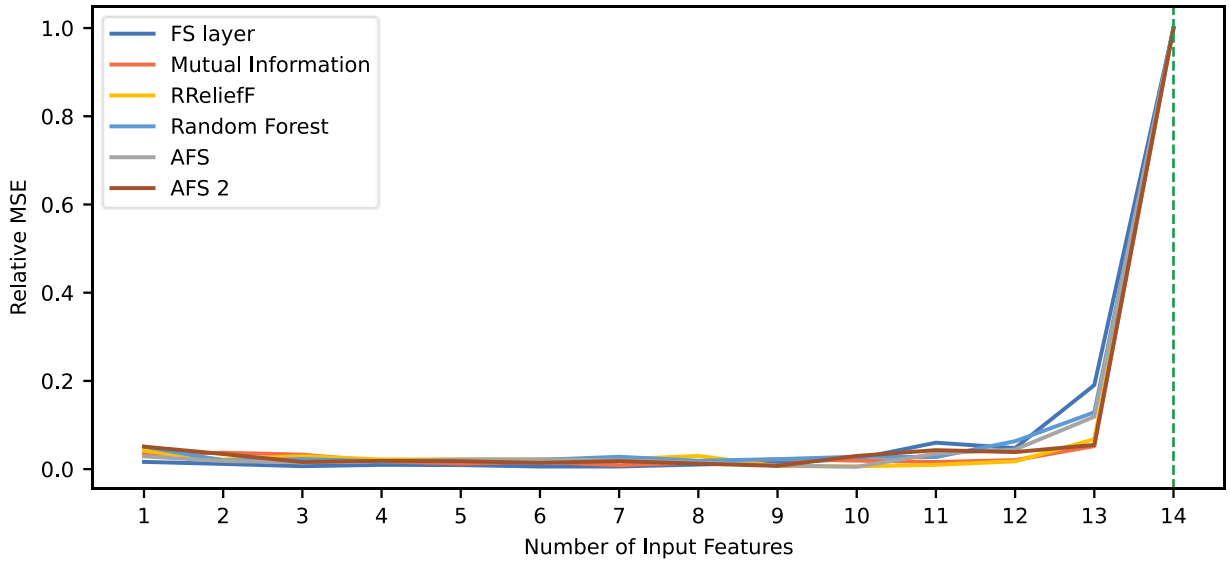

Figure 32 - Comparison of relative MSE based on six different rankings for C-MAPSS FD004 dataset. The green dotted line indicates the minimum number of features needed when using the FS layer technique to reach a 95% of the maximum performance value. Since there is an abrupt decay in performance when using 13 features, no features can be discarded to keep a performance level above 95% of the maximum value.

Table 19 shows the performance associated to each technique, along with its ranking quality. Results show how, despite the FS layer-based model achieves the higher RQS, it does not achieve a relative MSE value higher than 1. In this case, the use of an FS layer does not imply an improvement in performance. However, the difference is of 0.05%. Thus, it can be concluded that the same level of performance is maintained. These results are valid for the two criteria for model selection, since discarding a feature results in an important decrease in performance, independently of the employed technique.

Table 19 - Task performance and ranking quality for C-MAPSS FD004 dataset.

Approach	RQS	$max(PM)$	Number of features for $max(PM)$	PM for 95% threshold	Number of features for 95% threshold
FS layer	**0.1750**	0.9995	14	0.9995	14
Mutual Info.	0.1524	**1.0000**	14	1.0000	14
RReliefF	0.1545	**1.0000**	14	1.0000	14
Random Forest	0.1715	**1.0000**	14	1.0000	14
AFS	0.1645	**1.0000**	14	0.9761	14
AFS 2	0.1586	**1.0000**	14	0.9652	14

Table 20 shows how much time each technique takes to calculate the feature's importance values. Like in the two previous cases, the mutual information and random forest techniques achieve the best results, followed by the proposed technique. RReliefF, AFS and AFS 2 take longer than the rest to calculate these values. When comparing the proposed FS technique with the AFS techniques, it can be noted that the AFS techniques take more than six times longer than the proposed technique. The difference with the C-MAPSS FD001 case comes from the fact that the FD004 dataset has three times more datapoints, and that it is trained for 10,000 epochs instead of 800.

Table 20 - Time duration for feature importance values generation (C-MAPSS FD004 case).

Approach	FS layer	Mutual Info.	ReliefF	Random Forest	AFS	AFS 2
Time [s]	1435.1	120.0	6347.0	120.7	9083.7	9124.3

5.1.4. Offshore Natural Gas Treatment Plant (NGTP)

Figure 33 shows the evolution in performance when adding less relevant features for different techniques. Note that the FS layer technique reaches significantly higher performance than the others rest. With 7 out of the 10 variables, relative MSE is >1. However, maximum performance is reached when using all 10 features.

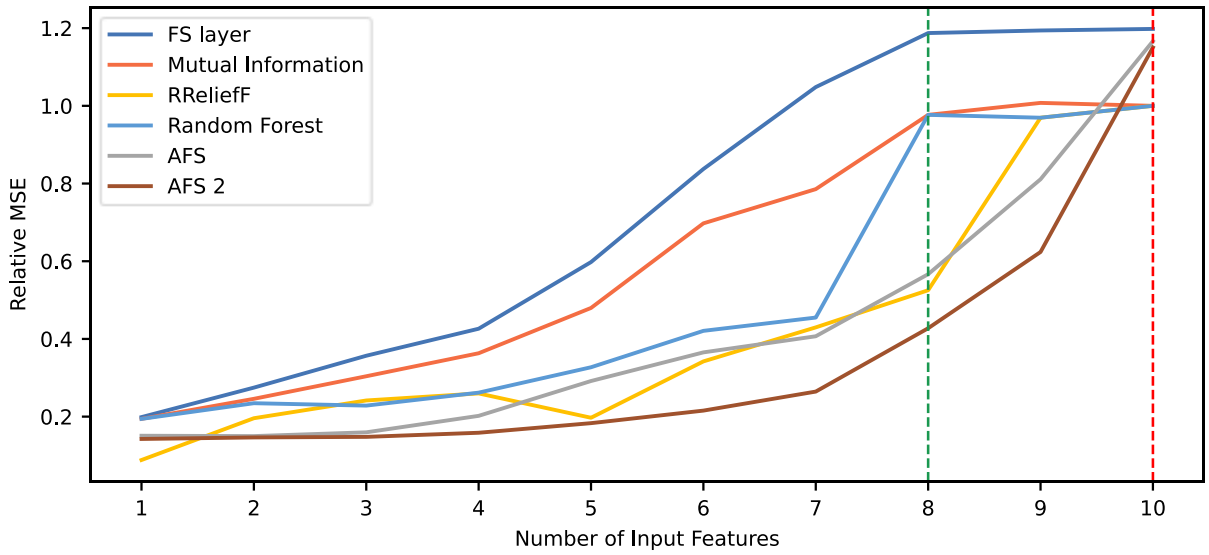

Figure 33 - Comparison of relative MSE based on six different rankings for NGTP dataset. The red and green dotted lines indicate the number of features needed when using the FS layer technique to reach maximum performance and a 95% of this value, respectively.

In order to understand how the FS layer model ranks features, Figure 34 shows the FS layer weights calculated by the model after training. It can be seen that the most relevant features are related to the amine contactor and the reboiler. Important features related to the contactor are the non-treated gas temperature, the amine temperature before entering the contactor, and the amine pressure at the contactor. This is coherent with the fact that the CO_2 removal process occurs in the contactor and, therefore, these variables directly affect its performance. The amine temperature at the reboiler appears as the second most relevant feature, higher than stripping tower-related features. Since the heating process that occurs in the reboiler comes immediately after the stripping process, there is a correlation between them. Results in Figure 34 show that the technique is able to capture this correlation and in the correct order. On the other hand, the least important feature is the temperature difference between non-treated gas and amine in the contactor. This means that the proposed approach can calculate internally that information from other features (i.e., non-treated gas temperature and temperature of the amine entering the contactor) and, thus, can dispose that feature. Despite the fact of the technique needing all 10 features to reach its maximum relative MSE (as seen in Figure 33), performance increments are marginal when adding the two least relevant features. A more robust neural network (more layers or neurons per layer) could solve this issue and reach maximum performance without the need of these two last features.

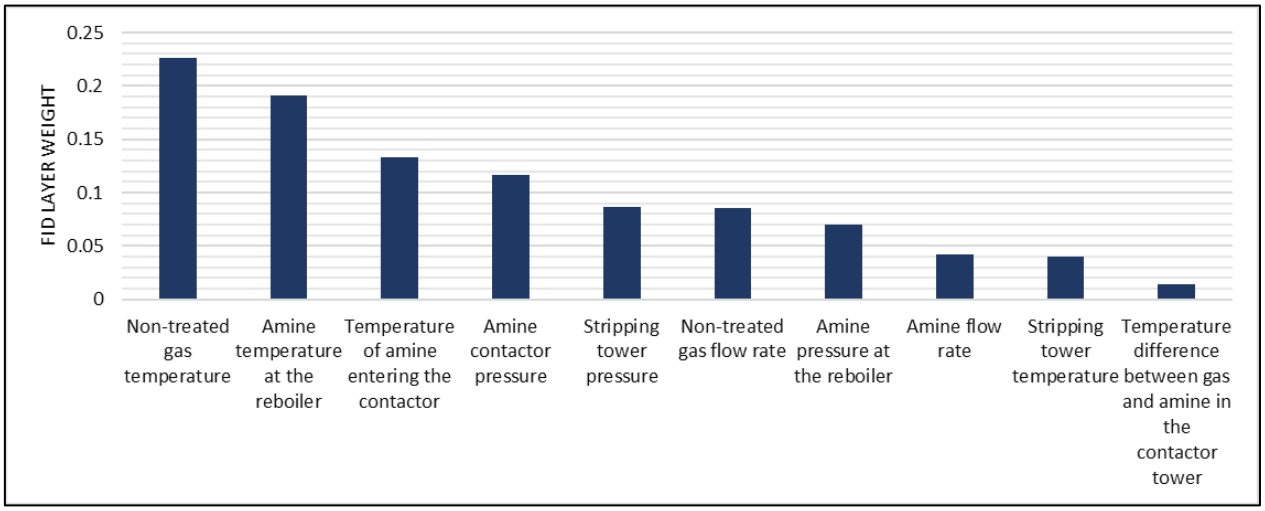

Figure 34 - Ranking of features according to the FS layer technique.

Table 21 shows the quantitative comparison of the different techniques. Results show that along with the FS layer technique, the AFS techniques (AFS and AFS 2) and the Mutual Information technique led to a relative MSE higher than 1. Despite the latter reaching the maximum performance with 9 variables, its value is considerably smaller than the one reached by the proposed approach. When using an FS layer, there is a performance increase of approximately 20%. Regarding ranking quality, the proposed approach has similar RQS value to the Mutual Information-based approach. This verifies what it shown in Figure 33. Although reaching different maximum values, the two curves show a similar behavior, with a performance improvement rate that stabilizes when using more than eight input features. On the contrary, the AFS and AFS 2 techniques reach a high maximum performance value, but relying heavily on the use of all input features available. For example, when using nine out of the ten features, both techniques present the lowest performance values. Thus, despite achieving high performance, their RQS values are the lowest of all six techniques presented. This emphasizes the fact that the RQS metric does not consider the maximum performance reached, but considers how fast that performance is reached in terms of input features required. This is also emphasized when looking at the results for the 95% performance threshold criterion. The AFS and AFS 2 techniques, which show the lowest RQS values, cannot discard any features from the feature set without having a major decrease in performance. This is not the case with the other techniques, in which one or two features can be discarded. In the particular case of the FS layer technique, 8 features can be used for deployment with a performance decrease below 5% with respect to the maximum value and still have a better performance value than the rest of the techniques. This shows how the RQS metric helps in model selection process even when different criteria are used. The RQS

indicates how likely it is to discard features without having an important performance decrease. The decision must be made by jointly analyzing performance values and RQS. After this, the corresponding criterion must be used to select the appropriate number of features.

Table 21 - Task performance and ranking quality for the NGTP case study.

Approach	RQS	$max(PM)$	Number of features for $max(PM)$	PM for 95% threshold	Number of features for 95% threshold
FS layer	**0.7755**	**1.1979**	10	1.1873	**8**
Mutual Info.	0.7582	1.0075	**9**	0.9767	**8**
RReliefF	0.5676	1.0000	10	0.9693	9
Random Forest	0.6596	1.0000	10	0.9767	**8**
AFS	0.4946	1.1664	10	1.1664	10
AFS 2	0.4126	1.1503	10	1.1503	10

Table 22 shows how much time each technique takes for calculating feature importance values. The same trend as the previous cases is shown, with mutual information and random forest achieving the best results, followed by the proposed FS layer. Like in the other cases, the AFS techniques take similar time to obtain these values. In this case, the AFS techniques take much more time for calculating feature importance values than the proposed technique, more than five times. Through the comparison of the three embedded ad-hoc techniques in the four case studies, it is able to see that the AFS techniques add a larger amount of complexity to the model in order to obtain feature importance values, which is reflected in the time they take.

Table 22 - Time duration for feature importance values generation (NGTP case).

Approach	FS layer	Mutual Info.	ReliefF	Random Forest	AFS	AFS 2
Time [s]	188.2	0.5	237.8	5.4	1059.6	1071.0

5.2. SCF-Net

This section presents the results obtained for two case studies. The specifications for all models are the following: Python v3.9, Tensorflow v, and Keras v. Both techniques for comparison were trained using the Alibi library (KLAISE; LOOVEREN; VACANTI, 2021). The hardware configuration is: Intel i5-9600K CPU, 16 GB DDR4 RAM and NVIDIA 11 GB Geforce RTX 2080 Ti GPU. For this framework, only classification tasks can be modelled. Thus, the CMAPSS dataset cannot be used. Furthermore, the amine treatment plant case is modified for a classification task. Two classes are used according to the 2500 ppm threshold. Thus, datapoints in which the amount of CO_2 in the treated gas is <2500 ppm belongs to the healthy state condition, and datapoints where CO_2 >2500 ppm belong to the fault class. Furthermore, in order to use case studies belonging to industrial real-world applications, the water injection pump case is used instead of the CWR case, which is not a case coming from an industrial application.

5.2.1. Case Study 1: Water Injection Pump

Table 23 shows the mean counterfactual metrics for all three techniques analyzed in this thesis. Results show that, through the SCF-Net, the generated counterfactuals show better quality than those generated by the two other techniques. In the case of the IM1 metric, the proposed technique achieves a lower value than the compared techniques. This means that, when comparing the distance to the data manifold of the counterfactual class with the distance to the data manifold of the original class, those counterfactuals generated using a SCF-Net present the better ratio. When compared to the next best ratio (namely, technique (2)), the proposed technique has a 39% reduction in IM1.

Table 23 - Counterfactual quality evaluation metrics for the water injection pump case study.

Technique	IM1 (↓)	Realism (↓)	Sparsity (↓)	Model Accuracy (↑)	CF Accuracy (↑)	CF generation success rate (↑)
SCF-Net	1.31	5.60×10^{4}	3.28	97.97%	99.50%	100.00%
Wachter et. al. (WACHTER; MITTELSTADT; RUSSELL, 2017) **(1)**	2.58	3.74×10^{5}	12.08	98.20%	88.28%	35.43%
Counterfactuals Guided by Prototypes (2)	2.15	1.51×10^{7}	7.67	98.20%	95.40%	67.99%

When looking at the Realism metric, a similar behavior is shown, with the SCF-Net having a better metric value. This means that the counterfactuals generated are closer to the train set data manifold than those generated by the techniques (1) and (2). In general, the proposed technique achieved better results in the two aforementioned metrics because of the architecture's structure. Specifically, because the desired class for the counterfactual is informed as an input value, the network learns to generate class-specific counterfactuals. Thus, the blue blocks in Figure 20 are able to identify the properties that differentiate each class from one another in a more accurate way than the other techniques, and apply them to the original values to change their class. Thus, the obtained counterfactuals are more interpretable, as they are closer to the corresponding class data manifold.

Regarding sparsity, Table 23 shows that, on average, the proposed technique needs to alter 3.28 input features to generate a counterfactual, leaving the rest unaltered. On the other hand, the two compared techniques require more than 7 input features to generate a counterfactual. This is because sparsity is directly enforced in the loss function of the proposed technique, whereas this does not occur in the other two techniques. While there is a $\|X - X'\|_1$ in both techniques (1) and (2), this does not ensure sparsity is been sought. Instead, it only ensures the distance between the two values is minimized, regardless of the number of variables being altered. Numerically, a small deviation in all of the input features gives the same result as if the total deviation is allocated in only one input feature. Thus, sparsity is not necessarily being enforced. However, with the $\|X - X'\|_0$ term in the proposed technique, the model is penalized for each altered feature, thus, enhancing sparsity. Figure 35 shows the distribution of altered features to generate a counterfactual. The information in

the figure shows that for the SCF-Net, the distribution of altered input features is much more left-skewed than the other two distributions. This supports the information shown in Table 23 about sparsity. Through the proposed technique, approximately 60% of the generated counterfactuals modify up to three input features, whereas this percentage is far less in techniques (1) and (2). In the case of technique (1), almost 60% of the counterfactuals require the modification of all 13 features. While this improves in technique (2), still a large number of counterfactuals require a high number of altered features. This shows that the $\|X - X'\|_0$ term in the SCF-Net's loss function forces the model to find sparser solutions successfully.

Besides evaluating the quality of the generated counterfactuals by each of the presented techniques, each of the models' performance is evaluated. According to Table 23, the accuracy of the three models is almost the same, with only a 0.23% difference. Since the model used for techniques (1) and (2) is a vanilla neural network, the results show that the proposed SCF-Net does not fall within the accuracy/interpretability tradeoff (BARRAZA; DROGUETT; MARTINS, 2021), as it increases interpretability in neural networks without affecting performance. CF accuracy and generation rate are the accuracy of the successfully generated counterfactuals when fed to the network and the proportion of counterfactuals that are successfully generated. Results show that the SCF-Net achieves higher CF accuracy and CF generation rate. When multiplying the two values, it can be noted that the SCF-Net successfully generates more correctly-classified counterfactuals than both techniques (1) and (2). Furthermore, it can be noted that the SCF-Net guarantees the generation of a counterfactual for every value fed to the network.

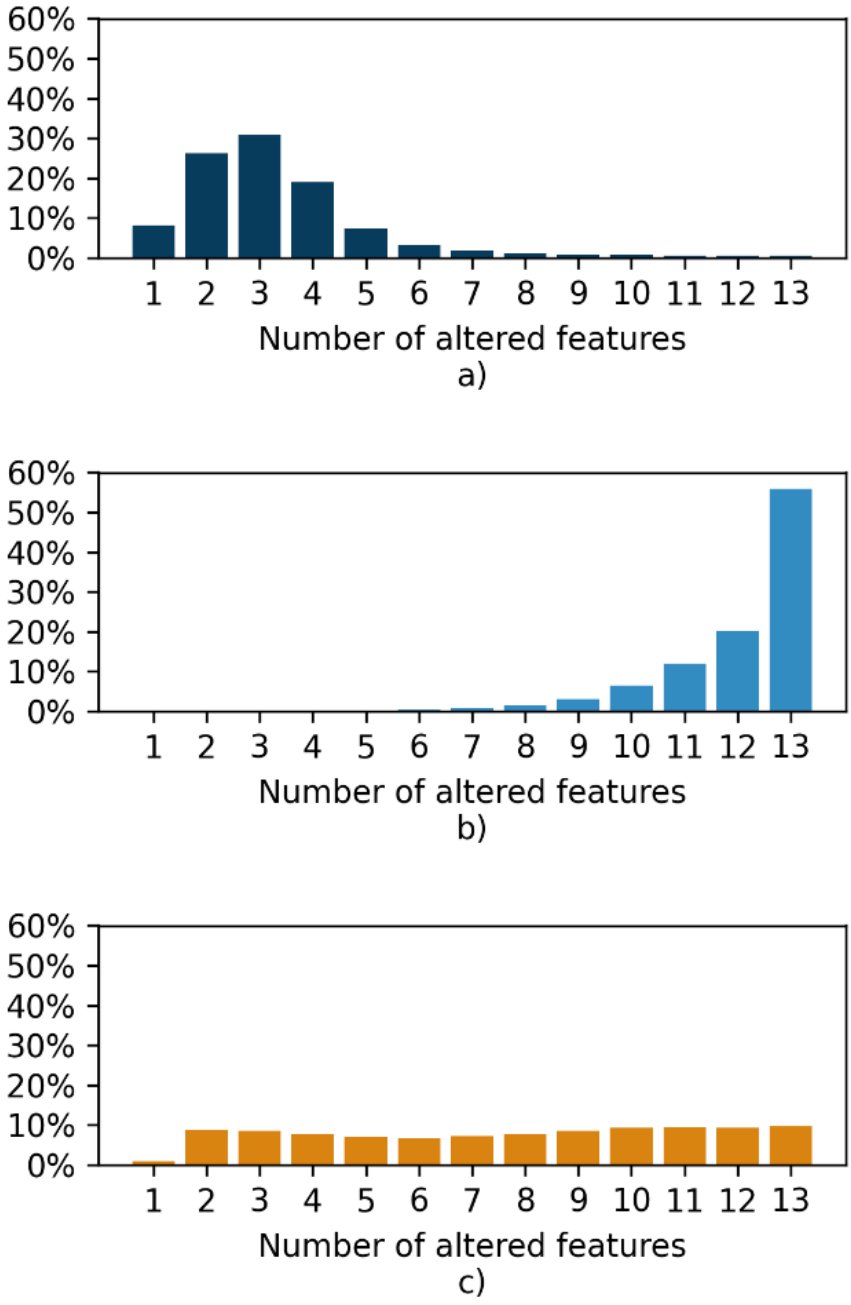

Figure 35 - Distribution of number of features needed for counterfactual generation in case study 1, for a) SCF-Net, b) technique (1) and c) technique (2).

Table 24 shows the time it takes for the three techniques to train, predict and generate a counterfactual. Regarding training time, it can be seen that the SCF-net takes almost ten times more time to train per epoch than the vanilla neural network used for the two other techniques. This is due to the number of parameters used in the SCF-Net. The complexity of the network due to the multi-task configuration increases the number of parameters used for training. Furthermore, for the SCF-Net training process, the training dataset is augmented.

For each datapoint, a counterfactual is calculated for each available class. Thus, the training set is increased according to the number of classes to fit this scenario. Since there are four classes in this case, the dataset used for training is four times the original dataset.

Regarding prediction time, the same tendency as the training time can be seen. This is due to the complexity and size of the architecture. However, a notable difference can be seen in the counterfactual generation time. The SCF-Net takes four orders of magnitude less time than techniques (1) and (2) to generate a single counterfactual. Since the SCF-Net is inherently interpretable, the generation of a counterfactual comes from the prediction process itself. Thus, the counterfactual generation time for the SCF-Net is calculated as the difference between the prediction time and the prediction time of a vanilla neural network. However, it could be argued that the whole prediction time is the counterfactual generation time, as predictions and counterfactual generation are done simultaneously. In this case, still the tendency is kept, with the counterfactual generation time being much less than its counterparts. On the other hand, techniques (1) and (2) generate counterfactuals by solving an optimization problem, which is naturally more demanding than a neural network prediction.

Table 24 – Time consumption values for all techniques, water injection pump case study.

Technique	Training time [s/epoch]	Prediction time [s/datapoint]	Counterfactual generation time [s/datapoint]
SCF-Net	9.90×10^{0}	6.16×10^{-5}	3.19×10^{-5}
Wachter et. al. (WACHTER; MITTELSTADT; RUSSELL, 2017) (1)	7.65×10^{-1}	2.97×10^{-5}	3.74×10^{-1}
Counterfactuals Guided by Prototypes (2)	7.65×10^{-1}	2.97×10^{-5}	5.22×10^{-1}

5.2.2. Case Study 2: Amine Treatment Plant

Table 25 shows the different metrics used to evaluate the counterfactuals generated using each of the analyzed techniques. Results show how the proposed technique generates better IM1 and sparsity values than the two other techniques. This means that the counterfactuals generated by the proposed technique are closer to the data manifold of the

counterfactual class than that of the original class. Also, it means that less altered features are needed in order to generate each counterfactual. According to Table 25, technique (1) generates counterfactuals with the best Realism metric. This means that the counterfactuals generated by (1) are close to the data manifold over all classes. The reason why the proposed technique does not achieve a better value than technique (1) is because of the autoencoder trained with the whole train set $AE()$. This autoencoder is trained in an unsupervised manner, thus it is not trained to describe class-specific data manifold, only the whole dataset data manifold. This can lead to the autoencoder to describe the whole dataset manifold roughly, but fail to describe in detail the class-specific data manifolds. Since the counterfactual class is fed to the proposed technique's model as an input, the network may learn to describe each of classes' space more accurately than $AE()$. Thus, it could happen that some of the counterfactuals generated by the proposed model are not successfully reconstructed by $AE()$, as they would not be a part of the autoencoder's limited description of the data manifold. While this explains why the proposed technique has a lower realism value, it also explains why it has a higher IM1 value, as it evaluates each counterfactual's ability to fit in its specific class data manifold, described by $AE_t()$. This confirms that the proposed technique is able to describe each class' data manifold more accurately than the other two techniques.

Table 25 - Counterfactual quality evaluation metrics for the amine treatment plant case.

Technique	IM1 (↓)	Realism (↓)	Sparsity (↓)	Model Accuracy (↑)	CF Accuracy (↑)	CF generation rate (↑)
SCF-Net	1.60×10^{3}	7.68×10^{5}	2.37	91.71%	99.96%	100.00%
Wachter et. al. (WACHTER; MITTELSTADT; RUSSELL, 2017) **(1)**	4.62×10^{3}	2.94×10^{5}	8.97	92.35%	50.65%	79.27%
Counterfactuals Guided by Prototypes (2)	3.82×10^{3}	1.72×10^{6}	4.78	92.35%	99.70%	90.88%

Regarding sparsity, results show that the counterfactuals generated by the proposed technique are more likely to be understood by a human, as less variables are altered to generate a valid counterfactual. Figure 36 shows a detailed breakdown of the results into number of altered features needed for generating a counterfactual. It can be noted that, when using the proposed technique, approximately 60% of the generated counterfactuals needed a modification of up to two input features. Furthermore, approximately 80% of the

counterfactuals needed up to three features to be altered. In contrast, techniques (1) and (2) show results that do not favor understanding. In the case of technique (1), almost all counterfactuals need a variation of all nine features. For technique (2), results are distributed evenly across all variables, with a slight predominance of counterfactuals needing four feature modification for being generate. Like detailed in the previous case study, this is due to the $\|X - X'\|_0$ term in the loss function in the proposed model, which enforces the generation of sparse solutions.

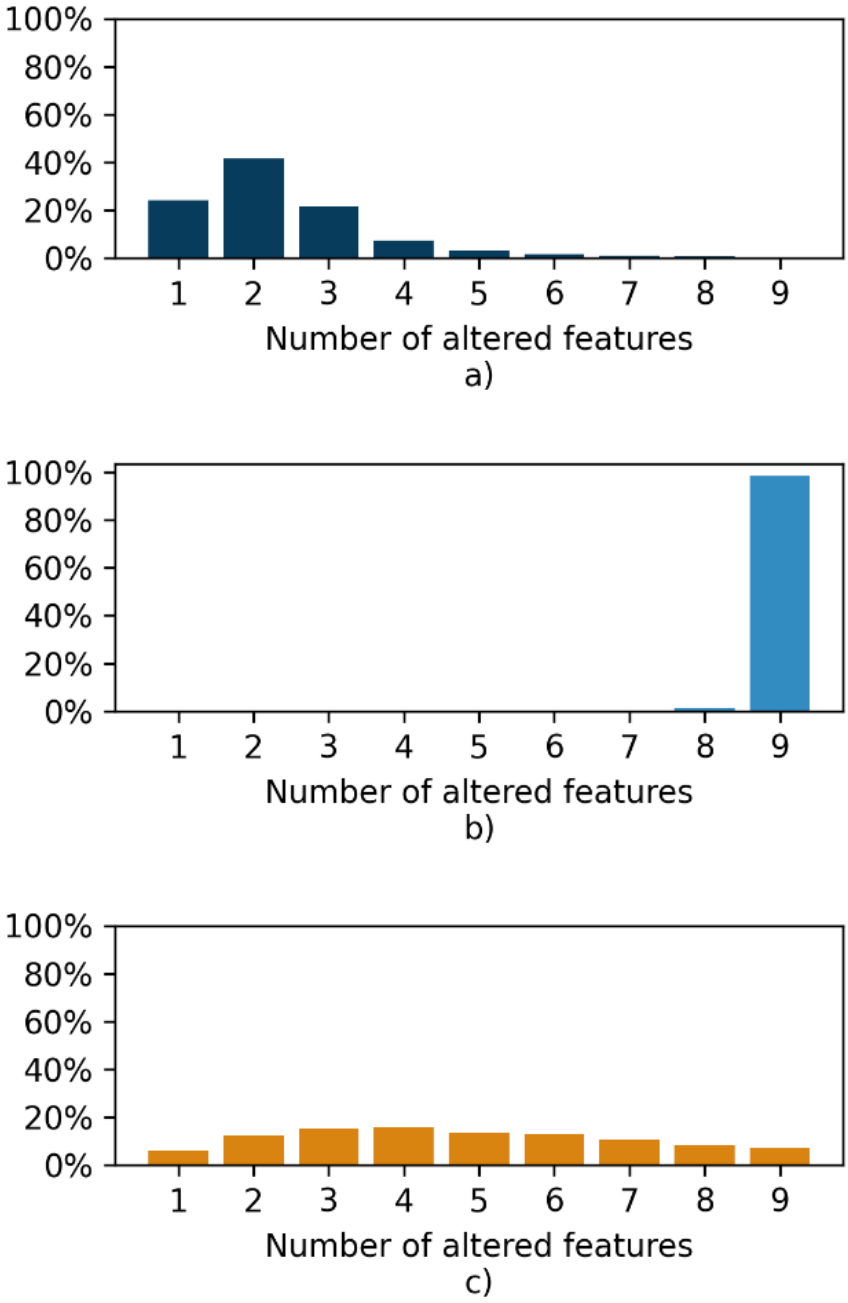

Figure 36 - Distribution of number of features needed for counterfactual generation in case study 2, for a) SCF-Net, b) technique (1) and c) technique (2).

Regarding accuracy, results in Table 25 show similar results than in the previous case study. When using a SCF-Net, performance is kept almost the same, with only a 0.64% difference with a vanilla neural network. This confirms that the SCF-Net successfully bypasses the accuracy/interpretability tradeoff. Regarding CF accuracy, SCF-Net presents higher accuracy than techniques (1) and (2). Furthermore, there is a considerable difference between the model accuracy and the CF accuracy. This occurs because the SCF-Net receives

a desired class value as input. The network is trained in order to create valid counterfactuals even in multiclass scenarios. Thus, a high accuracy value is expected. Regarding CF generation rate, the SCF-Net presents a 100.00% success rate. This is also expected, as a neural network generates a valid output for each valid input fed. This is not the case with techniques (1) and (2), where each counterfactual is generated through a separate optimization problem with a limited number of search steps. Thus, it is possible that not all counterfactuals are successfully generated.

Table 26 shows the time consumption values for all three techniques. It can be seen that due to the number of parameters in the SCF-Net, training takes longer than the other two techniques, in which a vanilla neural network is used. However, values are under one second, and considering that 2000 epochs where necessary for training, the total amount of time used for training is not a restriction. The same tendency is presented in the prediction time, where the SCF-Net takes an order of magnitude more than the vanilla neural network to make a prediction. However, this value still is not a restriction towards application. Regarding counterfactual generation time, it can be seen there is a notable difference between the SCF-Net and the two techniques used for comparison. The SCF-Net takes approximately four orders of magnitude less time than the other techniques. This is due to the fact that the SCF-Net generates counterfactuals during prediction. Indeed, the counterfactual generation time for the SCF-Net is calculated as the difference between its prediction time and the prediction time of a vanilla neural network. Regarding the two other techniques, they generate counterfactuals through the solving of an optimization problem

Table 26 – Time consumption values for all techniques, amine treatment plant case study.

Technique	Training time [s/epoch]	Prediction time [s/datapoint]	Counterfactual generation time [s/datapoint]
SCF-Net	4.42×10^{-1}	1.22×10^{-4}	5.40×10^{-5}
Wachter et. al. (WACHTER; MITTELSTADT; RUSSELL, 2017) (1)	5.52×10^{-2}	6.63×10^{-5}	4.83×10^{-1}
Counterfactuals Guided by Prototypes (2)	5.52×10^{-2}	6.63×10^{-5}	5.15×10^{-1}

5.3. FS-SCF

In this subsection, results for the FS-SCF net are presented and discussed. Like in the previous framework, the amine treatment plant and the water injection pump case studies are used for evaluation. Hardware specifications are the following: Intel i5-9600K CPU, 16 GB DDR4 RAM and NVIDIA 11 GB Geforce RTX 2080 Ti GPU. Like in the SCF-Net framework, only real-world industrial applications were used, namely, the water injection pump and amine treatment plant cases.

5.3.1. Case Study 1: Water Injection Pump

Table 27 shows the evaluation metrics for the FS-SCF network and how they are compared to other techniques for counterfactual generation. Results show the FS-SCF net having similar metric values to the SCF-Net, except on realism. According to the IM1 metric, the counterfactuals from the FS-SCF net are similarly interpretable to those generated by the SCF-Net, with only 0.02 difference in their values. Compared to techniques (1) and (2), the FS-SCF generates more interpretable counterfactuals according to IM1, as is the case with the SCF-Net. This is not the case with the realism metric. It can be noted that the FS-SCF net achieves a better realism metric than technique (1), but not better than (2) nor SCF-Net. With the use of the FS layer, the counterfactuals generated by the network are different from the training data manifold that considers all classes, as the reconstruction of the counterfactual by the $AE()$ autoencoder is less similar to the original value than those using the SCF-Net and (1) (see equation (40)). However, this does not translate in a loss in the IM1 metric nor in the CF accuracy metric. Actually, the latter sees a slight improvement. Thus, the realism metric is not aligned with the accuracy of the generated counterfactuals, as it occurs with the IM1 metric. In this case, a lower value for the IM1 metric is followed by a higher value in the generated counterfactuals' accuracy. In this regard, the realism metric is a less informative metric than the IM1. The $AE()$ autoencoder used for calculating realism captures the representation for the whole data manifold, failing to detect the differences between classes. In a task where counterfactuals are being generated, it is relevant to capture these

differences, as is the case with the IM1. Although the calculation of the metric needs the training of one autoencoder for every class in the dataset (in contrast to one autoencoder needed for the realism metric), IM1 is a more elaborate metric that gives more useful information. By training class-specific autoencoders, interpretability can be measured according to the two classes (the original and the counterfactual), rather than an average behavior obtained by $AE()$. Furthermore, IM1 is likely related to the counterfactual accuracy, because a counterfactual closer to the data manifold of the counterfactual class will have an increase probability of being classified as that class, thus being a valid counterfactual.

Other metrics from the table below show that the results achieved by the FS-SCF network are similar to those from the SCF-Net. Sparsity shows a slightly better result, while the model accuracy presents a 0.05% decline compared to the SCF-Net. When passing the generated counterfactuals through the network, their accuracy is 0.14% higher than those generated using the SCF-Net. This similarity in results is explained by the fact that the FS-SCF network has a few number of parameters added with respect to the SCF-Net, one per input feature. Thus, the models are similar to each other and similar results were expected.

Table 27 - Counterfactual quality evaluation metrics for the water injection pump case, including FS-SCF network.

Technique	IM1 (↓)	Realism (↓)	Sparsity (↓)	Model Accuracy (↑)	CF Accuracy (↑)	CF generation success rate (↑)
FS-SCF Net	1.33	4.67×10^{5}	3.06	97.92%	99.64%	100.00%
SCF-Net	1.31	5.60×10^{4}	3.28	97.97%	99.50%	100.00%
Wachter et. al. (WACHTER; MITTELSTADT; RUSSELL, 2017) **(1)**	2.58	3.74×10^{5}	12.08	98.20%	88.28%	35.43%
Counterfactuals Guided by Prototypes (2)	2.15	1.51×10^{7}	7.67	98.20%	95.40%	67.99%

Figure 37 shows a comparison between the SCF-Net and the FS-SCF network regarding sparsity. Results in Table 27 show that the FS-SCF net generates slightly sparser counterfactuals than the SCF-Net. This is confirmed by the figure below. Both distributions are very similar, however, there is a slight difference in two and three altered features.

Whereas the counterfactuals generated by the SCF-Net mostly need three features to be altered followed closely by two, it is the opposite with the FS-SCF network. Furthermore, a higher number of counterfactuals need four altered features in the SCF-Net than in the FS-SCF network. Since the difference is not considerable, the mean values presented in Table 27 are very close too.

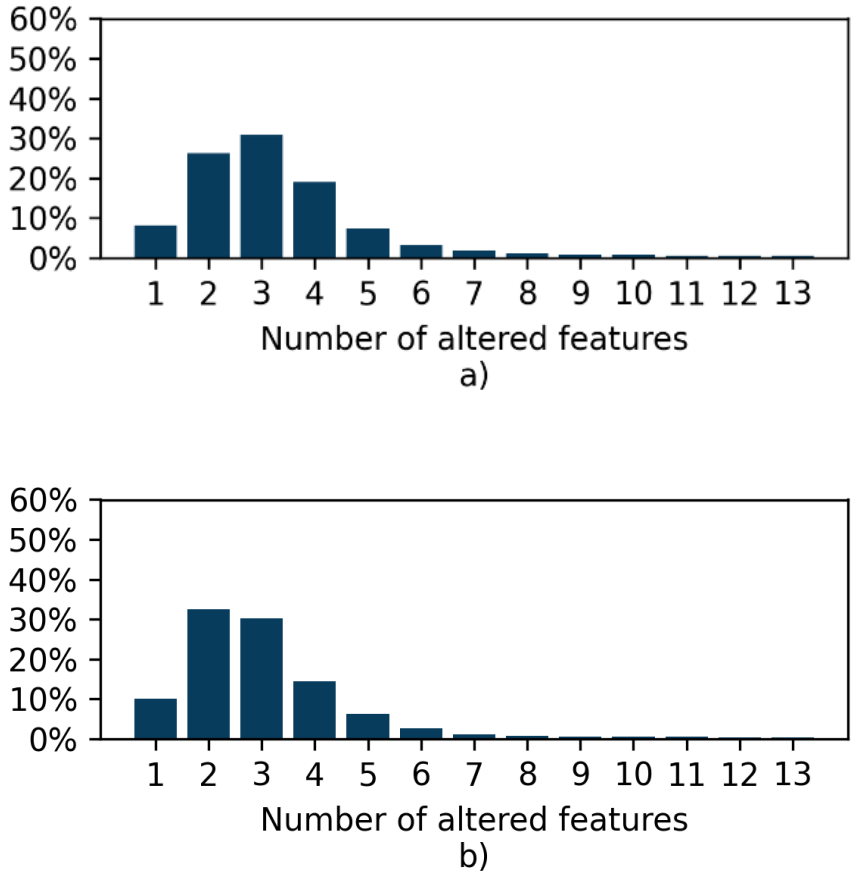

Figure 37 – Distribution of number of features altered for counterfactual generation in case study 1, comparison between a) SCF-Net and b) FS-SCF net.

Table 28 shows the correlation between the ranking obtained using the values from the FS layer and those using causality measures such as necessity, sufficiency, NaS and NoS. Results show that all four correlation values have an associated p-value lower than 0.05. Thus, there is a high level of significance of the results at a 0.05 level. This means that it is highly unlikely for the obtained correlations to be generated out of chance (XIAO et al., 2016). Out of the four correlation coefficients, the one with highest value is the one corresponding to necessity, with a value of 0.8285. This shows that, when using an FS layer

jointly with the SCF-Net configuration, the FS layer will most likely give more importance to those features necessary to determine the output of the model. This addresses the issue presented in (KOMMIYA MOTHILAL et al., 2021), since there is a connection between the feature selection method and the counterfactual generation technique, thus successfully unifying both. Furthermore, the correlation with the rest of the causality-based rankings also presents high values, strengthening the argument. The NoS ranking presents a correlation to the FS layer ranking very similar to the necessity ranking, above 0.82. The NaS and sufficiency rankings, despite having lower values, still present a high correlation value with a low p-value. The difference between necessity and sufficiency values indicates that necessary features are not always sufficient, and vice versa. This is a normal situation, as an explanation may need more than one feature to explain its output. In such a situation, all of the required features will have their necessity value equals to 1. However, none of them will be sufficient, as every one of them alone will not be enough. The opposite case also may occur. In a case where two features are independently sufficient to explain an output value, they will not be necessary. Thus, only when the outputs are explained by one necessary and sufficient feature value, the two corresponding rankings will be the same. In every other case, the rankings will be different and will have different correlation values with the FS-layer ranking.

Table 28 – Spearman correlation between causality-based rankings and FS-layer ranking, water injection pump case.

Ranking	Correlation coefficient with FS-layer ranking	p-value
Necessity	0.8285	6.50×10^{-4}
Sufficiency	0.6307	3.48×10^{-2}
Necessity or Sufficiency (NoS)	0.8219	6.90×10^{-4}
Necessity and Sufficiency (NaS)	0.6641	1.53×10^{-2}

Table 29 shows the time values for training the FS-SCF network, obtaining a prediction, generate counterfactuals and calculate necessity and sufficiency values for each datapoint. Regarding the first three, values are similar to the SCF-Net, as expected. This is because the architectures are very similar, with the only difference being the FS layer. Regarding the causality-based values, the FS-SCF network takes 1.04 seconds to calculate necessity and sufficiency. This is because several combinations of values must be tested, thus, several predictions are made to obtain these values.

Table 29 – Time consumption values for the FS-SCF framework, water injection pump case.

Technique	Training time [s/epoch]	Prediction time [s/datapoint]	Counterfactual generation time [s/datapoint]	Causality-based values calculation time [s/datapoint]
FS-SCF	8.18×10^{0}	5.76×10^{-5}	2.79×10^{-5}	1.04×10^{0}

To analyze the results obtained for the FS-SCF net in more detail, two examples of separate training processes are shown. Table 30 shows the causality-based values for every feature. Results show that features 5 and 9 have an increased importance across all causality values, while the rest of the features have low values. It can be noted that features 0,1,3,6,8,11 and 12 never were sufficient causes for an output value. This creates an issue as the ranking has seven out of 13 features with the same value. While the Spearman correlation can be calculated in a situation where values are repeated, this is not as informative as to having different values for each feature. This issue also occurs with the NaS value. Thus, the importance of calculating not one but several causality-based values. In this scenario, the most informative rankings are the necessity and NoS. These are useful to compare with the FS-layer ranking and, consequently, obtain a more trustworthy correlation.

Table 30 – Causality-based values for each feature, example #1 for water injection pump case.

Feature	Feature Tag	Necessity %	Sufficiency %	NoS %	NaS %
0	**FIT-323D**	2.32	0	2.32	0
1	**IT-1031**	2.07	0	2.07	0
2	**PT-305D**	3.00	0.51	3.49	0.03
3	**PT-306D**	2.07	0	2.07	0
4	**TIT-305D**	6.27	0.02	6.27	0.02
5	**TIT-306D**	29.26	26.65	44.34	11.57
6	**TT-308D**	2.32	0	2.32	0
7	**VT-301D**	3.70	0.02	3.71	0.01
8	**VT-302D**	2.07	0	2.07	0
9	**VT-303D**	69.69	70.15	88.89	50.95
10	**VT-304D**	3.01	0.02	3.02	0.02
11	**ZT-301D**	2.12	0	2.12	0
12	**ZT-302D**	2.24	0	2.24	0

Performance evolution according to each feature ranking is shown in Figure 38. Results show that the necessity, NoS and FS rankings have similar RQS values and higher

than the sufficiency and NaS rankings. This indicates that besides having a strong correlation, the first three rankings mentioned above are the ones that reach a maximum level of performance with less features. The image shows that with 10 out of 13 features, performances reaches its maximum level, being able to discard the three last features with almost no loss in performance (features 1, 3 and 8, according to Table 30). However, this is not the case with the sufficiency and NaS rankings. The sufficiency and NaS rankings present a different evolution from the rest of the rankings, which translates in lower RQS values. When analyzing jointly with the results in Table 30, it can be noted that this behavior is related to the fact that the sufficiency and NaS rankings are less informative than the others due to the repeated values. This makes the performance evolution suboptimal when following these rankings. It can also be noted that, despite features 9 and 5 presenting considerably higher causality-based values than the rest of the features, more features are needed to reach the maximum level of performance. This is due to the fact that features 9 and 5 do not explain every output. Looking at the NoS value, it can be noted that 88.89% of the outputs can be explained involving feature 9. This means that 11.11% of the outputs are not explained by that feature. In turn, this shows that feature 9 alone is not enough to explain 11.11% of the outputs, justifying the loss in performance shown in the figure below when only using feature 9.

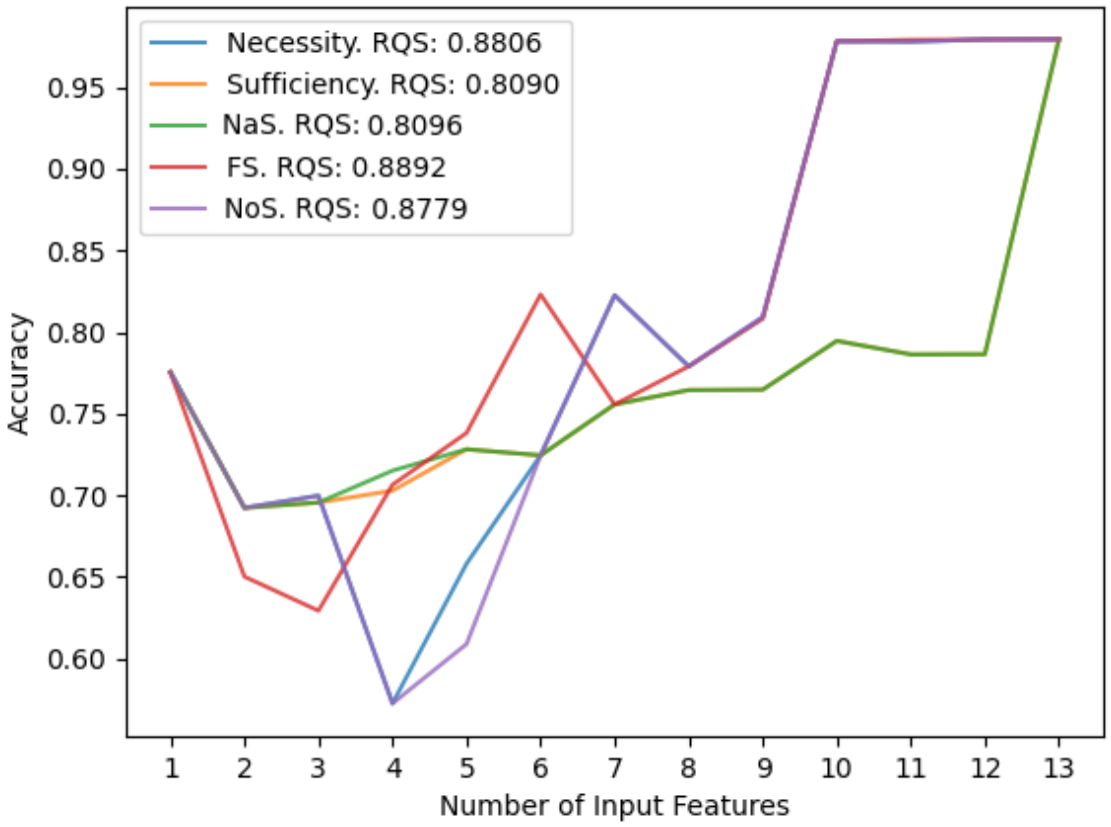

Figure 38 – Accuracy evolution following different feature rankings, with their corresponding RQS value, example #1 for water injection pump case.

Figure 39 shows the relation-between-features (RBF) map for the first example. This is useful to better visualize the connections between features. Results show that features 9 and 5 are highly related to the output feature. This is in line with the NoS value for these features. After those two features, feature 4 is the one with the strongest connection with the output. Looking at the connections between input features can be noted that there is a strong relation between features 9 and 5. This indicates that the two features are simultaneously necessary with a higher frequency than the rest of the features. The thinner lines between the rest of the features indicates a weaker connection between them. However, there is a connection between every input feature, which shows that there are some values whose minimal explanation set (MES) contains one of these features. A minimal explanation set is composed of the minimal set of features that explain an output. If a feature value alone is a sufficient explanation, then it is a minimal explanation set, as no more feature values are needed for explaining the output. If several features are necessary, those features are considered to be the minimal explanation set.

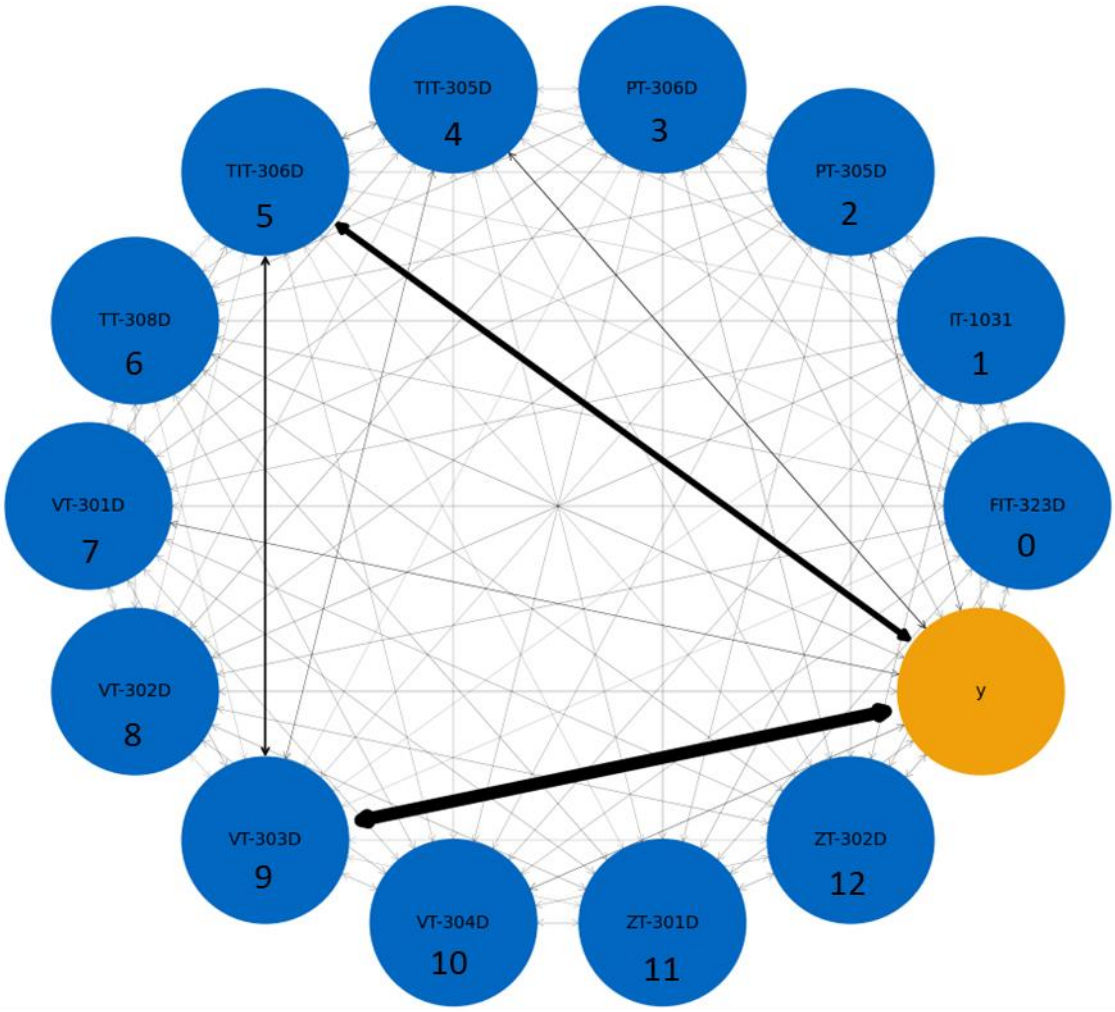

Figure 39 – Relation-between-features (RBF) map, example #1 for water injection pump case. Blue circles represent input features and the orange circle represents the output feature. The width of the connecting arrows between an input feature and the output is determined by the number of times the feature is necessary of sufficient for explaining the output. The connecting arrows between input features is determined by the number of times both features are simultaneously necessary for explaining the output.

Figure 40 shows the top-15 most frequent MES. It can be seen from the image that almost 60% of the MES correspond to feature 9 alone. Next is feature 5, with approximately 25%. Third place is for [5,9]. This verifies what is shown in the RBF map above regarding the relation between features 5 and 9. After that, comes [4], [4,5,9] and [2]. However, after [2], there are no combinations of feature 2 and 4, 5 or 9. This shows there is no linear combination or pattern that determines the MES. Furthermore, it can be noted that the set with all the features is a MES, although with a low frequency. This is why in the previous image there are connections between every input feature, because at one point they were all necessary for an explanation.

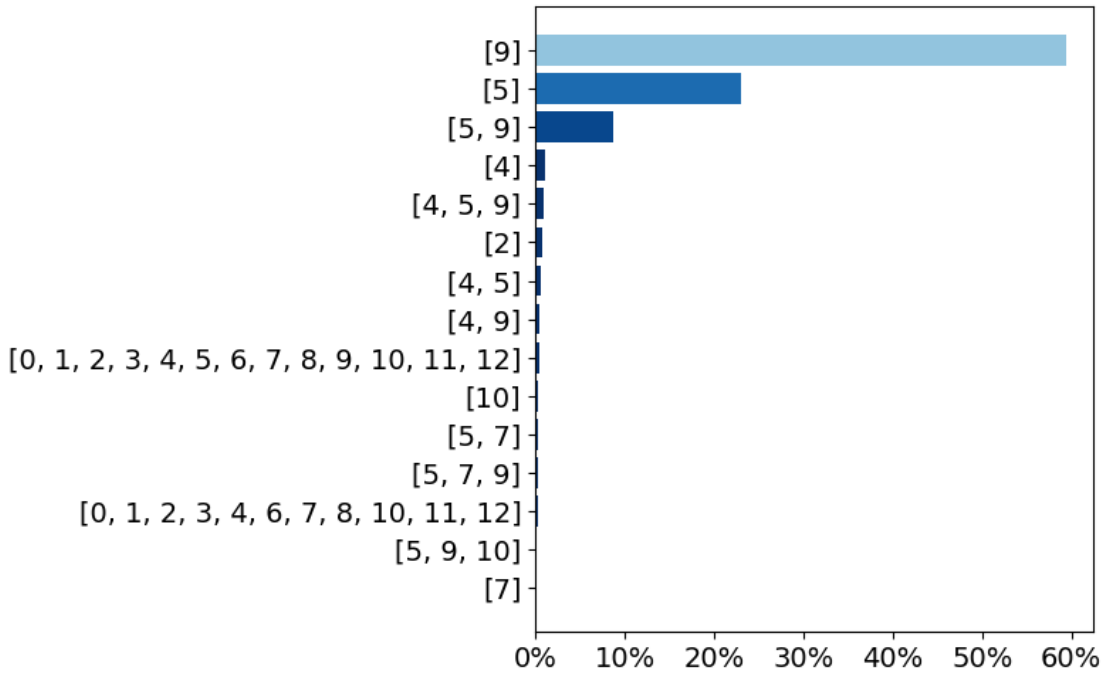

Figure 40 – Top-15 most frequent minimal explanation sets, example #1 for water injection pump case.

Like in the previous example, Table 31 shows the obtained causality-based values for example #2. Results show similar values with the previous example. Features 9 and 5 are the most relevant according to the causality-based values, like in the previous example. However, there is an ambiguity regarding features 4 and 10. According to necessity and NoS rankings, feature 10 is more important than feature 4. However, the opposite occurs when looking at sufficiency and NaS. There is no correct answer, as the rankings are obtained according to different criteria. Moreover, the same issue with the repeated values shown in the previous example occurs in this example. Thus, sufficiency and NaS rankings are less informative than the others are.

Table 31 – Causality-based values for each feature, example #2 for water injection pump case.

Feature	Feature Tag	Necessity %	Sufficiency %	NoS %	NaS %
0	**FIT-323D**	2.09	0	2.09	0
1	**IT-1031**	1.97	0	1.97	0
2	**PT-305D**	2.34	0.01	2.35	0
3	**PT-306D**	1.97	0	1.97	0
4	**TIT-305D**	4.65	0.89	4.92	0.62
5	**TIT-306D**	30.49	26.43	45.31	11.61
6	**TT-308D**	2.51	0	2.51	0
7	**VT-301D**	2.35	0.01	2.36	0
8	**VT-302D**	1.97	0	1.97	0
9	**VT-303D**	69.74	66.15	84.81	51.08
10	**VT-304D**	6.21	0.01	6.21	0
11	**ZT-301D**	2.02	0	2.02	0
12	**ZT-302D**	2.12	0	2.13	0

Figure 41 shows the performance evolution when following the rankings presented above. As expected, sufficiency and NaS rankings present the lowest RQS values. This is because they are less informative. The only way they could achieve a better RQS is randomly on the features with the same value, which is unlikely and not representative of the underlying process. For the other rankings, it can be noted that, like in the first example, accuracy does not vary when using more than 10 variables.

From the figure below, it can be also noted that the FS ranking achieves a higher RQS than the necessity and NoS rankings. Although performance reaches a valley when using six features and following the FS ranking, it increases at a faster rate when adding more features. This shows the capabilities of the FS layer to find a combination of weight values that maximizes performance, as it is a part of the FS-SCF network involved in the classification process.

Looking at the performance evolution curves, it can be seen that the necessity, NoS (which are the same curve, see Table 31) and FS rankings follow similar behavior. This is in line with the results shown in Table 28 indicating a strong correlation between the two rankings.

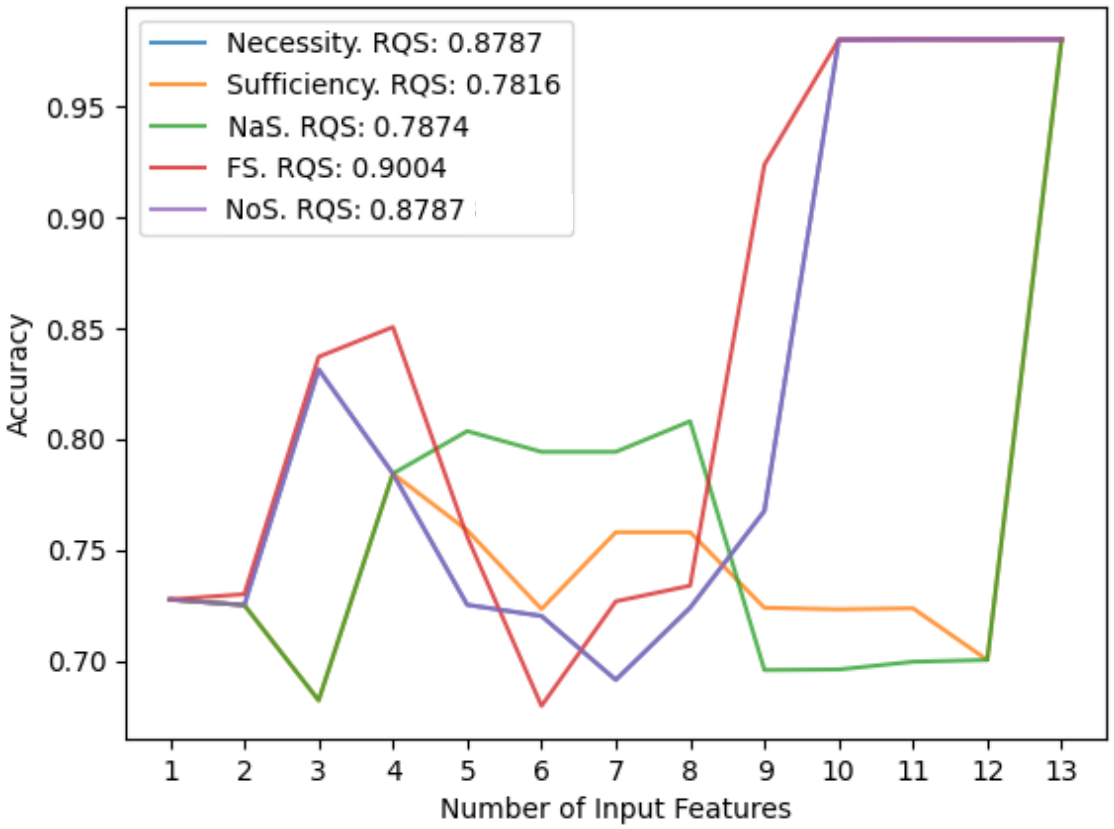

Figure 41 – Accuracy evolution following different feature rankings, with their corresponding RQS value, example #2 for water injection pump case.

Figure 42 shows the RBF map for the second example. Results are similar to those in the previous example, with a strong relationship between features 9, 5 and the output feature. Regarding the rest of the features, as in the previous example, the relationships are not as strong, indicating they are not as present in the explanation process.

The similarities between and Figure 39 and Figure 42 suggest there is a stability in the training process. It indicates that the network consistently finds the same path towards maximizing performance, regardless of the initial weights. This is due to the quality of the data, the size of the dataset and the task itself. Deep neural networks obtain better results when a large dataset is being used. Furthermore, the quality of the data is also key. If there is too much noisy data, faulty sensors, or missing data that must be repaired by interpolation techniques, performance will most likely be affected. Also, the task itself must be suitable. Deep neural networks will not find useful associations or patterns within the data if there is no existing association. Thus, the problem definition is key. In this case, the health state of a pump is diagnosed by using the sensor data from the system itself. Features such as temperature, pressure and vibrations contain information about the health state of the system.

Thus, it seems appropriate to use deep learning techniques to address the given task. However, features like the engine current (feature 1) are not useful for the task. In the two examples, the four causality-based values for feature 1 are the minimum when compared to the rest of the features. Furthermore, Figure 38 and Figure 41 indicate that the feature can be discarded without performance loss. This is consistent with the problem's nature. The current feature can be used as a proxy to determine whether the system's power is on or off, but not to diagnose failures. As shown in this case study, other features are used successfully instead. Through the use of feature selection, counterfactuals and causality-based values, it is possible to analyze the trained model and obtain valuable information, rather than just trying to maximize performance. If the results showed an unusual pattern, corrective actions could be taken. For example, if the model's results where highly reliant on the engine current feature, further analysis should be made, as the nature of the problem suggests that this feature is not appropriate for diagnosing failures. If after a thorough analysis, the apparently unusual behavior proves not to occur due to a mistake, the use of interpretability tools could be useful for learning from the trained model. However, the need of an expert opinion is important to determine the difference between an error in the model and a novel unseen insight.

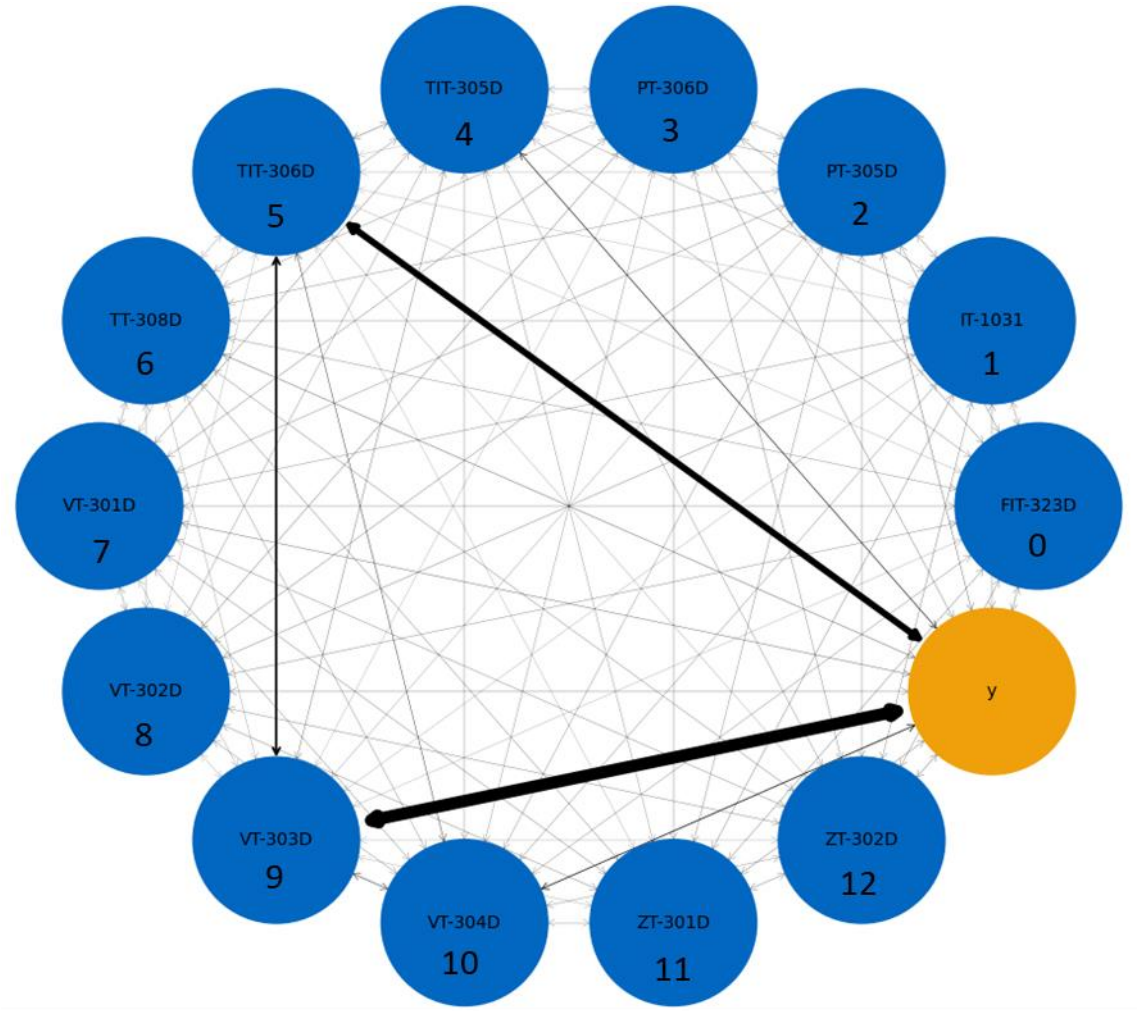

Figure 42 – Relation-between-features (RBF) map, example #2 for water injection pump case.

Figure 43 shows the top-15 MES for the second example. Results show that the top-3 MES are in the same order as those shown in Figure 40, accounting for more than 95% of all MES.

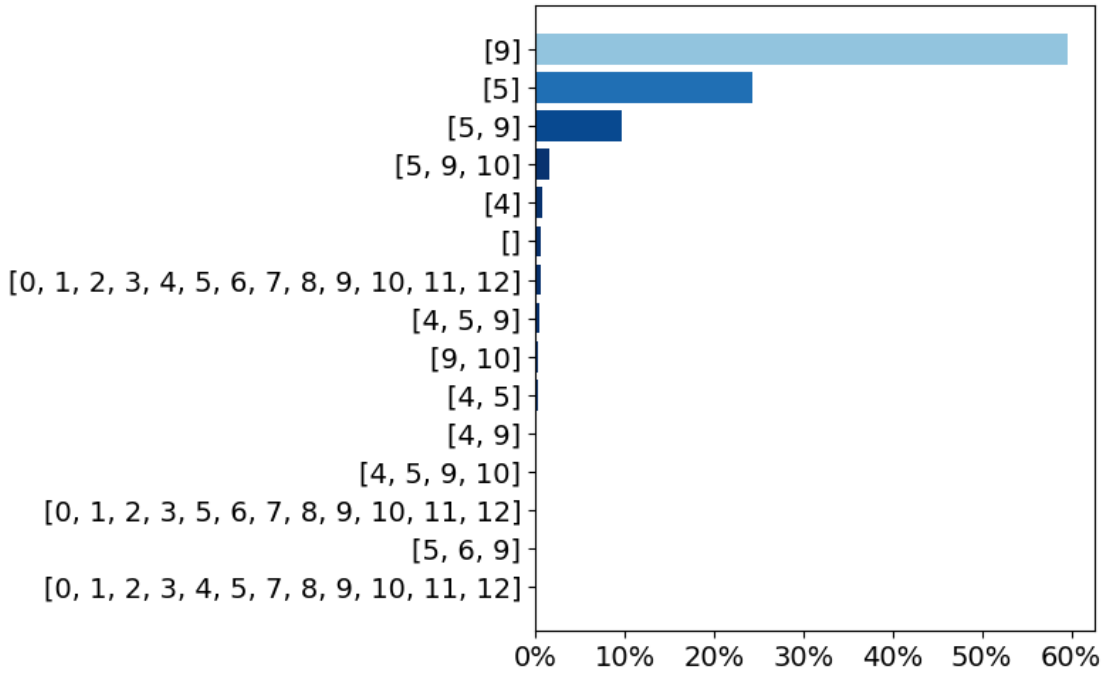

Figure 43 – Top-15 most frequent minimal explanation sets, example #2 for water injection pump case.

5.3.2. Case Study 2: Amine Treatment Plant

Table 32 presents the results obtained with the FS-SCF network, compared to the SCF-Net results and techniques (1) and (2) described before. It can be seen from the table that sparsity, CF accuracy and CF generation rate values are kept the same as the ones in the SCF-Net. Regarding counterfactual quality, it can be noted that the IM1 value improves in a 59% from the SCF-Net value, while the realism value increases a 68%. The addition of the FS layer to the SCF-Net seems to have influenced the generation of counterfactuals by making them more interpretable regarding class-specific data manifolds. The network is able to recognize more accurately the differences between classes. While the FS layer is not present in the encoder part of the FS-SCF network, it is present in the counterfactual classification part, where weights are copied in order to do a forward pass. While the intention of this forward pass was to ensure and enhance the validity of the generated counterfactuals, in this case interpretability was also influenced by this, due to the use of the FS layer. However, this is not the case for the realism metric. This might be explained by the fact that there is class imbalance and the dataset have few datapoints. A majority of the data

corresponds to the healthy class (i.e. CO_2 < 2500 ppm). Thus, the $AE()$ autoencoder (the one that is not class-specific, see equation (40)) learns a representation of the data manifold based mainly in the healthy data. Because of this, the valid generated counterfactuals will belong mostly to the 'unhealthy' class, which the $AE()$ autoencoder will have more trouble to reconstruct. In a balanced dataset, this would not occur. Thus, the apparent tradeoff between IM1 and realism is explained by class imbalance. This does not occur in the previous case because the dataset is larger. Thus, despite being an imbalance between the classes, each of them is more numerous than this case, and the $AE()$ autoencoder learns from more examples.

Regarding accuracy, it can be noted that it reaches its maximum value, even higher than techniques (1) and (2). For this case, the inclusion of the FS layer generates a compensation on the accuracy loss from the SCF-Net. However, the relative difference between the FS-SCF and SCF-Net is of 1.12%, which is not a significant value. It can be argued that performance is kept at a same level. On the other hand, a 1% increase is not sufficient to conclude that the FS-SCF network will always be more accurate than the SCF-Net. Thus, regarding the accuracy-interpretability tradeoff mentioned in Chapter 2, this case's results indicate that interpretability is increased without performance loss, thus addressing the tradeoff issue.

Table 32 - Counterfactual quality evaluation metrics for the amine treatment plant case, including FS-SCF network.

Technique	IM1 (↓)	Realism (↓)	Sparsity (↓)	Model Accuracy (↑)	CF Accuracy (↑)	CF generation rate (↑)
FS-SCF Net	6.55×10^{2}	2.42×10^{6}	2.37	92.74%	99.96%	100.00%
SCF-Net	1.60×10^{3}	7.68×10^{5}	2.37	91.71%	99.96%	100.00%
Wachter et. al. (WACHTER; MITTELSTADT; RUSSELL, 2017) **(1)**	4.62×10^{3}	2.94×10^{5}	8.97	92.35%	50.65%	79.27%
Counterfactuals Guided by Prototypes (2)	3.82×10^{3}	1.72×10^{6}	4.78	92.35%	99.70%	90.88%

Regarding sparsity, the distribution for the number of altered features used to generate a counterfactual when using the FS-SCF network is compared to that of the SCF-Net. Results are shown in Figure 44. It can be noted that values are very similar to each other. This is

compatible with the fact that mean sparsity value, as shown in Table 32, is the same for both techniques. In the case of the FS-SCF network, it can be noted that values corresponding to one to three altered features are more equally distributed than those in the SCF-Net. However, the difference is not conclusive.

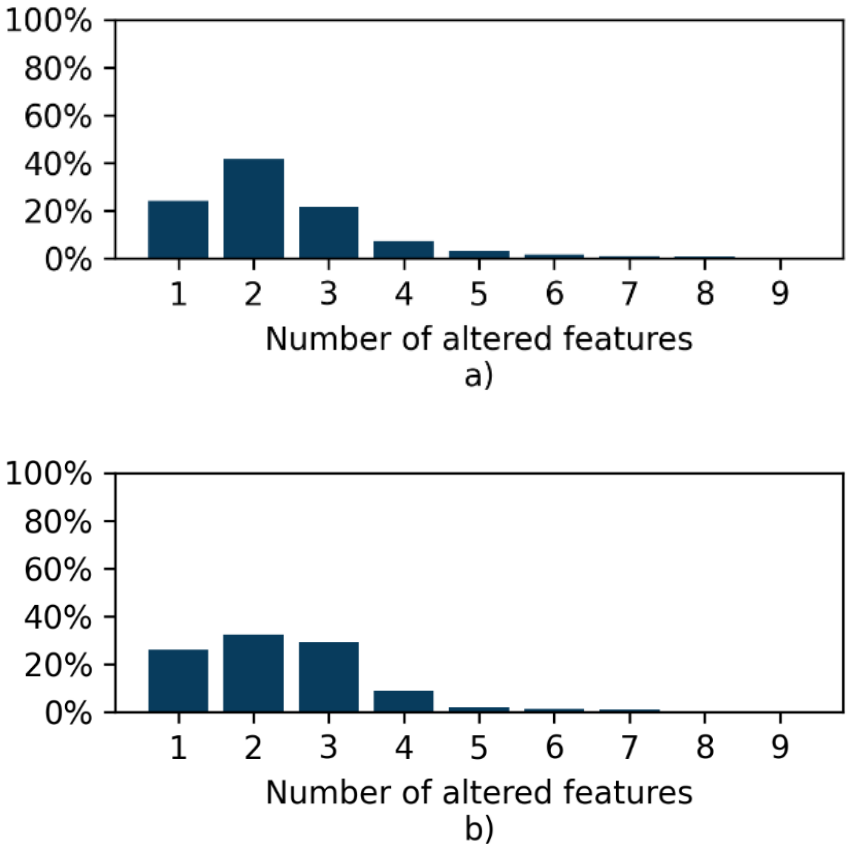

Figure 44 – Distribution of number of features altered for counterfactual generation in case study 2, comparison between a) SCF-Net and b) FS-SCF net

After analyzing the model's performance and the generated counterfactuals, necessity and sufficiency are quantified. With this, features are ranked according to necessity, sufficiency, NoS and NaS and compared to the FS-layer ranking using the Spearman correlation. Results are shown in Table 33. According to the table, all correlation coefficients are above 0.68. This indicates that there is a strong correlation between the FS-layer ranking and each of the causality-based rankings. Also, all of the p-values are below 0.05, except that for the NaS ranking. This leads to statistical significance of the obtained results. Out of the four rankings, the one that is most correlated to the FS-layer ranking is the sufficiency one. However, values are similar to NoS and necessity rankings, with the NaS ranking having a

lower value. All correlations can be considered strong. However, the NaS correlation coefficient is not statistically significant at a 0.05 level. Comparing the results with the previous case study, it can be noted that these correlation coefficients are lower than the previous case.

Table 33 – Spearman correlation between causality-based rankings and FS-layer ranking.

Ranking	Correlation coefficient with FS-layer ranking	p-value
Necessity	0.7652	0.0303
Sufficiency	0.7818	0.0306
Necessity or Sufficiency (NoS)	0.7669	0.0304
Necessity and Sufficiency (NaS)	0.6882	0.0663

Table 34 shows how much time does the FS-SCF network takes to train, predict, generate counterfactuals and calculate causality-based values. As in the previous case, results are similar to the SCF-Net, due to the similarities between the two. Furthermore, to calculate necessity and sufficiency, 0.74 seconds are needed. Like in the previous case, this is due to the several combinations to be tested in order to determine each of the features' necessity and sufficiency. For necessity of one feature, one prediction must be calculated. Thus, in this scenario where there are nine features, a total of 18 combinations must be testes to calculate necessity and sufficiency.

Table 34 – Time consumption values for the FS-SCF framework, amine treatment plant case.

Technique	Training time [s/epoch]	Prediction time [s/datapoint]	Counterfactual generation time [s/datapoint]	Causality-based values calculation time [s/datapoint]
FS-SCF	4.18×10^{-1}	1.02×10^{-4}	4.42×10^{-1}	7.40×10^{-1}

As with the previous case study, two examples are presented from separate training processes. Table 35 shows the causality-based values for example #1. Results show a repetition of several values between features. In the case of necessity, three features (0, 1 and 5) have the same value. For sufficiency, five features have the same value (0, 1, 3, 4 and 5). In the case of NoS, the same features for necessity have the same values, and for NaS, features 0, 1, 3, 4, 5 and 6 have the same value. In general, this affects the information that can be

obtained from these rankings. This is a reason why the highest correlation value according to Table 33 is lower than the highest value in Table 28. It also explains why the p-values are higher. According to Table 33, the sufficiency ranking is the one whose correlation with the FS ranking is the strongest. However, its p-value is not the smallest, with necessity and NoS having slightly better p-values. A possible explanation is what is shown in the table below. The higher the number of repeated values, higher is the probability of the correlation value be obtained as a result of chance rather than an underlying phenomenon. This highlights the importance of calculating more than one causality-based value. Furthermore, the issue of the repeated values is presented here more severely than in the previous case study, where the dataset is larger. Having a larger dataset does not guarantee that the issue will be solved. For example, two features may never be sufficient, regardless of the datapoint being fed. However, a large dataset increases the probability of exploring the whole data manifold and get more results, potentially avoiding the repetition of values in the rankings.

It can be also noted from the table below that the NaS ranking has more repeated values than the NoS ranking. This is due to the fact that the NaS ranking measures two simultaneous conditions, whereas the NoS ranking measures the occurrence of one condition from two possibilities. Thus, generally a feature will most likely be (necessary of sufficient) than (necessary and sufficient).

Regarding the most important features, it can be seen from Table 35 that according to all rankings, feature 2 is the most important. However, necessity and NaS rankings indicate the next most important feature is feature 8, whereas sufficiency and NoS rankings show feature 7 as the second most important one. Feature 2 corresponds to the amine temperature at the reboiler. This means that in this example the trained model uses that feature's values to determine whether the treated gas will have proper amount of CO_2. In (PARK et al., 2017), authors analyze the effect of the reboiler temperature in the CO_2 removal process from biogas using amines. They establish that high temperatures lead to high removal efficiency. This indicates that there is no bias from the trained model towards a misleading feature. This verification process shows the importance of interpretability of black-box models such as deep neural networks. With vanilla neural networks (or CNNs or RNNs), only a performance value is obtained, without knowing why different input values are classified according to their health state. With the use of counterfactuals, causality-based values can be calculated, which give useful insights about the model's use of the different input features. The obtained results can be verified with the previous knowledge about the system, to determine whether

the model uses the input features in a coherent way. Furthermore, these values can be used to further analyze feature selection techniques.

Table 35 – Causality-based values for each feature, example #1 for amine treatment plant case.

Feature	Feature Tag	Necessity %	Sufficiency %	NoS %	NaS %
0	**T_Torre_Stripper**	7.53	0	7.53	0
1	**P_Torre_Stripper**	7.53	0	7.53	0
2	**T_Amina_Reboiler**	82.74	83.08	94.33	71.49
3	**P_Amina_Reboiler**	7.78	0	7.78	0
4	**Fluxo_Gas_A**	7.7	0	7.7	0
5	**Fluxo_Amina**	7.53	0	7.53	0
6	**P_Torre_Contactora**	7.95	0.76	8.71	0
7	**T_gas**	10.91	10.41	19.29	2.03
8	**T_Amina**	12.27	2.96	12.77	2.45

Figure 45 shows the performance evolution following the five different rankings for the first example. It can be noted from the image and Table 35 that all five rankings identify the same feature as the most important one, which is feature 2. Furthermore, using this feature alone yields an accuracy value close to 0.78, whereas when using all features, accuracy reaches a value close to 0.92. This is in line with the values shown in the table above, indicating feature 2 is far more relevant than the other features. Regarding RQS values, it can be seen that the NoS ranking achieves the highest value, followed by FS, necessity, sufficiency and NaS. When using two features, FS, NoS and sufficiency rankings have a better performance than necessity and NaS rankings, indicating that the [2,7] feature subset leads to a higher performance than the [2,8] subset. Furthermore, it can be noted that all features must be used to reach maximum performance, unlike the previous case study. Thus, no features can be discarded if maximum performance is to be achieved. However, the performance evolution can be fully trusted only until repeated values begin to appear. When this occurs, features with equal values can have interchangeable places in the rankings. For example, in the case of the NaS ranking, after using the top three features, the remaining features have the same value. Thus, they can be ordered according to different ranks.

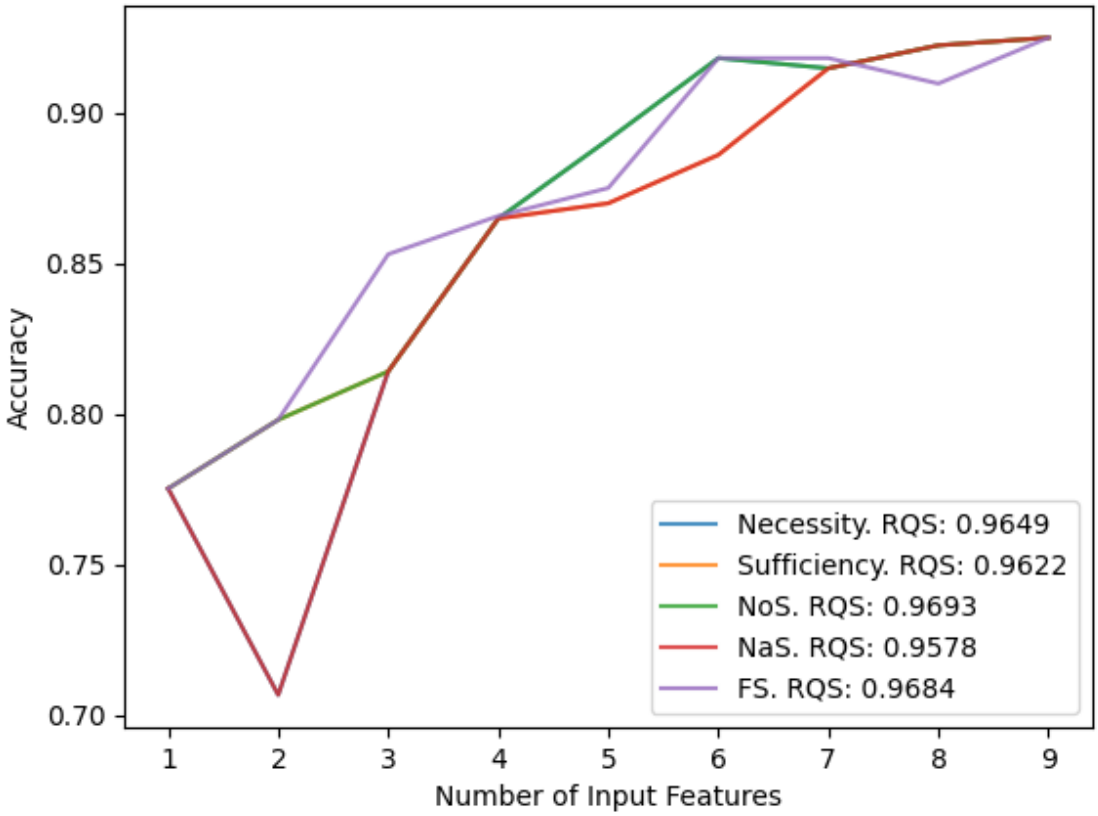

Figure 45 – Accuracy evolution following different feature rankings, with their corresponding RQS value, example #1 for amine treatment plant case.

Figure 46 shows the RBF map for the first example. Results show a strong connection between feature 2 and the output. This is in line with the results shown in the previous image and table, indicating that feature 2 is highly important in terms of necessity or sufficiency. Feature 7 is the next most relevant feature, and the rest of the features seem to be equally important. Furthermore, it can be noted that connections between input features are similar, with exemption of features 2, 7 and 8. To complement this result, Figure 47 shows the minimal explanation sets for this example. According to the image, features 2 and 7 are the most relevant, as they independently explain approximately 90% of the outputs. The next most frequent minimal explanation set is [0, l, 2, 3, 4, 5, 6, 7, 8]. This justifies the similar connections between input features in Figure 46, remembering that a connection between two input features is increased in the figure when the two are part of the minimal explanation set for the corresponding input value. It can be noted that an absolute notion of feature relevance is difficult to obtain from minimal explanation sets, due to the fact that sets may have two or more features. In this case, no feature is more important than the others are; this

information must be obtained from other MES. In Figure 47, it can be noted that features 2 and 7 are more relevant than the rest, as was showed before. However, it is unclear what feature comes next in this relevance order. This also shows that it is important to do an analysis not only viewing features as independent characteristics, but as parts of a whole. In a real system, as with this case study and the previous one, information from one feature can be complementary to another feature. Thus, situations may arise in which a feature by itself does not explain outputs, but in conjunction with other specific features, it may explain a variety of values. In such a case, it is important to analyze causality-based measures such as necessity, but it is also important to analyze whole sets. This is useful also in cases where there are repeated values of causality-based measures. For example, when two features have the same necessity value (different from zero), it may occur that the two constitute a minimal explanation set. Thus, it is not possible to determine which if them is more important. This may also be a reason why the FS layer importance values generate a ranking that is similar to the causality-based rankings but not the same. As shown before in this thesis, the FS layer determines the weights of the different features considering the interactions between them, due to the regularization term. This is different from the causality-based values, where each feature is analyzed independently.

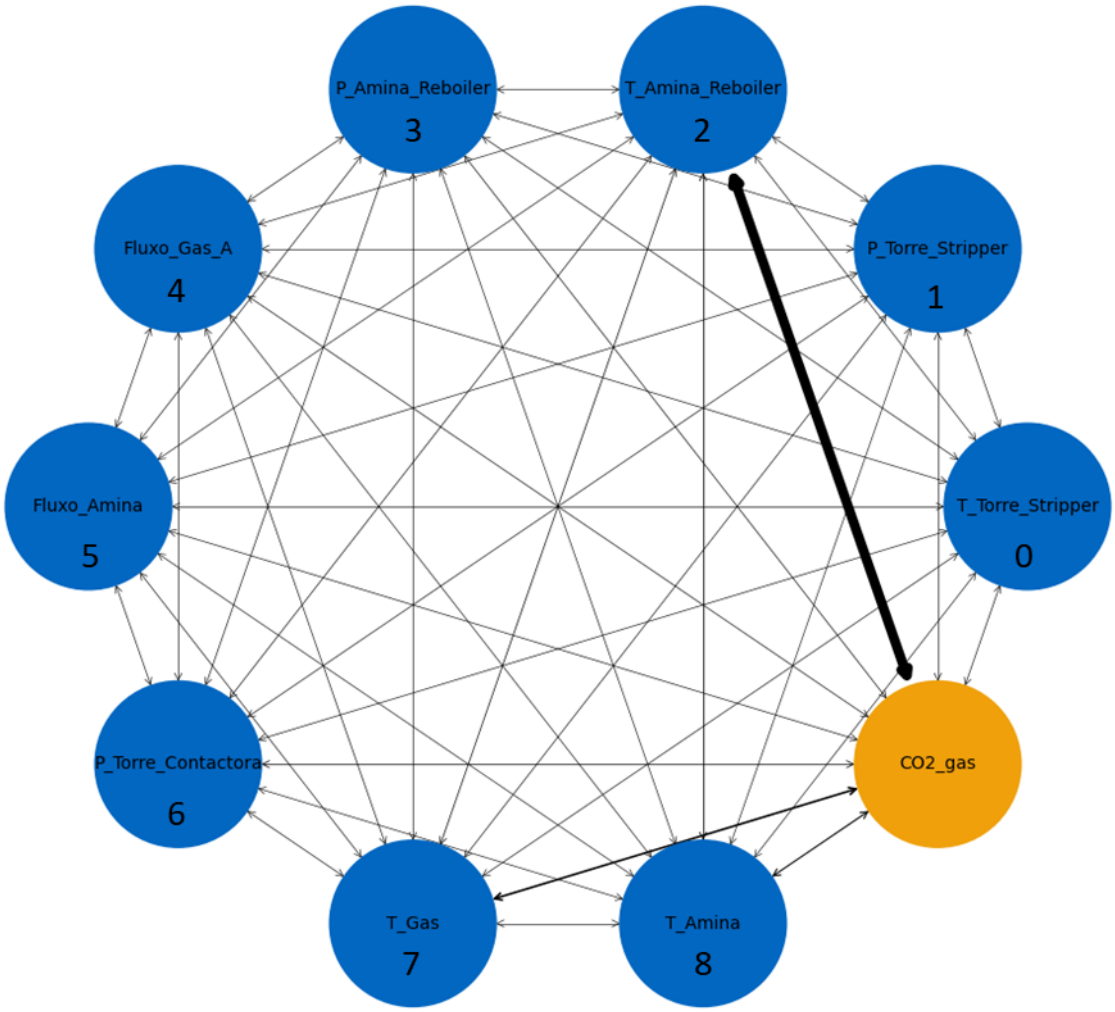

Figure 46 – Relation-between-features (RBF) map, example #1 for amine treatment plant case.

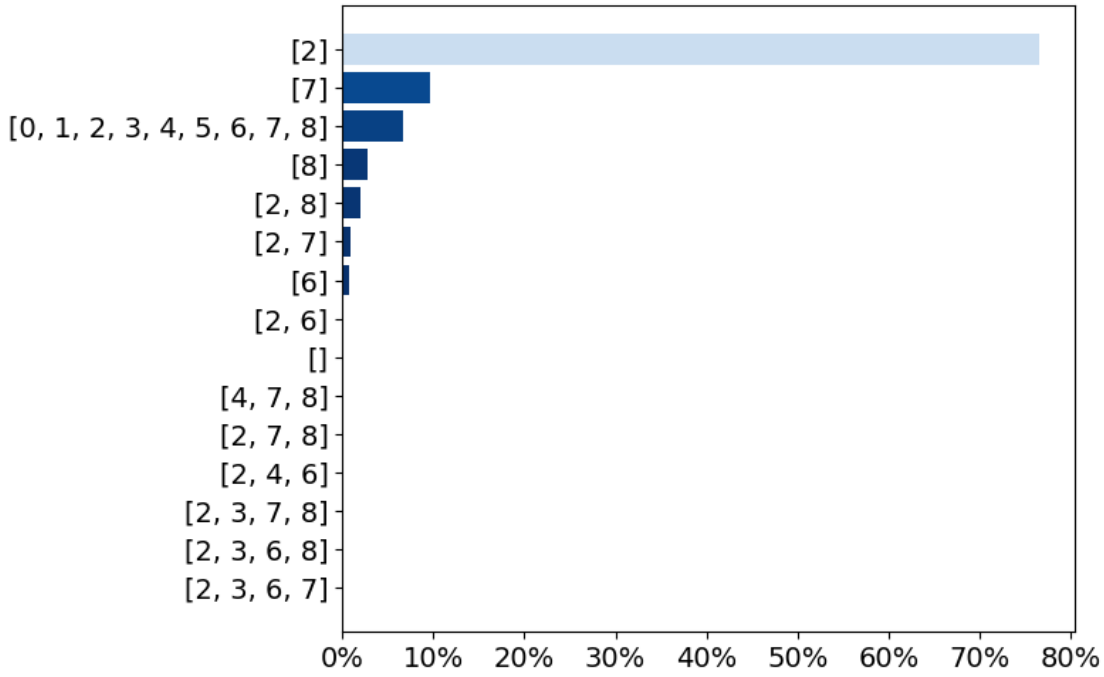

Figure 47 – Top-15 most frequent minimal explanation sets, example #1 for amine treatment plant case.

Like in the previous case study, two examples are shown for comparison. Table 36 shows the causality-based values for the second example. Comparing with Table 35, the most notable difference is with regard to the most relevant feature. Whereas in the previous example feature 2 is the most relevant according to all causality-based values and the FS layer, in this example feature 7 is the most relevant, and feature 2 is the next. In contrast with the previous case study, results between the two examples are very different. One of the big differences between the two case studies is the dataset size. For this case study, 4732 datapoints are used for training and validation, and 1183 for testing. Testing accuracy reaches a mean value of 92.74%. While this is considered a high value, there is room for improvement. One alternative would be to increase the dataset's size. However, this is limited to logistic issues, such as the availability of data. Also, performance improvement is not guaranteed, as data quality is also a requirement.

The link between performance and stability within the two examples shown in this case study (and compared with the previous case study) is that there are signs that the size of the dataset influences the stability of the results and performance values. In the previous case study, the two examples showed similar results, while in this one, results appear to have relevant differences. While in this case study, 92.74% accuracy is reached, in the previous case study 97.92% accuracy is reached.

Regarding necessity, it can be seen that the first four most relevant features are 7, 2, 8, and 6. Although in different order, they are the same values than in the previous example. After that, features 1, 3, 4 and 5 have the same necessity value. Finally, feature 0 presents the lowest value. In the case of sufficiency, the same order is followed for the first four most relevant features. After that, features 3 and 4 have the next best values, and features 0, 1 and 5 have equal values. This is very similar to the necessity ranking in the previous example. Regarding NoS, the order is the following: [7, 2, (6, 8), 3, 4, (1, 5), 0]. Finally, in the case of NaS, the order is the following: [7, 2, 8, 6, (0, 1, 3, 4, 5)]. With regard to the previous example, there are more differences between the two examples than in the previous case study's two examples. However, a tendency is still being followed. It can be noted than in both examples in this case study, features 2, 6, 7 and 8 are more relevant than the rest, despite the order of the rankings being different.

Table 36 – Causality-based values for each feature, example #2 for amine treatment plant case.

Feature	Feature Tag	Necessity %	Sufficiency %	NoS %	NaS %
0	**T_Torre_Stripper**	7.2	0	7.2	0
1	**P_Torre_Stripper**	7.29	0	7.29	0
2	**T_Amina_Reboiler**	17.34	8.19	20.85	4.69
3	**P_Amina_Reboiler**	7.29	0.34	7.63	0
4	**Fluxo_Gas_A**	7.29	0.03	7.32	0
5	**Fluxo_Amina**	7.29	0	7.29	0
6	**P_Torre_Contactora**	11.3	3.14	14.27	0.17
7	**T_gas**	79.86	71.07	87.37	63.56
8	**T_Amina**	13.5	3.84	14.27	3.08

Figure 48 shows the accuracy evolution following each of the different feature rankings. While all of them follow a similar tendency, it can be noted that the FS ranking has a higher RQS. This is mainly influenced by the four first values. It can be noted that the FS ranking is different from the causality-based rankings, especially regarding the top-3 features. While all the causality-based rankings put feature 7 as the most important, the FS ranking chooses another feature.

When looking at the peak performance, it can be noted that eight features are needed. Thus, one feature can be discarded. According to the necessity and NoS rankings, the feature that can be discarded is feature 0, which is the temperature at the stripping tower.

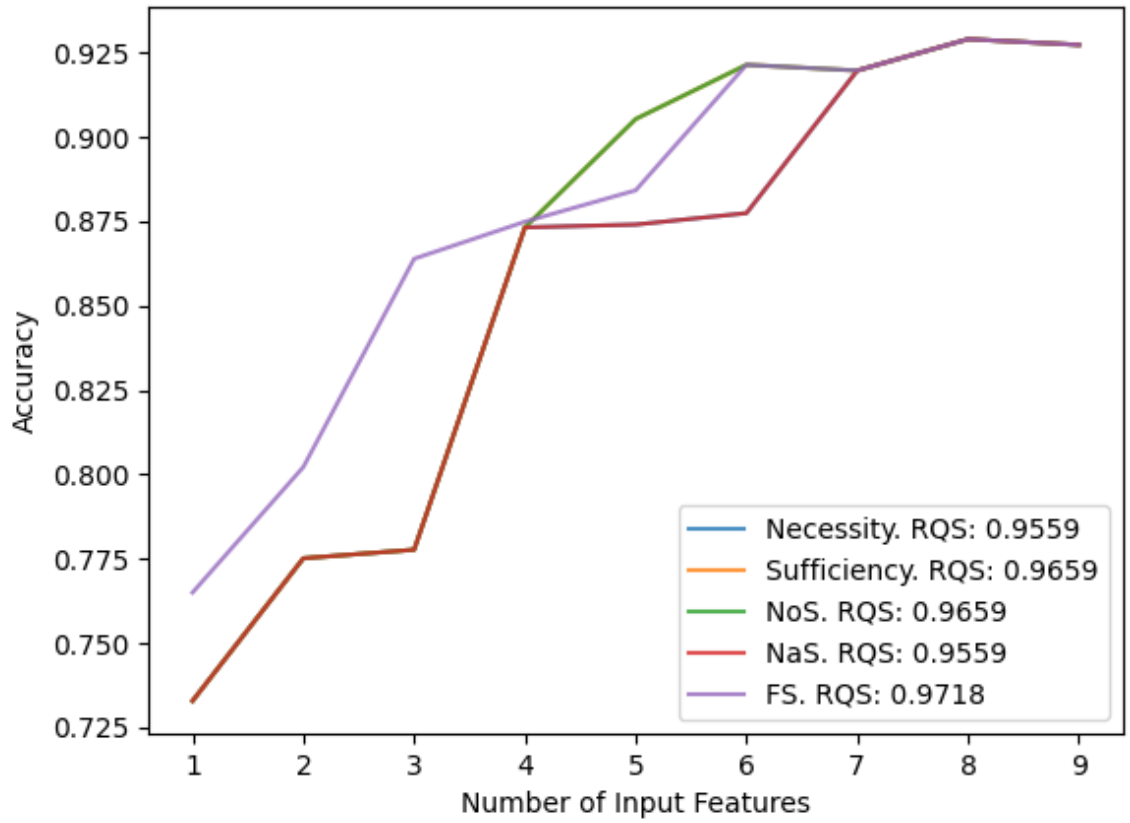

Figure 48 – Accuracy evolution following different feature rankings, with their corresponding RQS value, example #2 for amine treatment plant case.

Figure 49 shows the RBF map for the second example, while Figure 50 shows the top-15 minimal explanation sets. Results show a strong connection between feature 7 and the output. This is confirmed by the fact that approximately 70% of the input values in the test set are explained by this feature. Regarding the rest of the features, it can be seen from the RBF map that there is a similar connection between all the input features, like in the previous example. This is in line with the MES distribution, which indicates that the feature subset [0, 1, 2, 3, 4, 5, 6, 7, 8,] is the third most frequent explanation set.

Through these two examples in this case study, it can be shown the importance of inherently interpretable models, rather than the use of external techniques that generate explanations using approximation methods, such as LIME. While LIME is useful when there is a black-box model already trained which needs to be analyzed, research must be directed towards developing interpretable models with the task performance of black-box models, which is the main objective of this thesis. In the case of SHAP, the main drawback for its use in neural networks is time consumption due to the extensive search of marginal contributions.

In this case study, the two examples reach similar performance values, as seen in Figure 45 and Figure 48. However, their dynamics are different. This is an example of the mentioned Rashomon effect; two different models achieve similar results. While in the first example, approximately 80% of the values are explained by the amine temperature at the reboiler, in the second one approximately 70% of the values are explained by the non-treated gas temperature.

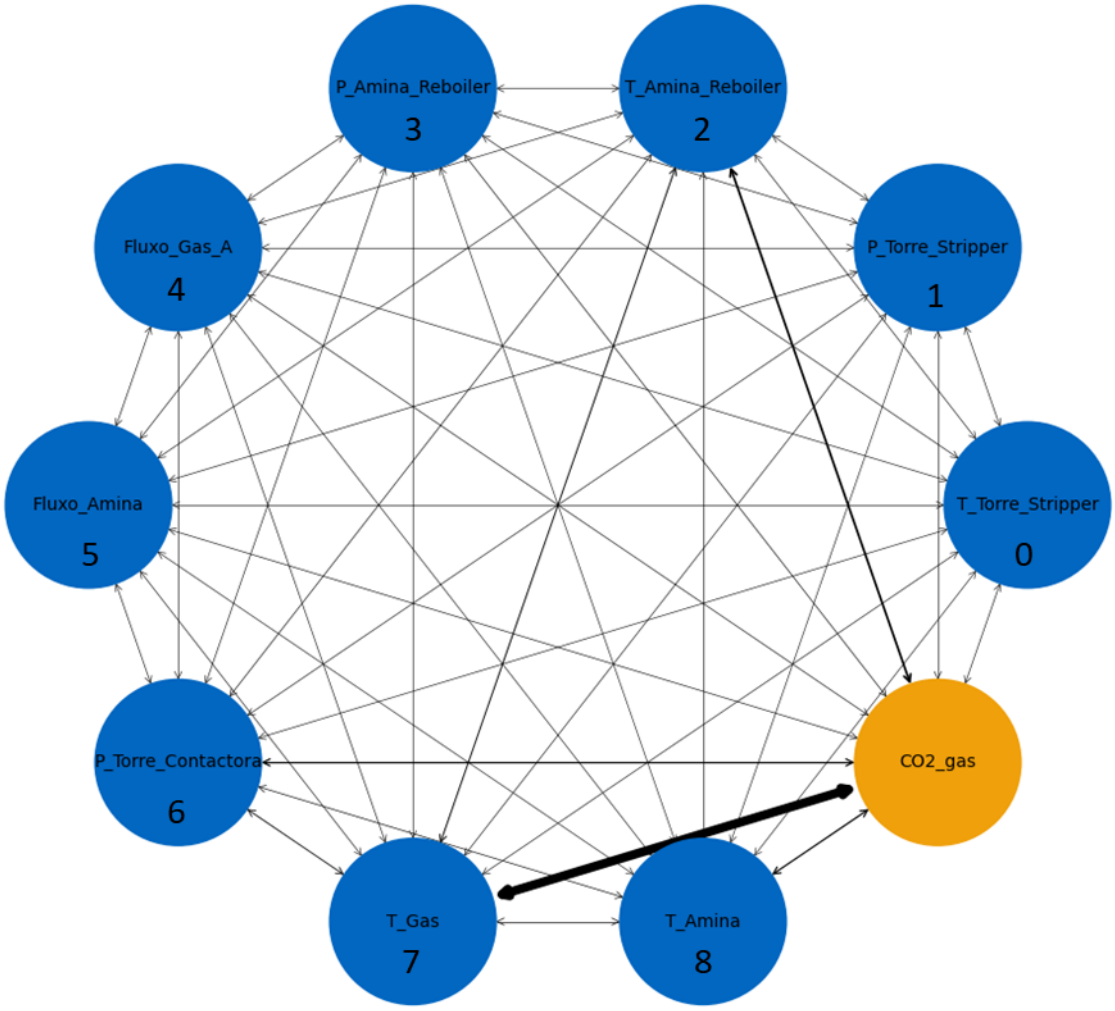

Figure 49 – Relation-between-features (RBF) map, example #2 for amine treatment plant case.

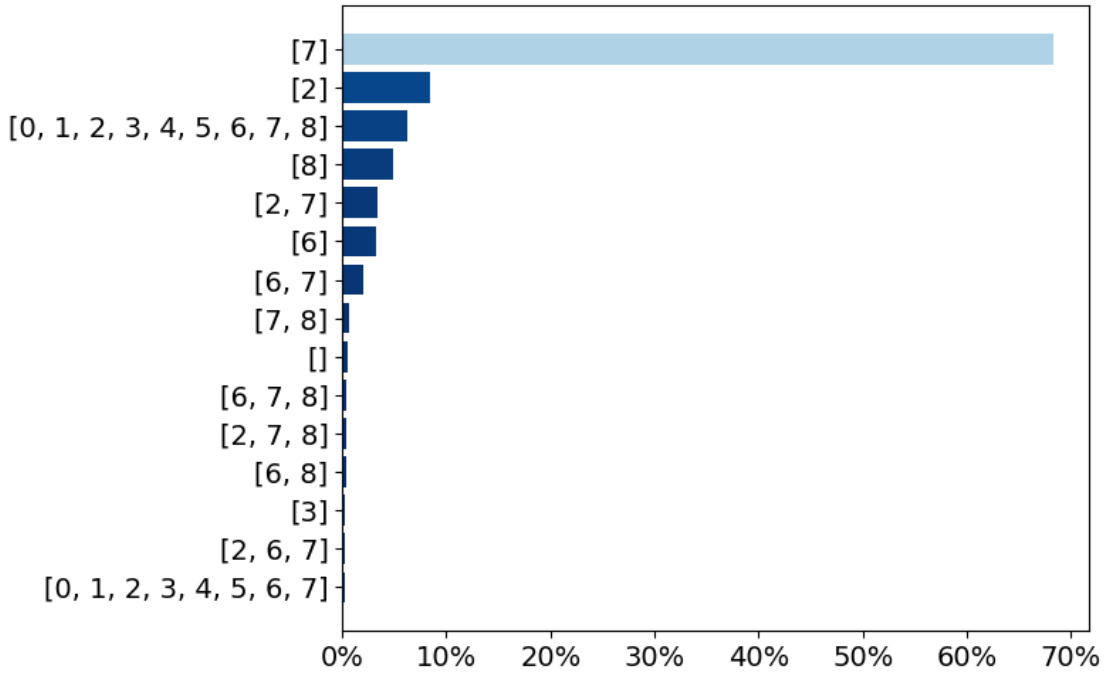

Figure 50 – Top-15 most frequent minimal explanation sets, example #2 for amine treatment plant case.

6. CONCLUSION

The objective of this thesis was to develop frameworks for interpretability of deep neural networks for prognostics and health management in order to address the black-box nature of neural networks. This objective has been met in this thesis through three frameworks. In the first one, a feature selection layer is embedded in a neural network between the input layer and the first hidden layer. This is used to determine the most important features for the model and if there are features that can be discarded. Also, a novel metric, referred to as ranking quality score (RQS), is proposed for comparison of feature rankings. In the second one, a multi-task architecture for simultaneous fault diagnosis and counterfactual generation is proposed. With this, the neural network is able to diagnose health states and give information about the minimal changes that must be made to the input values in order for the health state to change, either from a healthy state condition to a faulty one or vice versa. In the third technique, the two approaches for interpretability are merged into one. Also, a methodology for calculation of causality-based values using counterfactuals is proposed, which is used to generate causality-based feature rankings and to compare them with the feature selection-based rankings.

Regarding the first technique, results across three case studies have shown that the technique achieves higher RQS values than the rest of the compared techniques. It identifies irrelevant features, allowing the model to reach maximum performance with a subset of the input features. Indeed, in the CWR case, maximum performance was reached with 44 out of the 100 input features. Regarding performance, it can be concluded that the inclusion of an FS layer to a deep neural network at least maintains the same level of performance. Indeed, in the NGTP case, performance presents a 19.79% increase in performance. On the other hand, results in the C-MAPSS FD004 case indicate a 0.05% decrease in RUL prediction MSE.

For the second technique, results show that the SCF-Net is able to generate a set of realistic, highly sparse counterfactuals in both binary and multiclass scenarios. The generated counterfactuals are suitable within the corresponding desired class data manifold, and most of them require the modification of up to three features to be generated. Indeed, for the water injection pump and amine treatment plant cases, the mean number of features required to generate valid counterfactuals are 3.28 and 2.37, respectively. This guarantees the generation of counterfactuals that can be understood by the human end user. Furthermore, the proposed

SCF-Net overcomes the accuracy/interpretability tradeoff. Thus, it can be deployed without significant performance loss.

Results regarding the third proposed technique show a model in which feature selection-based rankings are highly correlated to causality-based feature rankings, more specifically necessity, sufficiency and NoS. This is useful to give information about why some features are given higher importance values than other in the feature selection layer. This information was not available when using only the FS layer, in the first proposed technique. Furthermore, by combining two approaches for interpretability, trained models can be analyzed more thoroughly to reveal previously unseen insights. In this work, accuracy evolution plots, relation-between-features maps and minimal explanation sets are the techniques used for this analysis.

This work attempts to improve the interpretability of DL-based PHM models by presenting techniques for interpretation of neural networks applied to PHM in order to increase transparency while keeping high performance level and thus enable more real-world applications in the industry. Through results for different frameworks and case studies, it can be concluded that the initial objective has been met. The three novel techniques show that inherently interpretable neural networks can be developed, trained and deployed in real-world applications achieving high performance levels, which is the main novelty of this thesis.

Future works include evaluating the proposed frameworks in other DL algorithms, such as autoencoders, convolutional neural networks, transformers, and long short-term memory networks. In the case of SCF-Net and FS-SCF network, these two approaches are limited to classification tasks only. Thus, an important line of research would be to adjust them for regression tasks, in order to be able to assess RUL prediction. In the case of the FS layer, the feature importance values can be utilized to aid in data visualization techniques. In (GRISCI; KRAUSE; DORN, 2021), the authors propose a technique for 2D visualization referred to as “Weighted t-SNE”, which is based in feature importance values and t-Distributed Stochastic Neighbor Embedding (t-SNE) (DER MAATEN; HINTON, 2008). The importance values obtained from the FS layer could be used in the weighted t-SNE to analyze if there are improvements in data visualization.

Regarding transformers (VASWANI et al., 2017), they have been used extensively for language recognition tasks. In the context of risk analysis, the authors in (MACÊDO et

al., 2022) use transformers to identify risk features from preliminary hazard analysis documents in an oil refinery. Interpretability techniques such as the ones developed in this thesis could be used in this context to better understand how the model generates its outputs.

REFERENCES

ALOM, M. Z. et al. A state-of-the-art survey on deep learning theory and architectures. **Electronics (Switzerland)**, v. 8, n. 3, 2019.

ALVAREZ-MELIS, D.; JAAKKOLA, T. S. **Towards robust interpretability with self-explaining neural networks**. Advances in Neural Information Processing Systems. **Anais**...Montreal, Canada, December 3-8: 2018

ARIA, A. et al. Estimating damage size and remaining useful life in degraded structures using deep learning-based multi-source data fusion. **Structural Health Monitoring**, v. 19, n. 5, p. 1542–1559, 2020.

ATAMURADOV, V. et al. Prognostics and health management for maintenance practitioners - review, implementation and tools evaluation. **International Journal of Prognostics and Health Management**, v. 8, n. Special Issue 7, 2017.

BAHDANAU, D.; CHO, K. H.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. **3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings**, p. 1–15, 2015.

BARRAZA, J. F. et al. Capsule Neural Networks for structural damage localization and quantification using transmissibility data. **Applied Soft Computing**, v. 97, p. 106732, 2020.

BARRAZA, J. F.; DROGUETT, E. L.; MARTINS, M. R. Towards Interpretable Deep Learning: A Feature Selection Framework for Prognostics and Health Management Using Deep Neural Networks. **Sensors**, v. 21, n. 17, p. 5888, 1 set. 2021.

BARREDO ARRIETA, A. et al. Explainable Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. **Information Fusion**, v. 58, n. December 2019, p. 82–115, 2020.

BASU, S. et al. Learning Sparse Feature Representations using Probabilistic Quadtrees and Deep Belief Nets. **Neural Processing Letters**, v. 45, n. 3, p. 855–867, 11 set. 2015.

BATTITI, R. Using Mutual Information for Selecting Features in Supervised Neural Net Learning. **IEEE Transactions on Neural Networks**, v. 5, n. 4, p. 537–550, 1994.

BAUR, M.; ALBERTELLI, P.; MONNO, M. A review of prognostics and health management of machine tools. **International Journal of Advanced Manufacturing Technology**, v. 107, n. 5–6, p. 2843–2863, 2020.

BAXTER, J. A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling. **Machine Learning**, v. 28, n. 1, p. 7–39, 1997.

BECK, A.; TEBOULLE, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. **SIAM journal on imaging sciences**, v. 2, n. 1, p. 183–202, 2009.

BELLMAN, R. Dynamic programming. **Science**, v. 153, n. 3731, p. 34–37, 1 jul. 1966.

BEN ALI, J. et al. Accurate bearing remaining useful life prediction based on Weibull distribution and artificial neural network. **Mechanical Systems and Signal Processing**, v. 56, p. 150–172, 2015.

BEN BRAHIM, A.; LIMAM, M. A hybrid feature selection method based on instance learning and cooperative subset search. **Pattern Recognition Letters**, v. 69, p. 28–34, 2016.

BENGIO, Y.; LECUN, Y. Scaling Learning Algorithms toward AI. **Large-Scale Kernel Machines**, n. 1, p. 1–41, 2007.

BENTLEY, J. L. Multidimensional binary search trees used for associative searching. **Communications of the ACM**, v. 18, n. 9, p. 509–517, 1 set. 1975.

BERGHOUT, T. et al. A deep supervised learning approach for condition-based maintenance of naval propulsion systems. **Ocean Engineering**, v. 221, n. September 2020, p. 108525, 2021.

BERGSTRA, J. et al. Theano: Deep Learning on GPUs with Python. **Journal of Machine Learning Research**, v. 1, p. 1–48, 2011.

BIGGIO, L.; KASTANIS, I. Prognostics and Health Management of Industrial Assets: Current Progress and Road Ahead. **Frontiers in Artificial Intelligence**, v. 3, n. November, p. 1–24, 2020.

BLUM, A. L.; LANGLEY, P. Artificial Intelligence Selection of relevant features and examples in machine. **Artificial Intelligence**, v. 97, n. 1–2, p. 245–271, 1997.

BORDT, S. et al. Post-Hoc Explanations Fail to Achieve their Purpose in Adversarial Contexts. 2022.

BOTTOU, L. **Large-scale machine learning with stochastic gradient descent**. Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers. **Anais**...Springer Science and Business Media Deutschland GmbH, 2010Disponível em: <https://link.springer.com/chapter/10.1007/978-3-7908-2604-3_16>. Acesso em: 17 abr. 2021

BREIMAN, L. et al. **Classification and regression trees**. [s.l.] CRC press, 1984.

BREIMAN, L. Random forests. **Machine learning**, v. 45, n. 1, p. 5–32, 2001.

BYRNE, R. M. J. Précis of the rational imagination: How people create alternatives to reality. **Behavioral and Brain Sciences**, v. 30, n. 5–6, p. 439–480, 2007.

BYRNE, R. M. J. Counterfactuals in explainable artificial intelligence (XAI): Evidence from human reasoning. **IJCAI International Joint Conference on Artificial Intelligence**, v. 2019- Augus, p. 6276–6282, 2019.

CAI, J. et al. Feature selection in machine learning: A new perspective. **Neurocomputing**, v. 300, p. 70–79, 2018.

CARABIN, G.; WEHRLE, E.; VIDONI, R. Smart Mechanical Systems for Manufacturing in the Era of. **Chiang Mai University Journal of Natural Sciences**, v. 20, n. 734713, p. 1–11, 2020.

CARVALHO, D. V.; PEREIRA, E. M.; CARDOSO, J. S. Machine learning interpretability: A survey on methods and metrics. **Electronics (Switzerland)**, v. 8, n. 8, p. 1–34, 2019.

CHANG, C.-H.; RAMPASEK, L.; GOLDENBERG, A. Dropout Feature Ranking for Deep Learning Models. 2017.

CHEN, X. et al. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. **Advances in Neural Information Processing Systems**, p. 2180–2188, 2016.

CHEN, Z. et al. Deep neural networks-based rolling bearing fault diagnosis. **Microelectronics Reliability**, v. 75, p. 327–333, 2017.

COFRE-MARTEL, S. et al. Deep Convolutional Neural Network-Based Structural Damage Localization and Quantification Using Transmissibility Data. **Shock and Vibration**, v. 2019, p. 1–27, 8 set. 2019.

COFRE-MARTEL, S.; DROGUETT, E. L.; MODARRES, M. **Uncovering the underlying physics of degrading system behavior through a deep neural network framework: The case of rul prognosis**. Disponível em: <https://arxiv.org/abs/2006.09288>. Acesso em: 20 maio. 2021.

DANDL, S. et al. Multi-objective counterfactual explanations. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 12269 LNCS, n. 01, p. 448–469, 2020.

DASGUPTA, A. et al. **Feature selection methods for text classification**. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. **Anais**...2007

DER MAATEN, L.; HINTON, G. Visualizing data using t-SNE. **Journal of machine learning research**, v. 9, n. 11, 2008.

DONG, Y. et al. Improving interpretability of deep neural networks with semantic information. **Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017**, v. 2017- Janua, p. 975–983, 2017.

DONIGER, W.; O'FLAHERTY, W. D. **Karma and rebirth in classical Indian traditions**. [s.l.] Univ of California Press, 1980.

DOSHI-VELEZ, F.; KIM, B. **Towards A Rigorous Science of Interpretable Machine Learning**. Disponível em: <http://arxiv.org/abs/1702.08608>. Acesso em: 9 mar. 2021.

DREOSSI, T. et al. **Systematic Testing of Convolutional Neural Networks for Autonomous Driving**. Disponível em: <http://arxiv.org/abs/1708.03309>. Acesso em: 3 dez. 2020.

DUCHI, J.; SINGER, Y. **Adaptive Subgradient Methods for Online Learning and Stochastic Optimization * Elad HazanJournal of Machine Learning Research**. [s.l:

s.n.]. Disponível em: <https://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf>. Acesso em: 17 abr. 2021.

ESTEVA, A. et al. Dermatologist-level classification of skin cancer with deep neural networks. **Nature**, v. 542, n. 7639, p. 115–118, 2017.

FAN, F.; XIONG, J.; WANG, G. **On Interpretability of Artificial Neural Networks: A Survey**. Disponível em: <https://arxiv.org/abs/2001.02522>. Acesso em: 1 mar. 2021.

FENG, C. et al. A data-driven multi-model methodology with deep feature selection for short-term wind forecasting. **Applied Energy**, v. 190, p. 1245–1257, 2017.

FERREIRA, A. J.; FIGUEIREDO, M. A. T. Efficient feature selection filters for high-dimensional data. **Pattern Recognition Letters**, v. 33, n. 13, p. 1794–1804, 2012.

FIGUEROA BARRAZA, J. et al. Deep learning health state prognostics of physical assets in the Oil and Gas industry. **Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability**, 2020.

FINK, O. et al. Potential, challenges and future directions for deep learning in prognostics and health management applications. **Engineering Applications of Artificial Intelligence**, v. 92, n. January, p. 103678, 2020.

FLORIAN, E.; SGARBOSSA, F.; ZENNARO, I. Machine learning-based predictive maintenance : A cost-oriented model for implementation. **International Journal of Production Economics**, v. 236, n. June 2020, p. 108114, 2021.

FREDERICK, D. K.; DECASTRO, J. A.; LITT, J. S. **User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)**. Washington, DC, USA: National Aeronautics and Space Administration, 2007.

GAN, M.; WANG, C.; ZHU, C. Construction of hierarchical diagnosis network based on deep learning and its application in the fault pattern recognition of rolling element bearings. **Mechanical Systems and Signal Processing**, v. 72–73, p. 92–104, 1 maio 2016.

GENUER, R.; POGGI, J. M.; TULEAU-MALOT, C. Variable selection using random forests. **Pattern Recognition Letters**, v. 31, n. 14, p. 2225–2236, 2010.

GOLDBERG, Y. **Neural Network Methods for Natural Language Processing**. [s.l.]

Morgan and Claypool Publishers, 2017. v. 10

GOODFELLOW, I. et al. **Deep learning**. [s.l.] MIT press Cambridge, 2016. v. 1

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning. **MIT press**, p. 1, 2016.

GOODMAN, N. The Problem of Counterfactual Conditionals. **The Journal of Philosophy**, v. 44, n. 5, p. 113, 1947.

GRATH, R. M. et al. Interpretable Credit Application Predictions With Counterfactual Explanations. p. 1–9, 2018.

GRISCI, B. I.; KRAUSE, M. J.; DORN, M. Relevance aggregation for neural networks interpretability and knowledge discovery on tabular data. **Information Sciences**, v. 559, p. 111–129, 2021.

GUI, N.; GE, D.; HU, Z. **AFS: An attention-based mechanism for supervised feature selection**. AAAI Conference on Artificial Intelligence. **Anais**...Honolulu, Hawaii, USA, January 27 - February 1: 2019a

GUI, N.; GE, D.; HU, Z. **The code of the AAAI-19 paper "AFS: An Attention-based mechanism for Supervised Feature Selection".** Disponível em: <https://github.com/upup123/AAAI-2019-AFS>. Acesso em: 15 fev. 2020b.

GUO, H.; NGUYEN, T. H.; YADAV, A. CounterNet: End-to-End Training of Counterfactual Aware Predictions. n. Ml, p. 1–24, 2021.

GUYON, I.; DE, A. M. **An Introduction to Variable and Feature Selection André ElisseeffJournal of Machine Learning Research**. [s.l: s.n.]. Disponível em: <https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf?ref=driverlayer.com/web> . Acesso em: 11 abr. 2021.

HALFORD, G. S. et al. How many variables can humans process? **Psychological Science**, v. 16, n. 1, p. 70–76, 2005.

HALPERN, J. Y. **Actual causality**. [s.l: s.n.].

HAMEED, S. S. et al. Filter-wrapper combination and embedded feature selection for gene expression data. **International Journal of Advances in Soft Computing and its**

Applications, v. 10, n. 1, p. 90–105, 2018.

HASHEMIAN, H. M.; BEAN, W. C. State-of-the-art predictive maintenance techniques. **IEEE Transactions on Instrumentation and Measurement**, v. 60, n. 10, p. 3480–3492, 2011.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The elements of statistical learning: data mining, inference, and prediction**. [s.l.] Springer Science \& Business Media, 2009.

HE, X.; CAI, D.; NIYOGI, P. Laplacian Score for feature selection. **Advances in Neural Information Processing Systems**, p. 507–514, 2005.

HELLEPUTTE, T.; DUPONT, P. **Partially supervised feature selection with regularized linear models**. 26th International Conference On Machine Learning, ICML 2009. **Anais**...Montreal, Canada, June 14-18: 2009

HOERL, A. E.; KENNARD, R. W. Ridge Regression: Biased Estimation for Nonorthogonal Problems. **Technometrics**, v. 12, n. 1, p. 55–67, 1970a.

HOERL, A. E.; KENNARD, R. W. Ridge Regression: Applications to Nonorthogonal Problems. **Technometrics**, v. 12, n. 1, p. 69–82, 1970b.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, v. 2, n. 5, p. 359–366, 1989.

IVAKHNENKO, A. G. Polynomial Theory of Complex Systems. **IEEE Transactions on Systems, Man and Cybernetics**, v. 1, n. 4, p. 364–378, 1971.

IVAKHNENKO, A.; LAPA, V. G. **Cybernetics and forecasting techniques**. New York: American Elsevier Pub. Co., 1967.

JARDINE, A. K. S.; LIN, D.; BANJEVIC, D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. **Mechanical Systems and Signal Processing**, v. 20, n. 7, p. 1483–1510, 2006.

JOVIĆ, A.; BRKIĆ, K.; BOGUNOVIĆ, N. A review of feature selection methods with applications. **2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015 - Proceedings**, p. 1200–1205, 2015.

KAHNEMAN, D.; MILLER, D. T. Norm Theory: Comparing Reality to Its Alternatives. **Psychological Review**, v. 93, n. 2, p. 136–153, 1986.

KIM, T. S.; SOHN, S. Y. Multitask learning for health condition identification and remaining useful life prediction: deep convolutional neural network approach. **Journal of Intelligent Manufacturing**, v. 32, n. 8, p. 2169–2179, 2021.

KINGMA, D. P.; BA, J. L. Adam: A method for stochastic optimization. **3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings**, p. 1–15, 2015.

KIRA, K.; RENDELL, L. A. **A Practical Approach to Feature Selection**. [s.l.] Morgan Kaufmann Publishers, Inc., 1992.

KLAISE, J.; LOOVEREN, A. VAN; VACANTI, G. Alibi Explain: Algorithms for Explaining Machine Learning Models Alexandru Coca *. **Journal of Machine Learning Research**, v. 22, p. 1–7, 2021.

KOMMIYA MOTHILAL, R. et al. **Towards Unifying Feature Attribution and Counterfactual Explanations: Different Means to the Same End**. [s.l.] Association for Computing Machinery, 2021. v. 1

KONG, Z. et al. Convolution and long short-term memory hybrid deep neural networks for remaining useful life prognostics. **Applied Sciences (Switzerland)**, v. 9, n. 19, 2019.

KONONENKO, I. Estimating attributes: Analysis and extensions of RELIEF. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 784 LNCS, p. 171–182, 1994.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. **ImageNet Classification with Deep Convolutional Neural NetworksAdvances In Neural Information Processing Systems**, 2012. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>. Acesso em: 17 jul. 2018

LAROCHELLE, H. et al. Exploring strategies for training deep neural networks. **Journal of Machine Learning Research**, v. 10, p. 1–40, 2009.

LASI, H. et al. Industry 4.0. **Business and Information Systems Engineering**, v. 6, n. 4, p. 239–242, 2014.

LECUN, Y.; BENGIO, Y.; HINTON, G. **Deep learningNature**Nature Publishing Group, , 27 maio 2015.

LECUN, Y.; CORTES, C.; BURGES, C. J. C. **The MNIST database of handwritten digits, 1998**. Disponível em: <http://yann.lecun.com/exdb/mnist/>. Acesso em: 14 dez. 2020.

LEWIS, D. Causation. **The Journal of Philosophy**, v. 70, n. 17, p. 556–567, 1 maio 1974.

LI, H.; YU, L.; HE, W. The Impact of GDPR on Global Technology Development. **Journal of Global Information Technology Management**, v. 22, n. 1, p. 1–6, 2019.

LI, J. et al. Feature Selection: A Data Perspective. **ACM Computing Surveys**, v. 50, n. 6, 29 jan. 2016.

LI, J. et al. Feature selection: A data perspective. **ACM Computing Surveys**, v. 50, n. 6, 2017.

LI, X.; DING, Q.; SUN, J. Q. Remaining useful life estimation in prognostics using deep convolution neural networks. **Reliability Engineering and System Safety**, v. 172, n. November 2017, p. 1–11, 2018.

LI, Y. et al. Random forest regression for online capacity estimation of lithium-ion batteries. **Applied Energy**, v. 232, n. September, p. 197–210, 2018.

LIN, T. et al. Multi-task Learning Based Classified-assisted Prediction Network for Remaining Useful Life Prediction. **ICSMD 2021 - 2nd International Conference on Sensing, Measurement and Data Analytics in the Era of Artificial Intelligence**, 2021.

LIPTON, Z. C. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. **Queue**, v. 16, n. 3, p. 1–28, 2018.

LIU, J.; JI, S.; YE, J. Multi-task feature learning via efficient ℓ2, 1-norm minimization. **Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, UAI 2009**, p. 339–348, 2009.

LIU, S. et al. Generative counterfactual introspection for explainable deep learning. **GlobalSIP 2019 - 7th IEEE Global Conference on Signal and Information Processing, Proceedings**, 2019.

LOPARO, K. A. **Bearing Data Center, Case Western Reserve University**. Disponível em: <https://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website>. Acesso em: 8 fev. 2021.

LUNDBERG, S. M.; LEE, S. I. **A unified approach to interpreting model predictions**. Advances in Neural Information Processing Systems. **Anais**...Long Beach, CA, USA, December 4-9: 2017

MACÊDO, J. B. et al. Identification of risk features using text mining and BERT-based models: Application to an oil refinery. **Process Safety and Environmental Protection**, v. 158, p. 382–399, 2022.

MAHAJAN, D.; TAN, C.; SHARMA, A. Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers. n. NeurIPS, 2019.

MAKHZANI, A. et al. Adversarial Autoencoders. 2015.

MALDONADO, S.; LÓPEZ, J. Dealing with high-dimensional class-imbalanced datasets: Embedded feature selection for SVM classification. **Applied Soft Computing Journal**, v. 67, p. 94–105, 2018.

MBUVHA, R.; BOULKAIBET, I.; MARWALA, T. Automatic Relevance Determination Bayesian Neural Networks for Credit Card Default Modelling. 2019.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The Bulletin of Mathematical Biophysics**, v. 5, n. 4, p. 115–133, dez. 1943.

MINSKY, M.; PAPERT, S. **Perceptrons - An Introduction to Computational Geometry**. Cambridge, Massachusetts: MIT Press, 1969.

MIRESHGHALLAH, F. et al. Privacy in deep learning: A survey. **arXiv**, p. 1–24, 2020.

MIRZA, M.; OSINDERO, S. Conditional Generative Adversarial Nets. p. 1–7, 2014.

MOBLEY, R. K. **An Introuction to Predictive Meintenance**. [s.l: s.n.]. v. 40

MORGAN, S. L.; WINSHIP, C. **Counterfactuals and causal inference**. [s.l.] Cambridge University Press, 2015.

MOTHILAL, R. K.; SHARMA, A.; TAN, C. Explaining machine learning classifiers

through diverse counterfactual explanations. **FAT* 2020 - Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency**, p. 607–617, 2020.

MURRAY, J. **Nine oil and gas companies leading the use of predictive maintenance**. Disponível em: <https://www.nsenergybusiness.com/features/predictive-maintenance-oil-and-gas/>. Acesso em: 30 mar. 2021.

MUTHUKRISHNAN, R.; ROHINI, R. **LASSO: A feature selection technique in predictive modeling for machine learning**. 2016 IEEE International Conference on Advances in Computer Applications, ICACA 2016. **Anais**...Institute of Electrical and Electronics Engineers Inc., 27 mar. 2017

NAJAFABADI, M. M. et al. Deep learning applications and challenges in big data analytics. **Journal of Big Data**, v. 2, n. 1, p. 1–21, 2015.

NARDONE, D.; CIARAMELLA, A.; STAIANO, A. A sparse-modeling based approach for class specific feature selection. **PeerJ Computer Science**, v. 5, p. 1–25, 2019.

NEMIROVSKY, D. et al. CounteRGAN: Generating Realistic Counterfactuals with Residual Generative Adversarial Nets. 2020.

NEZHAD, M. Z. et al. SAFS: A deep feature selection approach for precision medicine. **Proceedings - 2016 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2016**, p. 501–506, 2017.

NG, A. Y. **Feature selection, L1 vs. L2**. Twenty-first international conference on Machine learning - ICML ’04. **Anais**...Alberta, Canada, July 4-8: 2004Disponível em: <http://portal.acm.org/citation.cfm?doid=1015330.1015435>

PAN, S. J.; YANG, Q. A Survey on Transfer Learning. **IEEE Transactions on Knowledge and Data Engineering**, v. 22, n. 10, p. 1345–1359, 2010.

PARK, P. et al. Fault detection and diagnosis using combined autoencoder and long short-term memory network. **Sensors (Switzerland)**, v. 19, n. 21, p. 1–17, 2019.

PARK, Y. C. et al. Performance comparison of aqueous MEA and AMP solutions for biogas upgrading. **Korean Journal of Chemical Engineering**, v. 34, n. 3, p. 921–927, 2017.

PEARL, J. Probabilities Of Causation: Three Counterfactual Interpretations And Their Identification. **Synthese**, v. 121, n. 1, p. 93–149, 1999.

PEARL, J. Causal and counterfactual inference. **The Handbook of Rationality**, p. 1–41, 2018.

PEARL, J. The seven tools of causal inference, with reflections on machine learning. **Communications of the ACM**, v. 62, n. 3, p. 54–60, 2019.

PEARL, J.; MACKENZIE, D. **The Book of Why**. [s.l.] Basic Books, 2018.

REDDY, K. K. et al. **Anomaly detection and fault disambiguation in large flight data: A multi-modal deep auto-encoder approach**. Annual Conference of the Prognostics and Health Management Society. **Anais**...Denver, Colorado, USA, October 3-6: 2016

REZAEIANJOUYBARI, B.; SHANG, Y. Deep learning for prognostics and health management: State of the art, challenges, and opportunities. **Measurement: Journal of the International Measurement Confederation**, v. 163, p. 107929, 2020.

RIBEIRO, M. T.; GUESTRIN, C. **" Why Should I Trust You ?" Explaining the Predictions of Any Classifier**. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. **Anais**...San Francisco, California, USA, August 13-17: 2016

ROBNIK-ŠIKONJA, M.; KONONENKO, I. An adaptation of Relief for attribute estimation in regression. **Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)**, v. 5, p. 296–304, 1997.

ROBNIK, M.; KONENKO, I. Theoretical and empirical analysis of ReliefF and RReliefF. **Machine Learning**, v. 53, n. 1--2, p. 23–69, 2003.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65, n. 6, p. 386–408, nov. 1958.

ROSS, B. C. Mutual information between discrete and continuous data sets. **PLoS ONE**, v. 9, n. 2, 2014.

ROTH, A. **The Shapley value: essays in honor of Lloyd S. Shapley**. [s.l: s.n.].

ROY, D.; MURTY, K. S. R.; MOHAN, C. K. Feature selection using Deep Neural

Networks. **Proceedings of the International Joint Conference on Neural Networks**, v. 2015- Septe, 2015.

RUIZ-TAGLE PALAZUELOS, A.; DROGUETT, E. L.; PASCUAL, R. A novel deep capsule neural network for remaining useful life estimation. **Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability**, v. 234, n. 1, p. 151–167, 2020.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, 1986.

RUSSELL, C. Efficient search for diverse coherent explanations. **FAT* 2019 - Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency**, p. 20–28, 29 jan. 2019.

SAHA, S. K.; SARKAR, S.; MITRA, P. Feature selection techniques for maximum entropy based biomedical named entity recognition. **Journal of Biomedical Informatics**, v. 42, n. 5, p. 905–911, 2009.

SAN MARTIN, G. et al. Deep variational auto-encoders: A promising tool for dimensionality reduction and ball bearing elements fault diagnosis. **Structural Health Monitoring**, v. 18, n. 4, p. 1092–1128, 2019.

SAUER, A.; GEIGER, A. Counterfactual Generative Networks. v. 1, 2021.

SAXENA, A. et al. **Damage propagation modeling for aircraft engine run-to-failure simulation**. 2008 International Conference on Prognostics and Health Management, PHM 2008. **Anais**...2008

SCHOLKOPF, B. et al. Toward Causal Representation Learning. **Proceedings of the IEEE**, p. 1–24, 2021.

SCHUT, L. et al. Generating Interpretable Counterfactual Explanations By Implicit Minimisation of Epistemic and Aleatoric Uncertainties. v. 130, 2021.

SEAWRIGHT, J. Testing for Necessary and/or Sufficient Causation: Which Cases Are Relevant? **Political Analysis**, v. 10, n. 2, p. 178–193, 2002.

SELCUK, S. Predictive maintenance, its implementation and latest trends. **Proceedings of**

the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, v. 231, n. 9, p. 1670–1679, 2017.

SHANNON, C. E. A mathematical theory of communication. **The Bell system technical journal**, v. 27, n. 3, p. 379–423, 1948.

ŠKRLJ, B. et al. Feature importance estimation with self-attention networks. **Frontiers in Artificial Intelligence and Applications**, v. 325, p. 1491–1498, 2020.

SLACK, D.; HILGARD, S.; JIA, E. **Fooling LIME and SHAP : Adversarial Attacks on Post hoc Explanation Methods**. AAAI/ACM Conference on AI, Ethics, and Society. **Anais**...New York, USA, February 7-8: 2020

SRIVASTAVA, N. et al. **Dropout: A Simple Way to Prevent Neural Networks from OverfittingJournal of Machine Learning Research**. [s.l: s.n.]. Disponível em: <http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf>. Acesso em: 10 ago. 2018.

SUNDARARAJAN, M.; TALY, A.; YAN, Q. Axiomatic Attribution for Deep Networks. **34th International Conference on Machine Learning, ICML 2017**, v. 7, p. 5109–5118, 2017.

SUSTO, G. A. et al. Machine learning for predictive maintenance: A multiple classifier approach. **IEEE Transactions on Industrial Informatics**, v. 11, n. 3, p. 812–820, 2015.

TIBSHIRANI, R. Regression Shrinkage and Selection Via the Lasso. **Journal of the Royal Statistical Society: Series B (Methodological)**, v. 58, n. 1, p. 267–288, 1996.

TIELEMAN, T.; HINTON, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. **COURSERA: Neural networks for machine learning**, v. 4, n. 2, p. 26–31, 2012.

VAN HORENBEEK, A.; PINTELON, L. A dynamic predictive maintenance policy for complex multi-component systems. **Reliability Engineering and System Safety**, v. 120, p. 39–50, 2013.

VAN LOOVEREN, A.; KLAISE, J. Interpretable Counterfactual Explanations Guided by Prototypes. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 12976 LNAI, p. 650–

665, 2021.

VASWANI, A. et al. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017.

VENKATESH, B.; ANURADHA, J. A review of Feature Selection and its methods. **Cybernetics and Information Technologies**, v. 19, n. 1, p. 3–26, 2019.

VERMA, S.; DICKERSON, J.; HINES, K. Counterfactual Explanations for Machine Learning: A Review. 2020.

VERSTRAETE, D. et al. Deep learning enabled fault diagnosis using time-frequency image analysis of rolling element bearings. **Shock and Vibration**, v. 2017, 2017.

VERSTRAETE, D.; DROGUETT, E.; MODARRES, M. A deep adversarial approach based on multisensor fusion for remaining useful life prognostics. **Proceedings of the 29th European Safety and Reliability Conference, ESREL 2019**, p. 1072–1077, 2020.

WACHTER, S.; MITTELSTADT, B.; RUSSELL, C. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. **Harv. JL \& Tech.**, v. 31, p. 841, 2017.

WANG, H. A survey of maintenance policies of deteriorating systems. **European Journal of Operational Research**, v. 139, n. 3, p. 469–489, 2002.

WERBOS, P.; JOHN, P. **Beyond regression : new tools for prediction and analysis in the behavioral sciences**. [s.l.] Harvard University, 1 jan. 1974.

WINTER, J. C. F. DE; GOSLING, S. D.; POTTER, J. Comparing the Pearson and Spearman Correlation Coefficients Across Distributions and Sample Sizes: A Tutorial Using Simulations and Empirical Data. **Psychological Methods**, v. 21, n. 3, p. 273–290, 2016.

WOODWARD, J. **Making things happen: A theory of causal explanation**. [s.l.] Oxford university press, 2005.

XIAO, C. et al. Using Spearman's correlation coefficients for exploratory data analysis on big dataset. **Concurrency and Computation: Practice and Experience**, v. 28, n. 14, p. 3866–3878, 2016.

YANG, F. et al. Generative Counterfactuals for Neural Networks via Attribute-Informed Perturbation. **ACM SIGKDD Explorations Newsletter**, v. 23, n. 1, p. 59–68, 2021.

YARMOLUK, D.; TRUEMPI, C. **Why move from condition monitoring to predictive maintenance? - Part 1**. Disponível em: <https://www.ibm.com/blogs/internet-of-things/iot-condition-monitoring-part-one/>. Acesso em: 16 abr. 2021.

YU, Y. et al. A novel deep learning-based method for damage identification of smart building structures. **Structural Health Monitoring**, v. 18, n. 1, p. 143–163, 2019.

YUAN, M.; WU, Y.; LIN, L. Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network. **AUS 2016 - 2016 IEEE/CSAA International Conference on Aircraft Utility Systems**, p. 135–140, 2016.

ZHANG, J. et al. Long short-term memory for machine remaining life prediction. **Journal of Manufacturing Systems**, v. 48, n. June, p. 78–86, 2018.

ZHOU, L. et al. Machine learning on big data: Opportunities and challenges. **Neurocomputing**, v. 237, n. September 2016, p. 350–361, 2017.

ZOU, H.; HASTIE, T. Regularization and variable selection via the elastic net. **Journal of the Royal Statistical Society. Series B: Statistical Methodology**, v. 67, n. 2, p. 301–320, 2005.

ZOU, Q. et al. Remote Sensing Scene Classification. **IEEE Transactions on Geoscience and Remote Sensing Letters**, v. 12, n. 11, p. 2321–2325, 2015.

Appendix A: RQS ≤ 1

To prove that

$$\frac{\sum_{n=1}^{N} PM_n \cdot n}{\sum_{n=1}^{N} \max(PM) \cdot n} \leq 1 \iff \sum_{n=1}^{N} PM_n \cdot n \leq \sum_{n=1}^{N} \max(PM) \cdot n$$

, mathematical induction is used. We generalize PM_n to any function $\alpha(n)$ with values within the range [0,1].

Base case ($n = 1$):

$$\sum_{n=1}^{1} \alpha(n) \cdot n = \alpha(1) \cdot 1 \leq \max_{n\in[1,1]}\{\alpha(n)\} \cdot 1 = \sum_{n=1}^{1} \max_{n\in[1,1]}\{\alpha(n)\} \cdot n$$

Induction step:

$$\sum_{n=1}^{K} \alpha(n) \cdot n \leq \sum_{n=1}^{K} \max_{n\in[1,K]}\{\alpha(n)\} \cdot n \Rightarrow \sum_{n=1}^{K+1} \alpha(n) \cdot n \leq \sum_{n=1}^{K+1} \max_{n\in[1,K+1]}\{\alpha(n)\} \cdot n$$

Proof:

$$\sum_{n=1}^{K+1} \alpha(n) \cdot n = \sum_{n=1}^{K} \alpha(n) \cdot n + \alpha(K+1) \cdot (K+1)$$

$$\leq \sum_{n=1}^{K} \max_{n\in[1,K]}\{\alpha(n)\} \cdot n + \alpha(K+1) \cdot (K+1)$$

$$= \max_{n\in[1,K]}\{\alpha(n)\} \cdot \frac{K(K+1)}{2} + \alpha(K+1) \cdot (K+1)$$

$$\leq \max_{n\in[1,K+1]}\{\alpha(n)\} \cdot \frac{K(K+1)}{2} + \max_{n\in[1,K+1]}\{\alpha(n)\} \cdot (K+1)$$

$$= \max_{n\in[1,K+1]}\{\alpha(n)\}\left(\frac{K(K+1)}{2} + (K+1)\right)$$

$$= \max_{n\in[1,K+1]}\{\alpha(n)\}\left(\frac{(K+1)(K+2)}{2}\right)$$

$$= \sum_{n=1}^{K+1} \max_{n\in[1,K+1]}\{\alpha(n)\}\cdot n$$

■