UNIVERSIDAD DE BURGOS

DOCTORAL THESIS

Machine Learning to Study and Predict Malfunctioning in Robot Software

Nuño Basurto Hornillos

Doctoral Program «Industrial Technologies and Civil Engineering»

Supervisors

Álvaro Herrero Cosío

Carlos Cambra Baseca

September, 2021

GICAP

La memoria titulada “Machine Learning to Study and Predict Malfunctioning in Robot Software” que presenta D. Nuño Basurto Hornillos para optar al Grado de Doctor por la Universidad de Burgos en el programa de doctorado en Tecnologías Industriales e Ingeniería Civil, ha sido realizada bajo la dirección del Dr. Álvaro Herrero Cosío y del Dr. Carlos Cambra Baseca, del Departamento de Ingeniería Informática de la Universidad de Burgos.

Vº. Bº. del Director:	Vº. Bº. del Codirector:	Vº. Bº. del doctorando:
Firmado digitalmente por HERRERO COSIO ALVARO - 71268510C Fecha: 2021.09.30 23:40:05 +02'00'	CAMBRA BASECA CARLOS - 18053084Q Firmado digitalmente por CAMBRA BASECA CARLOS - 18053084Q Fecha: 2021.09.30 23:36:58 +02'00'	Firmado digitalmente por BASURTO HORNILLOS NUÑO - 71295798F Fecha: 2021.09.30 23:33:31 +02'00'
Dr. Álvaro Herrero Cosío	Dr. Carlos Cambra Baseca	D. Nuño Basurto Hornillos

En Burgos, a 30 de septiembre de 2021

A todos aquellos que siempre habéis creido en mi, que nunca dudasteis que lo conseguiría.

Agradecimientos

En primer lugar, me gustaría agradecer a la Universidad de Burgos por concederme la posibilidad de llevar a cabo mi Tesis Doctoral.

La elaboración de una tesis doctoral es un largo camino en el cual te vas encontrando varios obstáculos. En mi caso particular, estos vienen dados por problemas en el código o por la dificultad de publicar ciertos artículos en revistas y sus largos tiempos de revisión. Ante dichos obstáculos uno va encontrando el apoyo de varias personas, la primera de ellas digna de una mención especial es Álvaro Herrero, quien me apoyó y tutorizó durante la realización de la tesis. Junto a él otros miembros del Grupo de Investigación GICAP como mi otro tutor Carlos Cambra y Ángel Arroyo me han ayudado en este proceso.

En mi tiempo de estancia en el extranjero en primer lugar en la ciudad de Breslavia (Polonia), recibí la ayuda de un grandísimo investigador como es Michał Woźniak, quien me abrió los ojos a nuevas formas de desarrollar algoritmos desde un punto de vista diferente. En segundo lugar agradecer a Alfredo Jiménez por recibirme en Burdeos (Francia), con el cual he podido estudiar otro tipo de conjutos de datos, obteniendo así una mayor versatilidad a la hora de afrontar el análisis de datos.

Fuera del mundo académico es tambien necesario reconocer la importancia del apoyo de la gente, siendo necesario mencionar a mi entorno familiar y a mis amigos. Por todos es sabido la dificultad de compaginar la vida social y las intensas horas de investigación, pero son esas personas las que cobran una especial importancia ayudando a evadirte de todo y así poder mirar hacia delante. Entre esta gente aquí comentada, hacer especial hincapié en Andrea, una persona que me apoyó desde el principio a embarcarme en la tesis doctoral.

Acknowledgements

First of all, I would like to thank the University of Burgos for giving me the opportunity to carry out my doctoral thesis.

The elaboration of a doctoral thesis is a long way in which you find several obstacles. In my particular case, there were problems with the code or the difficulty of getting certain articles published in journals. Faced with these obstacles one finds the support of several people, the first of them worthy of special mention is Álvaro Herrero, who supported and supervised me during the development of the thesis. Together with him, other members of the GICAP Research Group such as, my other supervisor Carlos Cambra and Ángel Arroyo have helped me in this process.

During my time abroad, firstly in the city of Wrocław (Poland), I received the help of a very great researcher, Michał Woźniak, who opened my eyes to new ways of developing algorithms from a different point of view. Secondly, I would like to thank Alfredo Jiménez for welcoming me in Bordeaux (France), with whom I have been able to approach other type of data sets, thus obtaining a greater versatility when dealing with data analysis.

Outside the academic world it is also important to acknowledge the importance of people's support, so it is worth mentioning the importance of my family and friends. Everyone knows how difficult it is to combine social life and intense hours of research, that is when these people become especially important, helping you to get away from everything and thus be able to look ahead. Among these people mentioned here, I would like to make special emphasis on Andrea, a person who supported me from the beginning to engage on my doctoral thesis.

Resumen

La industria 4.0 es un paradigma que despierta un interés creciente, siendo uno de sus objetivos la automatización de los sistemas de producción, lo que implica la incorporación de sistemas robóticos modernos. El mantenimiento de estos sistemas es clave para la productividad de las empresas, teniendo que minimizar el tiempo de inactividad. El desarrollo de una herramienta capaz de predecir cuándo van a ocurrir los fallos es clave para el devenir de esta industria.

En la presente tesis doctoral se han elaborado varias estrategias con el objetivo de poder llevar a cabo una satisfactoria monitorización y detección de anomalías que afectan al software de un sistema robótico, empleando para ello técnicas de aprendizaje automático.

En primer lugar, se han aplicado novedosas técnicas exploratorias para analizar los conjuntos de datos bajo estudio, relacionados con el rendimiento de distintos componentes software de un robot.

Uno de los mayores retos a acometer con esta tipología de datos es el desbalanceo existente, dado que las anomalías ocurren en una minoría de ocasiones en comparación con el funcionamiento de un robot en un estado normal. Para mejorar el rendimiento de los diferentes clasificadores se han aplicado varios algoritmos de balanceo de datos.

Adicionalmente, para lograr una mejor optimización de los algoritmos se han aplicado técnicas de imputación bajos diferentes perspectivas. La primera perspectiva de una índole más tradicional aborda la problemática bajo el uso de regresiones a cada uno de los atributos de manera individualizada. Por otro lado, se propone un sistema híbrido de imputación el cual mezclaba el uso de técnicas de clustering con técnicas de regresión, llevando a cabo las regresiones por cada uno de los clusters obtenidosde forma independiente.

Con la combinación de todas estas técnias, se han obtenido interesantes resultados, permitiendo la satisfactoria monitorización del rendimiento de sistemas robóticos, así como la detección automática de anomalías que afectan a este.

Abstract

Industry 4.0 is a paradigm causing an increasing interest, being one of its objectives the automation of production systems, leading to the incorporation of modern robotic systems. The maintenance of these, is a keystone for the productivity of companies, minimizing the downtime. The development of a tool capable of predicting when failures will happen is essential for the future of this industry.

In this doctoral thesis, several strategies have been developed with the aim of being able to carry out a satisfactory monitoring and detection of failures affecting the software of a robotic system, using Machine Learning techniques.

Firstly, novel exploratory techniques have been applied in order to analyze the studied data about the performance of several software components of a robot.

One of the major challenges to be tackled regarding this type of data is the existing imbalance, as anomalies take place in a minority of times compared to the operation of a robot in a normal state. To improve the performance of some classifiers, several balancing algorithms have been applied.

Additionally, in order to achieve a better optimization of the algorithms, imputation techniques have been applied under different perspectives. The first one is more traditional, addresses the problem under the use of regressions for each one of the attributes individually. On the other hand, a hybrid imputation system is proposed that mixes the use of clustering techniques with regression techniques, carrying out the regressions for each of the clusters obtained independently.

Thanks to the combination of all these techniques, interesting results have been obtained, enabling a success full monitoring of robot performance, as well as the automatic detection of anomalies affecting it.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The number of autonomous mobile robots has been steadily increasing in recent times [1], several sectors are affected by this, such as logistics, cleaning or medicine.

Robotic systems require significant maintenance aimed at minimizing downtime, thus increasing productivity. Performance improvements in both hardware and software are much needed. there are a number of novel research papers dealing with these problems in robotics, where anomaly detection in robotic systems is addressed [1] [2] [3].

The research carried out in this thesis uses a dataset of a component-based autonomous robot, in which several anomalies have been induced by software. These anomalies do not prevent the robot from carrying out the aim task, but penalize the counters of the system, more in detail in section 1.4. We have also used a novel methodology validated in this field of study, as detailed in section 1.3

1.1 Previous Work

A complete Machine Learning framework is proposed throughout the different papers of this thesis, comprising different three-dimensional visualization , data imputation techniques or analysis of the validity of applying different data balancing methods.

Anomaly detection has been used in a wide variety of fields, not only in those related to robotics [4] [5] [6]. At the end of the last century, the first research on the use of machine learning techniques in robots was carried out [7]. Subsequently, their application to a variety of types of learning has been proposed: supervised [8], unsupervised [9] and reinforcement [10]. Several problems related to robotics have been discussed, such as communications [11] and control [12] [13] problems. The approach taken so far was mainly concerned with the problems caused on the hardware [8]. Whereas the problems related to the detection of anomalies in the software have hardly been given any attention. Dealing with these problems is a challenge given the lack of available open data sets, even though it is a problem of relative importance. In addition, the presence of Missing Values is a constant in the data collected.

When anomaly detection is conducted in a supervised learning context, the presence of high levels of unbalance is common, since they appear in a smaller number of occasions than the normal state of the data. This leads to the use of balancing techniques is absolutely necessary for good classification ratios [14] [15]. A variety of data balancing techniques have been explored in this thesis.

[1] International Federation of Robotics. Mobile Robots Revolutionize Industry (August 2021). URL:https://ifr.org/ifr-press-releases/news/mobile-robots-revolutionize-industry

The balancing techniques can be divided into three types, oversampling, undersampling and hybrids. Oversampling, deals with the generation of new instances of the minority class, in order to obtain a more balanced data set, it is important to highlight the presence of Sytethic Minority Over-sampling Techniques (SMOTE) [16] [17] and its varieties [18]. On the other hand, the use of undersampling techniques tries to reduce the instances of the majority class in order to obtain this balancing,it has an impact on the loss of information. Finally, hybrid techniques use both approaches at the same time, eliminating instances of the majority class and generating new ones in the minority class.

According to the classification of missing data proposed in [19], in this research the problem is missing completely at random, because the probability that a MV is located in one or more attributes of any instance does not depend on particular circumstances (past values or other missing values). The imputation carried out is a Single Imputation, where the method fills one value for each MV [20]. To optimize the imputed values, it is necessary to do a regression work. In the present research, the applied techniques are: multiple-regression (linear and nonlinear) [21], regression trees [22] and Artificial Neural Networks (ANN), more precisely the Radial Basis Function Network (RBFN) [23] and The Multilayer Perceptron (MLP).

For the imputation of MV, many different methods have been previously proposed, including those based on AI [24]. Nevertheless, little effort has been devoted so far to research on the goodness of ML methods when facing such problem in robotics contexts. One of these scant research works is [25], whose author proposed a probabilistic approach using incomplete data for classification (failure detection). Data samples were classified by calculating, from the data samples that are not missing, the a-priori probability of MV. This proposal was applied to datasets containing only anomalies affecting the hardware and that are outdated (coming from 1999). Advancing the previous proposals, the present work is the first approach to impute MV in a dataset from a component-based robot in order to improve subsequent classification by using data balancing techniques. In order to validate such pioneer proposal, a complete benchmark involving many ML methods has been performed.

In the first approach to this dataset [26] it was shown how data selection could improve the performance of the classifiers, in particular this data selection was done by considering the amount of missing values found in the data. This process has led to the study of other approaches that have been presented throughout this thesis, such as the use of a variety of imputation techniques.

The authors Wienke et al. [27], propose a novel publicly available dataset [28] based on software-induced anomaly detection on a component-based robot. Showing the data collected by the different component indicators. The approach used in the research [29], which is used on the data set exposed above, is the use of supervised learning techniques to obtain a performance improvement in anomaly detection on it. For this purpose they apply a variety of balancing techniques as well as the use of two different classifiers.

Subsequently, they present a doctoral dissertation in which the research is carried out on this data set, showing a set of tools for the systematization of resource control. In addition, the study the detection of anomalies in real time

for the automation of reactions, through the use of Machine Learning techniques, is performed.

1.2 Objectives

In this doctoral thesis have been developing and validating the application of different machine learning techniques in the field of robotics.

The first and most necessary objective deals with the location of a dataset to work with, which must be novel and well-documented.

The review of the state of the art is another objective to be addressed, in order to understand what are the approaches used in this field and then and the most important machine learning techniques used. In this way, it makes possible to develop a work with a valid basis, as well as the application of novel techniques.

To carry out the application of a large number of techniques, as well as to validate them. It is important to later make a fair comparison of these and to realize the differences in behavior at each moment.

1.3 Methodology

During the present research, the application of a novel methodology to improve the cost-efficiently monitoring for industrial 4.0 has been applied [30].

This methodology denominated Analyze, Sense, Preprocess, Predict, Implement and Deploy (ASPPID) 1.1, it covers several stages, such as the acquisition of detection equipment or the evaluation of the captured data. The research validates the use on an industrial dataset, where faults are detected in an alignment process resulting in a highly unbalanced dataset, similar to the one presented in this thesis.

1.3.1 Analyze

The methodology faces the problem of the lack of knowledge to exploit the data, so an analysis of the objective problem must be carried out before attempting to collect the data, in this way it will be possible to focus the collection of data to achieve the desired result. The use of the contract is proposed, where the problem for which an improvement is to be made, the costs of the project, the objective and the time required among others are formally defined. Finally, in this phase the different participants make a brainstorming giving different points of view of how to achieve the objective, in this way the ideas are compiled and a schedule is established with which to work.

For this first phase, the first thing was the search for a data set that would suit our needs, it had to be novel and focused on a large data imbalance. After several searches, the one discussed in this thesis was found [27] and analyzed in a first publication [26].

In the first publication collected in this thesis I, using unsupervised learning, it was possible to observe a general visualization of the problem, seeing its

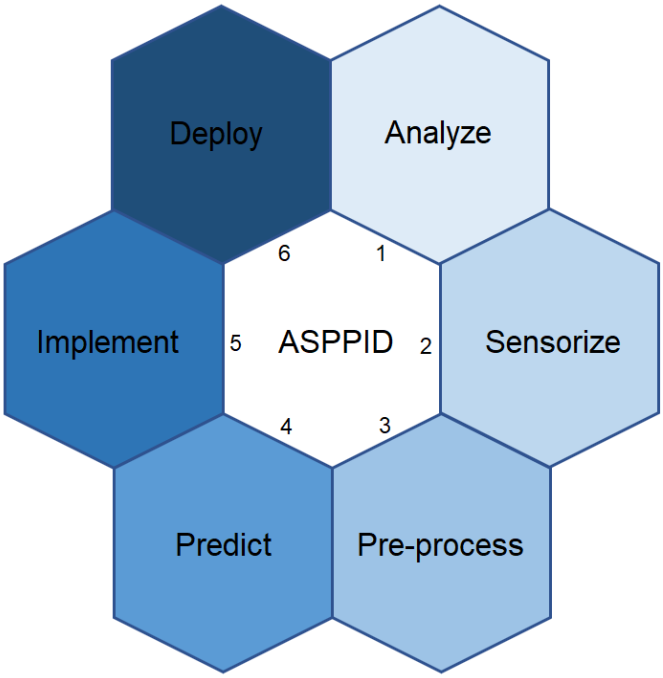

Figure 1.1: Methodology Workflow. Adapted from [30].

distribution and the ability of different algorithms to understand the differences between the classes.

1.3.2 Sense

After the analysis of the problem, the next step is to prepare the system to collect the required data, this is done through the use of sensors. Can be performed under several scenarios.

Thanks to the work done by the authors of the dataset, the completion of this phase was easier, since the dataset was well differentiated between the different components and the various data sources existing in each one.

1.3.3 Preprocess

In this step, data preparation is carried out with the objective of building a model to achieve the project's goal. This is done by using different techniques such as cleaning, data scaling, imputation, noise identification or data transformation.

For data preprocessing different approaches have been taken in the other three remaining papers, the second II one focused in the elimination of missing values, while the other two papers III IV show different imputation algorithms approach to perform the treatment of those absent data.

1.3.4 Predict

During the realization of this phase, the interpretability of the model has to be worked on, in order to improve the implementation with respect to the imposed objective. Within the variety of existing scenarios it is important to work with class balancing techniques for an appropriate modeling.

Similar to what was discussed in the previous phase, several data balancing techniques have been proposed in the last three papers II III IV, with the aim of analyzing which ones obtain the best performance in each of the situations.

1.3.5 Implement

During the previous phases, a model was generated and validated in the present phase. It is important that this phase is well documented.

In the implementation of the different proposed models, the use of an SVM classifier has always been chosen to validate the problem, thus allowing a fair comparison between the different approaches.

1.3.6 Deploy

All the knowledge acquired and used throughout the whole process has to be compiled in documents, so that the knowledge obtained can be transmitted to the rest of the people.

All the knowledge generated throughout the different treatments of the dataset is compiled in all the published papers 2, obtaining a certain consistency in them as time goes by.

1.4 Data selection

This paper proposes the use of a component-based dataset [28]. The development of this dataset has been carried out by the University of Bielefeld (Germany).

This robotic system is composed by several components, all of them added to the base PatrolBot, which is powered by the platform GuiaBot by the company OmronAdept Technologies.

The robot is equipped with different items, such as an arm, a laser for detecting people or two RGBD cameras for object recognition, among others. The sensor components provide with information about the environment. On the other hand, there are some other actuator components to manipulate the environment. All these components communicate thanks to the RSB middleware [31]. The so-called Robotics Service Bus (RSB) use event-based, message-oriented communication. There is a tool (rsbag) that collects RSB information that can be used for analysis. This can record some parts of the communication subsystem or the whole system. For the exchange of information between the bus and the user code there are different participants; the listener if the information goes from the bus to the usercode and informer otherwise. This information are events encoded as notifications. For the coordination of the sensor and actuator components, the BonSAI framework is used. It is in charge

of creating RSB participants when a state or transition depends on them. A Finite State Machine (FSM) is applied to represent and control execution flows. A visual summary of the above can be seen in Figure 1.2.

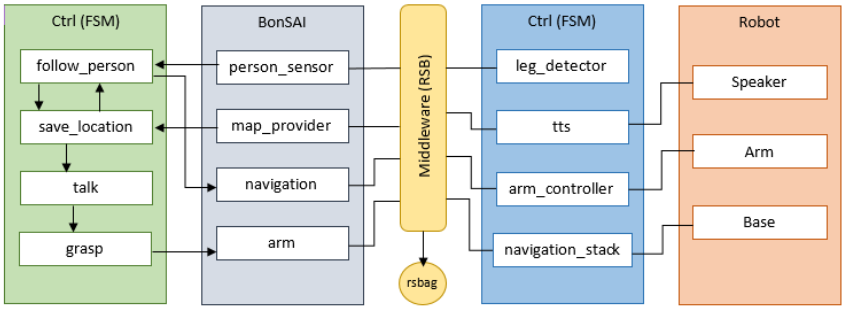

Figure 1.2: Overview of the robot system architecture. Extracted from paper I

The dataset is divided into the values collected by each of the different components, within each component there are three different types of datasets: features, counters and events. The data collected, are the different states of the component at a moment in time, the temporal periodicity is the same in features and counters, but not in events. The data of features are the data of events but put in the temporal arrangement of counters, so we discard the use of events and only use the other two. For a fair understanding of the structure comprising this dataset, see Figure 1.3.

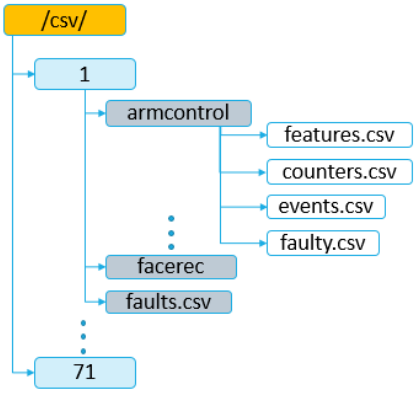

Figure 1.3: Structure of the dataset under analysis. Extracted from paper II

The different components have anomalies induced in them by software, this implies that the system counters are penalized, but it does not prevent the component from carrying out the task, for example, the legdetector component

is affected by the legDetectorSkippable anomaly which performs an analysis several times of the subject's legs, even though it has performed well at first.

This data set has been induced with different anomalies by software, this does not imply that the task is not carried out but that the performance is affected in the system counters, for example the component in charge of leg detection performs several scans instead of just one. The purpose of inducing these anomalies is to be able to detect future incidents in the components. In the data set, the time instants in which anomalies have been triggered are reflected and a labeling of the anomalies can be obtained.

There are 71 trials in which the experimentation has been repeated, these always reproduce the experiment in the same order making the data set consistent. Anomalies are not induced in all trials, only in some cases. The authors define those data sets that are suitable for experimentation because for various reasons some are either invalid or have unidentified anomalies.

The scenario in which the robot acts tries to imitate the behavior of a waiter, where it has to detect a human, have the ability to communicate with him, move around a defined space, have the ability to detect objects as well as pick up a glass and serve it to the customer.

The anomalies which affect to the different components are the followings: armServerAlgo, legDetectorSkippable, objectBuilderSkippable, clafuSleep, pocketSphinxLeak, btlAngleAlgo, bonsaiParticipantLeak, bonsaiTalkTimeout and facerecSkippable, clockShift and spreadLatency. All of them except the last two affect a single component, making it easier to isolate them for handling.

As previously mentioned, the anomalies are induced in various ways, but in the end the objective is that the meters are affected. Throughout the thesis we have specialized in the analysis of two main anomalies, armServerAlgo, which affects the armcontrol component and legDetectorSkippable which affects the legDetector component. The case of the latter has already been discussed, in the case of the former the anomaly affects the component leading it to perform different unnecessary movements in the arm. The anomalies are of different natures, mathematical and logical as in the case of armServerAlgo, or skippable communication in the case of legDetectorSkippable, and other anomalies affect resources leak, on the configuration, threading or inter-process communication. Table 1.1 provides an abbreviated description of the way in which each anomaly acts.

Further details are provided in the different sections of the data explanation in the papers II.3 III.3 IV.4. On the other hand, this data set is much more in-depth in the original author's research [27] [32].

References

[1] Khalastchi, E. and Kalech, M. "On Fault Detection and Diagnosis in Robotic Systems". In: *ACM Comput. Surv.* vol. 51, no. 1 (Jan. 2018), pp. 1–24.

Table 1.1: Brief description of each anomaly. Extracted from paper II.

Code	Name	Description
A1	armServerAlgo	Certain movements of the arm are performed from known valid poses
A2	legDetectorSkippable	The 'legdetector' processed each scan multiple times
A3	objectbuilderSkippable	The person tracking performed transformations for each person multiple times
A4	clafuSleep	The results are returned only after a delay of 5 seconds
A5	pocketSphinxLeak	The speech recognition component accumulates memory for each sound
A6	btlAngleAlgo	Adds a mathematical error used to track people
A7	bonsaiParticipantLeak	Participants are not cleaned up properly
A8	bonsaiTalkTimeout	Configuring a wrong RSB scope for the text-to-speech engine
A9	fecrecSkippable	Temporarily removes a throttling of the main loop of the 'facerec' component

[2] Dares, M. et al. "Development of AGV as Test Bed for Fault Detection". In: *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*. 2020, pp. 379–383.

[3] Janarthanan, R., Doss, S., and Balamurali, R. "Robotic-based nonlinear device fault detection with sensor fault and limited capacity for communication". In: *Journal of Ambient Intelligence and Humanized Computing 2020 11:12* vol. 11, no. 12 (Apr. 2020), pp. 6373–6385.

[4] Xu, X., Liu, H., and Yao, M. "Recent Progress of Anomaly Detection". In: *Complexity* vol. 2019 (2019).

[5] Ranshous, S. et al. "Anomaly detection in dynamic networks: a survey". In: *Wiley Interdisciplinary Reviews: Computational Statistics* vol. 7, no. 3 (2015), pp. 223–247.

[6] Jove, E. et al. "A fault detection system based on unsupervised techniques for industrial control loops". In: *Expert Systems* vol. 0, no. 0 (2019), e12395.

[7] Dorigo, M. and Schnepf, U. "Genetics-based machine learning and behavior-based robotics: a new synthesis". In: *IEEE Transactions on Systems, Man, and Cybernetics* vol. 23, no. 1 (Jan. 1993), pp. 141–154.

[8] Lu, H. et al. "Motor Anomaly Detection for Unmanned Aerial Vehicles Using Reinforcement Learning". In: *IEEE Internet of Things Journal* vol. 5, no. 4 (Aug. 2018), pp. 2315–2322.

[9] Jayaratne, M., de Silva, D., and Alahakoon, D. "Unsupervised Machine Learning Based Scalable Fusion for Active Perception". In: *IEEE Transactions on Automation Science and Engineering* vol. 16, no. 4 (Oct. 2019), pp. 1653–1663.

[10] Kober, J., Bagnell, J. A., and Peters, J. "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* vol. 32, no. 11 (2013), pp. 1238–1274.

[11] H. Alsamhi, s., Ma, O., and Ansari, M. S. “Survey on artificial intelligence based techniques for emerging robotic communication”. In: *Telecommunication Systems* vol. 72, no. 3 (Nov. 2019), pp. 483–503.

[12] Zhao, D., Ni, W., and Zhu, Q. “A framework of neural networks based consensus control for multiple robotic manipulators”. In: *Neurocomputing* vol. 140 (2014), pp. 8–18.

[13] Xiao, B. and Yin, S. “Exponential Tracking Control of Robotic Manipulators With Uncertain Dynamics and Kinematics”. In: *IEEE Transactions on Industrial Informatics* vol. 15, no. 2 (Feb. 2019), pp. 689–698.

[14] Susan, S. and Kumar, A. “The balancing trick: Optimized sampling of imbalanced datasets—A brief survey of the recent State of the Art”. In: *Engineering Reports* vol. 3, no. 4 (Apr. 2021), e12298.

[15] Dudjak, M. and Martinović, G. “An empirical study of data intrinsic characteristics that make learning from imbalanced data difficult”. In: *Expert Systems with Applications* vol. 182 (Nov. 2021), p. 115297.

[16] Chawla, N. V. et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* vol. 16 (2002), pp. 321–357.

[17] Fernandez, A. et al. “SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary”. In: *Journal of Artificial Intelligence Research* vol. 61 (Apr. 2018), pp. 863–905.

[18] Fernández, A. et al. “Data Level Preprocessing Methods”. In: *Learning from Imbalanced Data Sets*. Cham: Springer International Publishing, 2018. Chap. 5, pp. 79–221.

[19] Schafer, J. L. “Multiple imputation: a primer”. In: *Statistical methods in medical research* vol. 8, no. 1 (1999), pp. 3–15.

[20] Plaia, A. and Bondi, A. “Single imputation method of missing values in environmental pollution data sets”. In: *Atmospheric Environment* vol. 40, no. 38 (2006), pp. 7316–7330.

[21] Yale, U. of. *Multiple Linear Regression*. 2017.

[22] Moisen, G. G. *Classification and Regression Trees*. 2018.

[23] Lippmann, R. P. “Pattern classification using neural networks”. In: *IEEE Communications Magazine* vol. 27, no. 11 (Nov. 1989), pp. 47–50.

[24] Arroyo, A. et al. “Neural Models for Imputation of Missing Ozone Data in Air-Quality Datasets”. In: *Complexity* vol. 2018 (Mar. 2018).

[25] Twala, B. “Robot execution failure prediction using incomplete data”. In: *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. Dec. 2009, pp. 1518–1523.

[26] Basurto, N. and Herrero, Á. “Data Selection to Improve Anomaly Detection in a Component-Based Robot”. In: *14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019)*. Ed. by Martínez Álvarez, F. et al. Cham: Springer International Publishing, 2020, pp. 241–250.

[27] Wienke, J., Meyer zu Borgsen, S., and Wrede, S. “A Data Set for Fault Detection Research on Component-Based Robotic Systems”. In: *Towards Autonomous Robotic Systems*. Ed. by Alboul, L., Damian, D., and Aitken, J. M. Vol. 9716. Cham: Springer International Publishing, 2016, pp. 339–350.

[28] Wienke, J. and Wrede, S. *A Fault Detection Data Set for Performance Bugs in Component-Based Robotic Systems*.

[29] Wienke, J. and Wrede, S. “Autonomous fault detection for performance bugs in component-based robotic systems”. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 3291–3297.

[30] Para, J. et al. “Analyze, Sense, Preprocess, Predict, Implement, and Deploy (ASPPID): An incremental methodology based on data analytics for cost-efficiently monitoring the industry 4.0”. In: *Engineering Applications of Artificial Intelligence* vol. 82 (June 2019), pp. 30–43.

[31] Wienke, J. and Wrede, S. “A middleware for collaborative research in experimental robotics”. In: *2011 IEEE/SICE International Symposium on System Integration (SII)*. Dec. 2011, pp. 1183–1190.

[32] Wienke, J. “Framework-level resouce awareness in robotics and intelligent systems”. PhD dissertation. Bielefeld University, 2018.

Chapter 2

Thesis Format

The thesis has been prepared in the form of a compendium of articles. In addition, the thesis is eligible for the international doctoral degree. The articles which are part of the thesis are listed below.

2.1 Selected papers

Paper I

The paper focuses on the study of anomalies for the software of a component-based robot. To deal with the problem, the use of the Hybrid Unsupervised Exploratory Plots visualization technique is proposed, improving it by using other projectionist techniques, such as Curvilinear Component Analysis (CCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE). The use of this technique allows obtaining interesting improvements in the detection of robot anomalies.

- **Authors**: Nuño Basurto, Carlos Cambra, Álvaro Herrero
- **Title**: A Visual Tool for Monitoring and Detecting Anomalies in Robot Performance
- **Journal**: Pattern Analysis and Applications - Tentatively Accepted
- **Year/Rank**: 2021 JCR 74/139 (COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE) - Q3
- **DOI**: Not available

Paper II

It proposes the use of different mechanisms for the improvement of detection rates in robot anomalies. Focusing on the different ways to deal with missing values and high unbalance between classes. For the validation of the results the cross validation strategy is used together with the Support Vector Machine classifier.

- **Authors**: Nuño Basurto, Carlos Cambra, Álvaro Herrero
- **Title**: Improving the Detection of Robot Anomalies by Handling Data Irregularities
- **Journal**: Neurocomputing - Published

- **Year/Rank**: 2021 JCR 30/139 (COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE) - Q1
- **DOI**: 10.1016/J.NEUCOM.2020.05.101

Paper III

The application of regression models is proposed in the paper in order to improve fault detection. For this purpose, a classical technique like linear regressions is used and other more innovative ones such as neural networks and decision trees. The main objective is to obtain a higher quality in the data set through the imputation of missing values.

- **Authors**: Nuño Basurto, Ángel Arroyo, Carlos Cambra, Álvaro Herrero
- **Title**: Imputation of Missing Values Affecting the Software Performance of Component-based Robots
- **Journal**: Computers and Electrical Engineering - Published
- **Year/Rank**: 2020 JCR 44/112 (COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS) - Q2
- **DOI**: 10.1016/j.compeleceng.2020.106766

Paper IV

This research proposes the use of a Hybrid Intelligent System divided into four stages, with the objective to obtain a higher classification ratios. In the first two steps, the combination of clustering and regression techniques is carried out in order to impute missing values, followed by data balancing and classification.

- Authors: Nuño Basurto, Ángel Arroyo, Carlos Cambra, Álvaro Herrero
- Title: A Hybrid Machine Learning System to Impute and Classify a Component-Based Robot
- Journal: Logic Journal of the IGPL - Accepted
- Year/Rank: 2021 JCR 2/21 (LOGIC) - Q1
- **DOI**: Not Avaliable

In addition to the papers presented here, there are other journal and conference publications, which are detailed in the following sections.

2.2 Other Journal papers

- **Authors**: Rafael Vega Vega, Héctor Quintián, Carlos Cambra, Nuño Basurto, Álvaro Herrero and José Luis Calvo-Rolle
- **Title**: Delving into Android Malware Families with a Novel Neural Projection Method
- **Journal**: Complexity - Published
- **Year/Rank**: 2019 JCR 31/71 (MULTIDISCIPLINARY SCIENCES) - Q2
- **DOI**: 10.1155/2019/6101697

- **Authors**: Nuño Basurto, Ángel Arroyo, Rafael Vega, Héctor Quintián, José Luis Calvo-Rolle and Álvaro Herrero
- **Title**: A Hybrid Intelligent System to forecast solar energy production
- **Journal**: Computers and Electrical Engineering - Published
- **Year/Rank**: 2019 JCR 50/109 (COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS) - Q2
- **DOI**: 10.1016/j.compeleceng.2019.07.023

2.3 Conference papers

1. **Authors**: Nuño Basurto and Álvaro Herrero
   Title: Data Selection to Improve Anomaly Detection in a Component-Based Robot
   Congress: 14th Computing Models in Industrial and Environmental Applications - SOCO
   Year: 2020
   DOI: 10.1007/978-3-030-20055-8_23

2. **Authors**: Nuño Basurto, Carlos Cambra and Álvaro Herrero
   Title: AI-driven Visualizations for Performance Monitoring and Anomaly Detection in Robots
   Congress: IEEE/ACS International Conference on Computer Systems and Applications - AICCSA
   Year: 2020
   DOI: 10.1109/AICCSA50499.2020.9316513

3. **Authors**: Héctor Quintián, Esteban Jove, José Luis Calvo-Rolle, Nuño Basurto, Carlos Cambra, Álvaro Herrero and Emilio Corchado
   Title: Detecting Performance Anomalies in the Multi-component Software a Collaborative Robot
   Congress: 21st International Conference on Intelligent Data Engineering and Automated Learning - IDEAL
   Year: 2020
   DOI: 10.1007/978-3-030-62365-4_51

4. **Authors**: Ángel Arroyo, Nuño Basurto, Carlos Cambra and Álvaro Herrero
   Title: Clustering and Regression to Impute Missing Values of Robot Performance
   Congress: Hybrid Artificial Intelligent Systems - HAIS
   Year:2020
   DOI: 10.1007/978-3-030-61705-9_8

5. **Authors**: Nuño Basurto, Michał Woźniak, Carlos Cambra and Álvaro Herrero
   Title: Advanced Oversampling for Improved Detection of Software Anomalies in a Robot
   Congress: 15th Computing Models in Industrial and Environmental Applications - SOCO
   Year: 2020
   DOI: 10.1007/978-3-030-57802-2_1

6. **Authors**: Nuño Basurto, Ángel Arroyo, Carlos Cambra and Álvaro Herrero
   Title: A Hybrid Intelligent System to Detect Anomalies in Robot Performance
   Congress: Hybrid Artificial Intelligent Systems - HAIS
   Year: 2021
   DOI: 10.1007/978-3-030-86271-8_35

7. **Authors**: Nuño Basurto, Carlos Cambra and Álvaro Herrero
   Title: Visually Monitoring the Performance of a Component-based Robot
   Congress: 16th Computing Models in Industrial and Environmental Applications - SOCO
   Year: 2021
   DOI: 10.1007/978-3-030-87869-6_11

Chapter 3

Conclusions and Future Work

3.1 Conclusions

First, in the development of this thesis, has been essential to locate a data set that suited our requirements. After finding it, its subsequent analysis was necessary to understand it better and to observe the countless opportunities it could provide. The analysis of the previous work has been thorough, given the need to develop new ways of using the existing data set and to develop new techniques as the development of the thesis progressed and new knowledge was obtained.

After all exposed in the development of this doctoral thesis, it can be concluded that the initial objectives have been satisfactorily achieved. In order to observe in more detail, each of the different approaches will be analyzed separately.

As discussed throughout this thesis, dealing with anomalies from a supervised learning approach, involves an unbalanced data set, which has led to the use of balancing techniques of different categories. The use of these techniques, has allowed to observe how the performance has improved significantly, respect the base value. It has not always been the same techniques which have obtained the best results, this provides greater versatility on the part of the data set.

On the other hand, when dealing with this dataset have also had to deal with the problem of Missing Values, for which have been decided to perform imputation under different perspectives, in which we have observed an improvement in performance. It can therefore be concluded that the use of imputation techniques has enriched the development of this thesis. To highlight the use of mixed techniques of cluster imputation, that is a new approach that had not been presented so far and which has achieved very good results.

It has not been only in the papers of the thesis that a more complete development of the data set in question has been possible, as it has been observed that there are a large number of publications in congresses in which other perspectives have been addressed. The use of visualization techniques has been very interesting in order to have a better overview of the layout of the information. In addition, the development of new balancing techniques has led to a greater enrichment of the knowledge and capabilities of the data set.

3.2 Future Work

As future lines of research, we will focus on the development of multi-class techniques that will allow a better discriminate in anomalies which affect more than one component, in this way can be improved their detection in several components at the same time.

Another interesting approach to consider is the use of time series on the data set, since the data set has the same consistency in its different trials.

In terms of machine learning techniques, it is interesting to consider the use of developed techniques in new areas, for example in the case of the hybrid technique of cluster imputation, which can allow improvements in the performance of the data set.

On a personal side, would be interesting the generation of new balancing algorithms, with special emphasis on the oversampling ones, where have been observed a lot of work done but the combination of existing techniques can rise successful algorithms.

Papers

Paper I

A Visual Tool for Monitoring and Detecting Anomalies in Robot Performance

Nuño Basurto, Carlos Cambra, Álvaro Herrero

Tentatively Accepted in Pattern Analysis and Applications, October 2021

Abstract

In robotic systems, both software and hardware components are equally important. However, scant attention has been devoted until now in order to detect anomalies/failures affecting the software component of robots while many proposals exist aimed at detecting physical anomalies. Accordingly, present paper focuses on the study of anomalies affecting the software performance of a robot by using a novel visualization tool. Recent unsupervised visualization methods from the Machine Learning field are applied in order to upgrade previously proposed Hybrid Unsupervised Exploratory Plots. More precisely, Curvilinear Component Analysis (CCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) are applied and comprehensively compared in present paper. Thanks to this intelligent visualization of robot status, interesting conclusions can be obtained to improve anomaly detection in robot performance.

I.1 Introduction

The European Commission identified smart robotics as an innovation field that would benefit from the development of Key Enabling Technologies (KETs) [1]. Undoubtedly, to successfully deploy autonomous robotics systems, further innovative digital solutions must be conceived and validated in real scenarios. In the past years, plenty of attention has been devoted to deploy such robots, with advanced capabilities not only for autonomous operation but also for self-diagnosis. However, the demands of enhanced systems also lead to a significant increase in the complexity of them, while reliability and robustness are also required. While operating in real-world environments, robots fail and analysing these failures is a keystone in the road to complete autonomy. Both the hardware and software components of robots suffer from failures. The former has been widely researched [2], [3] while little effort has been devoted to the latter so far [4]. In order to bridge this gap, a new tool is introduced in this novel work, based

on previously proposed Hybrid Unsupervised Exploratory Plots (HUEPs) [5], to successfully monitor the performance of software components within a robot while supporting the detection of software anomalies.

Detection of anomalies can be defined as, once expected behavior is known, finding certain patterns in the data that do not conform to it [6]. In order to do that, many proposals based on supervised Machine Learning (ML) are been successfully applied to anomaly detection in a wide variety of industrial problems, ranging from service elevator [7] to solar panels [8] or Unmanned Aerial Vehicles [9], among others. From a complementary perspective, present work investigates the use of unsupervised ML techniques to exploratory study performance in order to know more about anomaly datasets. As a result of the obtained knowledge, the application of some other supervised ML methods may be enhanced, being out of the scope of present proposal. According to recent and comprehensive methodologies for cost-efficiently monitoring the Industry 4.0, "exploratory data analysis must be first done, comprising techniques such as descriptive statistics, dimensionality reduction and clustering, among others" [10]. Coherently, a clustering [11] extension of HUEPs is proposed as a novel combination of some of the above mentioned techniques; more precisely dimensionality reduction (exploratory projection) and Density-based clustering ones.

ML methods based on unsupervised learning has been previously applied for analysing and detecting faults/anomalies. In most cases, these methods have been proposed in a first (pre-processing) stage to be carried out before applying a method based on supervised learning. Under this approach, characteristic components are extracted in [12] by means of PCA. After that, Hidden Markov Models are applied to process these components. This hybrid system is applied for fault detection in three-phase asynchronous machines; the faults to be detected consist of rotor asymmetries caused by the broken bars. Similarly, different EPP methods (such as Isomap, Sammon mapping, and PCA) were applied in [13]. After a preprocessing of the sensory data, these methods were applied in order to reduce the dimensionality and extract features to afterwards detect faults by means of a supervised classifier. Anomaly detection was performed in two industrial fields: bearing balls and electrical faults in an induction motor. Zhang et al. [14] applied a variant of Locally Linear Embedding (LLE), called robust LLE, whose goal is to prevent noise in the data. It was combined with a Support Vector Machine (SVM) classifier on data obtained from a platform that has an engine, a gearbox, and bearings. More recently, authors [15] have proposed a Self-organizing Feature Map as an initial step before applying some variants of SVM. This way, faults have been detected between motor-end and reduction gearbox, and between brake-end and planetary gearbox.

Some other applications of unsupervised ML for anomaly detection have also been proposed. A combination of PCA, k-means, hierarchical, and Fuzzy C-means clustering methods has been proposed in [16] to improve the modelling of the target system through a Gaussian Mixture Model. Located in a manufacturing environment, an exhaust fan has been studied from vibration data. In [17] authors have tried to detect abnormal condition patterns in gearboxes. To do that, supervised ML is proposed: a novel method inspired by the main principles

of the One Nearest Neighbour and k-means methods.

As mentioned above, little effort has been devoted so far to research on anomalies affecting the software of robots. One of the pioneer works is [18], whose authors described the only open dataset [19]. This dataset (comprehensively described in section I.3) contains data about performance indicators of a robotic system, comprising both "normal" and anomalous states. It is analyzed in present work and has been previously studied from a supervised ML perspective. SVMs were applied by the dataset authors [20] in order to automatically detect the software anomalies. In the doctoral dissertation [21] associated to this dataset, that compiles results from all the previous publications by these authors, two different approaches are described. On the one hand, the previously-mentioned application of supervised ML for detecting anomalies and consequently activating automatic reactions in execution time, based on the use of component resources. On the other hand, a set of tools has been developed to understand and systematize resource control under the frame of the robotic system itself. Under the same perspective, supervised ML has been previously applied [22] by present paper authors, trying to improve the classification results by dealing with missing values and applying data balancing techniques.

Differentiating from previous work, unsupervised methods are not proposed as an initial stept for subsequent supervised methods aimed at classifying data (normal/anomalous)). The idea behind the present proposal is to apply unsupervised ML in isolation (visualization and clustering techniques are applied to the same data and then combined) to support the monitoring and study of robot performance status. Thanks to this improved visualization of high-dimensional data, deep knowledge can be gained about the structure of anomaly datasets.

On the other hand, authors [23] have previously explored the application of dimensionality-reduction techniques for the visualization of similar data. However, in this previous work exploratory techniques were not combined with any clustering method, as it is proposed in the present work.

The rest of this paper is organized as follows: the extended HUEPs formulation is described in section I.2 while section I.3 presents the the real-life case and the associated data that are analysed. The results obtained by the proposed solution are presented in section I.4 and the main conclusions are discussed in section I.5.

I.2 Novel Visualization Techniques for HUEPs

Hybrid Unsupervised Exploratory Plots (HUEPs) [5] have been recently proposed as a new visualization tool to combine the outputs of Exploratory Projection Pursuit (EPP) and Clustering methods in a novel and informative way. To address the well-know "curse of dimensionality" challenge and advancing in descriptive data analysis, both EPP and Clustering methods are independently applied and their outputs combined in a new way. More precisely, 3 EPP methods (mainly based on Artificial Neural Networks) were proposed, namely Principal Component Analysis (PCA), Maximum Likelihood Hebbian Learning (MLHL),

and Cooperative MLHL (CMLHL). Going one step further, a visualization extension is proposed in present paper, to improve the original HUEPs formulation. Initially, HUEPs were conceived as a new way of intuitively visualizing data by applying one partitional (k-means) or one hierarchical (agglomerative) clustering method together with one EPP method. Advancing this initial proposal, present paper incorporates more recent visualization methods, as shown in Figure I.1 (original formulation in grey and present clustering extension in blue).

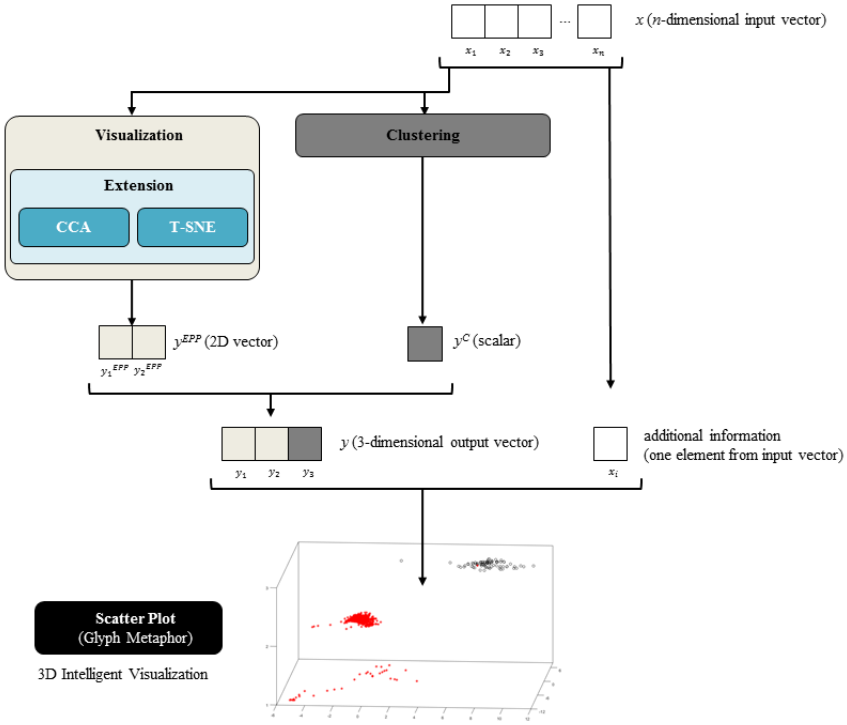

Figure I.1: HUEP novel formulation comprising the proposed Visualization Extension. Adapted from [5].

Visualization is a challenging task, specially when analyzing high-dimensional and real-life data as the one in present research. To address this issue, the proposed visualization extension comprises the following methods, that are applied and validated under the frame of HUEPs for the first time: t-SNE and CCA. These methods are briefly introduced in the following subsections and further details about them can be found in the given references.

I.2.1 Curvilinear Component Analysis

Curvilinear Component Analysis (CCA) was proposed by Demartines and Herault [24] as a self-organizing neural network to find a representation of multidimensional datasets by reducing their dimensionality. To do so, an H-dimensional dataset is projected in an R-dimensional map. CCA is similar to other nonlinear mapping projection techniques such as Sammon's nonlinear mapping [25]. But it mainly differs from these other methods in the use of a new cost function and a greater courage when it comes to represent. It is proposed as an improvement of Sammon's mapping because the latter cannot reproduce all distances, CCA does this by reproducing first the nearest and then the farthest distances. The error function used by CCA is the following one.

$$E_{CCA} = \sum_{i,j=1}^{N} (d_{i,j}^{n} - (d_{i,j}^{p})^2 F_{\lambda}(d_{i,j}^{p}) \tag{I.1}$$

I.2.2 t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding (t-SNE) was proposed by van der Maaten and Hinton [26]. It is a variation of Stochastic Neighbor Embedding [27] producing a better visualization of high-dimensional data while reducing the concentration of points at the same point in the map. This visualization method is able to take into account the concentration of global structures, revealing at the same time the presence of different clusters. One of the tricks used by t-SNE it's the "early compression" which tries to keep the points together at the beginning of the compression, as shorter distance between the points would ease for the clusters to differentiate between them. It tries to reduce the divergence between two distributions, the first of which measures the similarities between the input objects in pairs, while the second measures the similarities in a low dimensionality in pairs of the points. It differs from PCA in that this visualization technique is non-linear, such as CCA. Its operation is as follows; first it creates a probability distribution between the different points and their neighbors. Then, t-SNE create a visualization with a lower dimensionality as a result of that distribution.

I.3 Analysing Performance Anomalies in Robots

As previously mentioned, present paper proposes HUEPs to study performance anomalies in the middleware of a component-based robot by analyzing an open dataset [19]. The development of this dataset was carried out by different researchers at the University of Bielefeld (Germany), for which different metrics of a robot were recorded during a test run on several times. This robot is based on a model developed by Omron Adept Technologies (former MobileRobots).

The robotic system has different components added to the base, which is why it is called a component-based robot. These components may have been developed by different companies and integrated, forming the robotic system, some of the different elements used are: sensors and cameras capable of

performing object recognition, voice recognition or person tracking, actuators responsible for navigation or translation of sentences from text to voice or an arm that performs human-like movement, along with a grip capable of lifting objects. All the information obtained from the different sensors together with the actions to be performed by the actuators are integrated thanks to the RSB Middleware [28] that provides communication among the components. The coordination of the system is carried out by the BonSAI Framework allowing the system to behave as a finite state machine.A graphical summary of the robot structure, comprising these elements, can be seen in figure I.2.

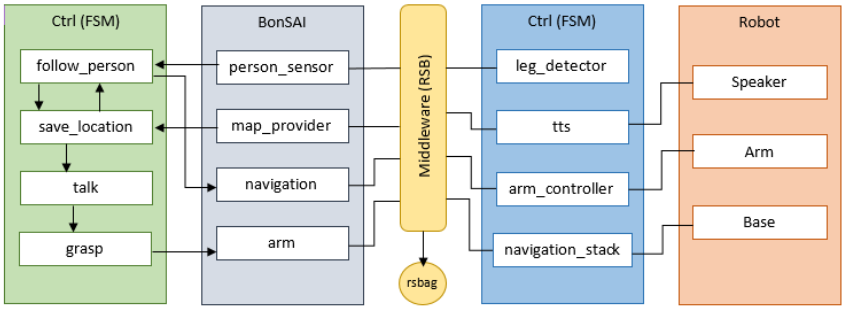

Figure I.2: Overview of the robot system architecture. Adapted from [18].

The authors of the dataset provide documents in which they indicate in which moments the anomalies are induced and on which components, allowing to perform a correct labelled of the data instances. They also comment on the time instants in which not planned anomalies occur in the system.

There are several anomalies that have been induced and affect one or several components. There are 11 the anomalies induced; armServerAlgo, legDetectorSkippable, objectBuilderSkippable, clafuSleep, pocketSphinxLeak, btlAngleAlgo, bonsaiParticipantLeak, bonsaiTalkTimeout and facerecSkippable just affect to one component, whereas clockShift affects four components and SpreadLatency affects all components. In present research two anomalies have been chosen (armSeverAlgo and legDetectorSkippable), according to the investigations of the creators of the dataset [20]. The first one (armSeverAlgo) got the worst results when evaluating the dataset through SVM, while for the second one (legDetectorSkippable), very accurate classification values were obtained. ArmServerAlgo (hereinafter referred as A1) affects the *armcontrol* component, that takes care of the movement of the mechanical arm as well as the movement of the gripper. The anomaly causes the arm to make a series of extra moves to carry out an action, which penalises the performance but does not prevent the task from being carried out. The anomaly legDetectorSkippable (hereinafter referred as A2) affects the *legdetector* component, that is in charge of detecting people's legs for recognizing the presence of human beings. Similarly to what happened with the armServerAlgo anomaly, several unnecessary were performed, thus affecting the performance but not preventing the task to be carried out.

When collecting information from the robot, it performs different tasks in a certain order. The different tasks affect the different components discussed above. This has been repeated a total of 71 times, which is why the data set have 71 trials. Not all of them are used in the experimentation because some of them have undesired anomalies.

For each one of these 71 trials, all the information gathered from each one of the components is available. Dataset contain the information of when and which anomalies were induced together with data from several data sources. The first one of these data sources are the *counters*, that export the performance counters of each one of the robot components or a regular basis (less than a second). The information from this source includes figures such as the active processor threads or the amount of information sent and received. The second source are the *events*, those that were sent between the system bus and the user code. This information tells which are the relevant events in the component, including the sending and receiving address or the size information. For every second of execution of the trial there can be multiple instances that appear in this dataset, being approximately 60 times larger than those observed in counters. That's why the third of the data sources (*features*) is required. It combines events with performance counters in counters time instants.The events presented in features will be the last one received as well as the averages for different temporal moments. This causes the features to be the dataset with the highest dimensionality. The authors published more detailed information on the dataset as well as the source files in [19]. From these three sources of data, features and counters are analyzed in present work as in previous experiments they were identified as those that implied best results [29].

Due to length limitation, results from all the anomalies and trials can not be included in present paper. In order to evaluate the proposed HUEPs extension under different circumstances, experiments were conducted including only one trial and all the trials for two anomalies. The motivation for that is to check whether the HUEP visualizations are equally useful depending on the amount of trials (and data instances) to be depicted.

Accordingly, anomalies were initially selected. The motivation for such selection is that A1 obtained the worst classification rate by the dataset authors when they use an One-class SVM classifier, while on the other hand the A2 anomaly is one of those that obtained best classification rates [20]. Then, trials for these 2 anomalies had to be selected. Among all the trials containing examples of the A1 and A2 anomalies, only one was chosen for each case. Firstly, the smallest trial (lowest number of data instances) was selected. Secondly, for those trials with the same number of occurrences, the one with lowest balance ratio between the normal and the anomaly class (the most unbalanced) was selected. As a result, the proposed HUEPs extension is validated both on small and unbalanced datasets.

As previously studied, the trials within this dataset contain missing values (MV), that must be pre-processed before applying most ML methods. In present paper, MV were removed according to a 0% rate [29]. Accordingly, all the features containing any MV were removed for the dataset. As a result, datasets to be analysed in present paper have a different dimensionality.

Figures about some characteristics of the employed datasets (one and all trials) are shown in table I.1.

Table I.1: Characteristics of the datasets analyzed in present paper.

Dataset	Trials	Cols.	Normal Data	Anomalous Data	Total Data
A1Trial41	1	87	500 (86.2%)	80 (13.8%)	580
A2Trial36	1	23	638 (88.9%)	80 (11.1%)	718
A1AllTrials	10	42	5773 (84.2%)	1032 (15.2%)	6805
A2AllTrials	12	21	6628 (14.5%)	1128 (85.5%)	7756

In addition to the visualization extension that is above described, the application of HUEPs to present problem unveils another novelty. The additional information that is provided to the HUEPs consist now on the class information for each data. Consequently, the glyph metaphor is used to depict the data according to the class (normal versus anomalous states) it belongs to. This way, hybridization in HUEPs is maximized as on the top of the combination of unsupervised methods, class information is used to better understand the patterns associated to anomalous states of the robot software. Thanks to it, the structure of the dataset and the results of other ML techniques can be easily understood.

I.4 Experiments and Obtained Results

In this section, the advance visualizations obtained by applying HUEPs (original and extended formulations) are shown. For the visualization and clustering methods, parameters have been tuned according to previous recommendations [30], [31]. As previously mentioned, the glyph metaphor is applied by using the class (normal versus anomalous) information, once the data are located in the 3D output space. Accordingly, normal data are depicted as red stars while anomalous data are depicted as black circles.

As there are many different visualizations obtained by combining several EPP (PCA, MLHL, CMLHL, CCA, and t-SNE) and clustering (k-means and Agglomerative) methods for each one of the datasets and parameter values, only some of them can be included in present paper. Visualizations have been selected according to different comparison criterion and are shown in the following subsections. Firstly, results for the A1 anomaly are shown in subsection I.4.1 and then, results for the A2 anomaly are shown in subsection I.4.2, comprising visualizations of both one and all-trial datasets each. One-trial datasets have been analyzed in present paper to check the ability of extended HUEPs to visually depict the structure of small datasets. On the contrary, all-trial datasets have been included in the comparative study as they contain a great amount of data.

I.4.1 A1 Anomaly Results

In order to compare the large number of visualizations for the A1 anomaly (A1Trial41 and A1AllTrials datasets) discussed above, only some of them are shown in present subsection for each one of the datasets. These HUEPs consist of the combination of the same clustering method (Agglomerative) and cluster number (5), together with 4 different visualization methods, namely PCA, MLHL, t-SNE, and CCA.

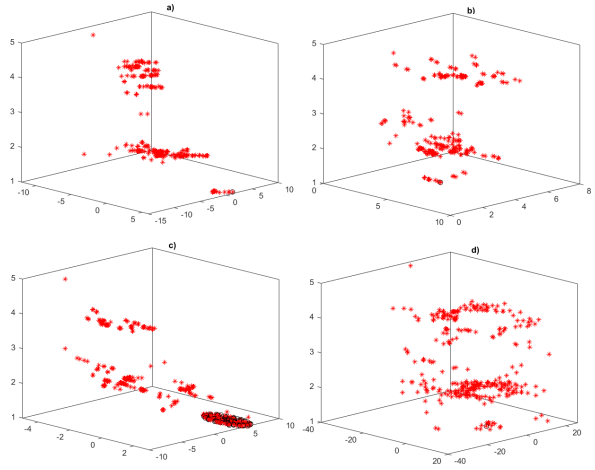

Figure I.3: HUEP visualizations for A1Trial41 dataset. Agglomerative with 5 clusters + projection techniques. a) PCA, b) MLHL, c) t-SNE, and d) CCA.

HUEPs visualizations of the one-trial (A1Trial41) dataset are shown in Fig. I.3. Before analyzing these visualizations, it is worth mentioning that in this dataset all the anomalous instances (80) are duplicated. That is, for all the features in the dataset, all of them took the same value at the different times when data were captured. This phenomenon can be observed in Fig. I.3 as for most of the visualization methods (PCA, MLHL, and CCA) anomalous instances overlap and are depicted as a single data, containing not only the anomalies but also normal data. However, it is not the same in the case of the t-SNE visualization, where anomalies do not overlap. Additionally, all these anomalous instances are grouped together. They are depicted as a group in the right-bottom corner of Fig. I.3.c). This way, it is easily seen that the group is formed of both anomalous and normal data.

For comparison purposes, 3D visualizations obtained by the same techniques are shown in Fig. I.4. It means that the three components of each data are only calculated by the visualization technique. That it, no clustering results are combined but the glyph metaphor is applied (depicting in a different way normal an anomalous data).

The 3D visualizations are quite similar to those obtained by HUEPs. Anomalous instances overlap except for the t-SNE visualization. Furthermore,

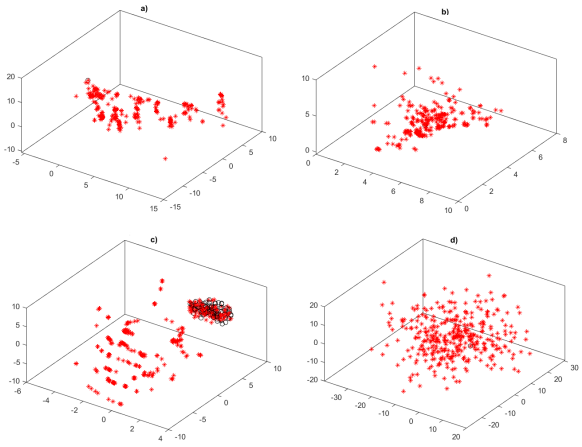

Figure I.4: 3D visualizations for A1Trial41 dataset: a) PCA, b) MLHL, c) t-SNE, and d) CCA.

data are not clearly grouped in the case of the 3D PCA and t-SNE visualizations. It is worth mentioning that in the case of CCA, the 3D visualization is almost useless as there is only one group containing all the data and the structure of the dataset is not revealed at all. In the 3D visualizations, data groups are less clearly defined that in the corresponding HUEPs. This way, it is shown one of the contributions of HUEPs, that is reinforced thanks to the novel visualizations techniques (t-SNE in this dataset): the representation capability of the visualization techniques is improved by adding the clustering information.

Similarly to the one-trial dataset, some visualizations of HUEPs are shown for the all-trial (A1AllTrials) dataset in Fig. I.5. For a fair comparison with the one-trial visualizations, the same clustering method (Agglomerative) is applied and together with 4 different visualization methods, namely PCA, CMLHL, t-SNE, and CCA. In this case, a higher number of clusters (10) is set in order to check the ability of HUEPs to depict such results.

It can be seen in Fig. I.5 that the high number of groups in the clustering causes poor visualizations where groups can be hardly identified and normal/anomalous data are not clearly separated. The worst visualization is obtained by CCA as there is no data grouping at all.

I.4.2 A2 Anomaly Results

As it has been previously explained, HUEPs are also applied to the A2 anomaly (A2Trial36 and A2AllTrials) datasets. Firstly, HUEPs visualizations for the one-trial dataset are shown. Each one of these figures shows the HUEPs obtained by combining the output of the new visualization techniques proposed in this research (CCA and t-SNE), together with the clustering techniques previously mentioned: k-means and Agglomerative. This way, a direct comparison between the two

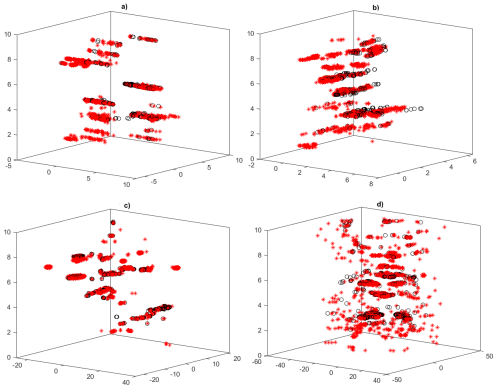

Figure I.5: EHUEP visualizations for A1AllTrials dataset. Agglomerative witch 10 clusters + projection techniques. a) PCA, b) CMLHL, c) t-SNE, and d) CCA.

novel techniques is provided for this dataset. Their visualization performance is also validated in conjunction with the clustering techniques by using a different number of clusters.

In order to compare the performance of HUEPS, executions have been performed with different number of clusters: 3 and 9. 3 was chosen as a relatively small number of clusters while 9 was chosen as a large one, although a larger value (10) has been used to obtain the visualization of the all-trials dataset. Thanks to it, the visualization ability of extended HUEPs is validated for clustering results comprising both low and high numbers of clusters.

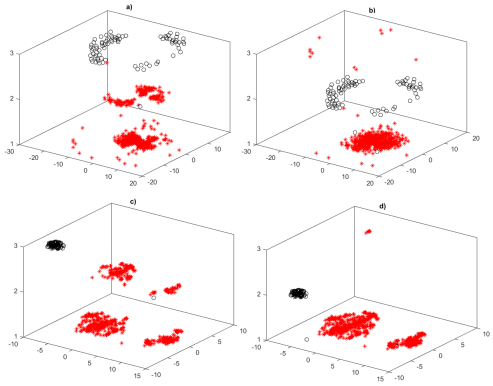

Figure I.6: EHUEP visualizations for A2Trial36 dataset. K-means and Agglomerative with 3 clusters + projection techniques. a) CCA + k-means, b) CCA + Agglomerative, c) t-SNE + k-means, and d) t-SNE + Agglomerative.

HUEP visualizations of the one-trial (A2Trial36) dataset with 3 clusters are shown in I.6, where the dataset structure is clearly depicted in all cases. The

group containing the anomalous data is more sparsely depicted by CCA while t-SNE then to concentrate the data in a group of small size. However, it is worth highlighting that in the case of t-SNE (c and d), the anomalous data is almost completely isolated from the normal one. Similarly, the clustering techniques are not able to split all normal and anomalous data: in the case of the Agglomerative clustering technique (b and d), there are two anomalous instances grouped with the normal ones. In the case of CCA (a and b) these instances are closer to the rest of the anomalous instances. Thanks to this visualization, these anomalous data have been further studied. In a more detailed analysis, it has been observed that they are data instances associated to those moments of time in which the induction of anomalies begins and ends. Thanks to the HUEP visualization, such situations can be identified, leading to a subsequent improvement on the anomaly detection. All in all, it can be said that HUEPs extended by the new visualization techniques are able to depict the dataset in a way that its structure is clearly revealed. As a result, anomalous data can be clearly identified and split from normal data.

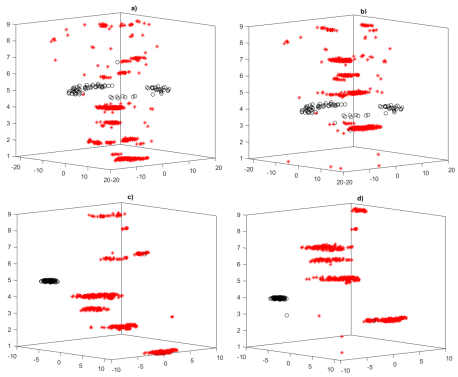

Figure I.7: EHUEP visualizations for A2Trial36 dataset. K-means and agglomerative with 9 clusters + projection techniques. a) CCA + k-means, b) CCA + Agglomerative, c) t-SNE + k-means, and d) t-SNE + Agglomerative.

In the HUEP visualization obtained with 9 clusters (Fig. I.7), it can be seen how the visualization is less clear than that with a smaller number of clusters (Fig. I.6). As in this previous figure, it can be observed that t-SNE (c and d) more clearly reveals the structure of the dataset as groups are shown in a compact way. CCA (a and b) does not generate visualizations as good a those obtained for a reduced number (3) of clusters. When visualizing same data with a higher number of clusters there is a more confusing separation between data classes. One common characteristic of visualizations in Fig. I.7 is that the normal class is split in a greater number of clusters while the anomalous data are kept in only one group. This group is much more concentrated in the case of t-SNE. By taking this into account, it can be concluded that increasing the number of clusters does not always mean a better visualization.

Finally, HUEPs are shown for all the trials of A2 anomaly (A2AllTrials dataset) in Fig. I.8. These visualizations have been obtained by combining the Agglomerative clustering with the PCA, MLHL, t-SNE, and CCA visualization methods. In this case, a reduced number of clusters (4) has been used.

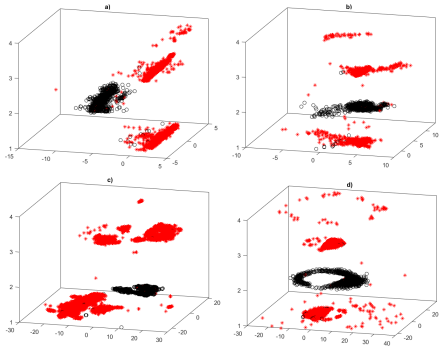

Figure I.8: EHUEP visualizations for A2AllTrials dataset. Agglomerative with 4 clusters + visualization techniques. a) PCA, b) MLHL, c) t-SNE, and d) CCA.

In this case, the visualizations are very similar to those obtained for the one-trial dataset of the same anomaly (A2Trial36). However, Fig. I.8 shows the visualizations obtained by two other techniques: MLHL (b) and PCA (a). Although they have achieved a good separation of clusters, in the case of MLHL the instances of the different classes overlap, that does not happen with any of the other visualization techniques. As for PCA, it achieves a very good separation although slightly worse than that obtained by t-SNE (c). Finally, CCA (d) generates a good separation of clusters, similar to the one previously observed in Figs. I.6 and I.7 but slightly worse. All in all, it can be said that the separation by clusters is quite good, as it happened with the dataset of a single trial (A2Trial36). It can be said that the extended HUEPs can be successfully applied to a large dataset (A2AllTrials dataset is the largest one in present study).

AS in the case of the previous anomaly, the 3D visualizations of A2 by the different techniques are shown in Fig. I.9. They are slightly different to the HUEP ones, in a different way to what happened with the A1 dataset (see section I.4.1). The 3D PCA (a) visualization is quite similar to what can be observed in the corresponding HUEP (Fig. I.8.a), where the two classes are separated but closely located in the output space. Visualization is also quite similar in the case of t-SNE (c), showing a good separation of classes. However, groups containing normal data are more clearly separated in the case of the HUEP visualization. The biggest differences can be observed in the case of the cases of MLHL and CCA. The 3D visualizations are quite bad; groups can not be identified and classes (normal/anomalous) are mixed up.

It can be concluded that HUEPs generate visualizations where the structure of datasets can be observed in a more clear way that in the case of visualizations obtained by other visualizations. By analysing the results of present study,

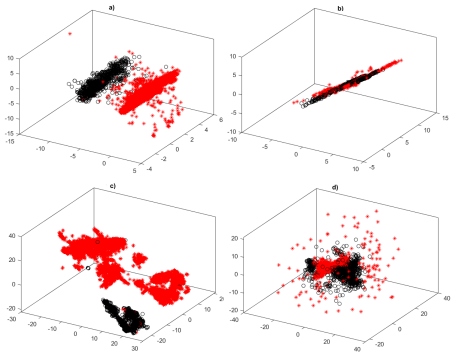

Figure I.9: 3D visualizations for A1Trial41 dataset. a) PCA, b) MLHL, c) t-SNE, and d) CCA.

classes h ave been more clearly grouped in the case of the A2 anomaly rather than in the case of the A1 one.

I.5 Conclusions and Future Work

To monitor the performance of the software components within a robot, a visualization extension of HUEPs is proposed and validated in the present paper. More precisely, CCA and t-SNE are introduced in the HUEPs formulation to benchmark such methods against originally proposed ones (PCA, MLHL, and CMLHL). A comprehensive experimental study has been carried out to validate the proposed extension, comprising experiments on all the different methods, parameters and datasets. As a result, the proposed visualization methods have been applied to the different (small/large and balanced/unbalanced) datasets when applying different visualization techniques as well as clustering ones. Furthermore, results are compared with a varying number of clusters.

From a general perspective, from the experimental results it can be concluded that HUEPs generated by the new visualization methods are better than those obtained by the original methods. It is worth mentioning that among all the methods that are applied for the first time, the t-SNE outperforms the other one, contributing to more informative and intuitive visualizations of normal/anomalous states. This is mainly given to the ability of such method to isolate data instances from one class. From a general perspective, it must be said that in order to select the appropriate visualizations technique, as well as the optimal number ob clusters, a benchmark study must be conducted on each dataset. There is not a visualization technique that clearly outperforms the other one in all cases.

Regarding the dataset size, it can be said that HUEPs visualizations are better for smaller datasets (one-trial). In the visualizations of these datasets data are more clearly grouped and projections are more sparse. In an opposite way, datasets with a larger size (all-trials) are visualized in a less informative

way, although good results have also been obtained for such datasets.

As a follow-up of this research line, authors aim at testing new clustering methods to be applied under the frame of HUEPs. Similarly, new ways of combining different sources of information in the same display could also be investigated in order to improve HUEP visualizations.

References

[1] Commission, E. *Study on cross-cutting KETs (Ro-cKETs)*. en. Text. July 2014.

[2] Khaldi, B. et al. "Monitoring a robot swarm using a data-driven fault detection approach". In: *Robotics and Autonomous Systems* vol. 97 (2017), pp. 193–203.

[3] Park, D., Kim, H., and Kemp, C. C. "Multimodal anomaly detection for assistive robots". In: *Autonomous Robots* vol. 43, no. 3 (Mar. 2019), pp. 611–629.

[4] Khalastchi, E. and Kalech, M. "On Fault Detection and Diagnosis in Robotic Systems". In: *ACM Comput. Surv.* vol. 51, no. 1 (Jan. 2018), pp. 1–24.

[5] Herrero, A., Jimenez, A., and Bayraktar, S. "Hybrid Unsupervised Exploratory Plots: A Case Study of Analysing Foreign Direct Investment". In: *Complexity* (2019), p. 14.

[6] Xu, X., Liu, H., and Yao, M. "Recent Progress of Anomaly Detection". In: *Complexity* vol. 2019 (2019).

[7] Canizo, M. et al. "Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study". In: *Neurocomputing* vol. 363 (2019), pp. 246–260.

[8] Murtada, W. A. and Omran, E. A. "Robust anomaly identification algorithm for noisy signals: spacecraft solar panels model". In: *Neural Computing and Applications* (Aug. 2019).

[9] Khalastchi, E. and Kalech, M. "A sensor-based approach for fault detection and diagnosis for robotic systems". In: *Autonomous Robots* vol. 42, no. 6 (Aug. 2018), pp. 1231–1248.

[10] Para, J. et al. "Analyze, Sense, Preprocess, Predict, Implement, and Deploy (ASPPID): An incremental methodology based on data analytics for cost-efficiently monitoring the industry 4.0". In: *Engineering Applications of Artificial Intelligence* vol. 82 (June 2019), pp. 30–43.

[11] Jain, A. K., Murty, M. N., and Flynn, P. J. "Data Clustering: A Review". In: *ACM Comput. Surv.* vol. 31, no. 3 (Sept. 1999), pp. 264–323.

[12] Georgoulas, G. et al. "Principal Component Analysis of the start-up transient and Hidden Markov Modeling for broken rotor bar fault diagnosis in asynchronous machines". In: *Expert Systems with Applications* vol. 40, no. 17 (2013), pp. 7024–7033.

[13] Jin, X. et al. “Weighted local and global regressive mapping: A new manifold learning method for machine fault classification”. In: *Engineering Applications of Artificial Intelligence* vol. 30 (2014), pp. 118–128.

[14] Yansheng, Z., Dong, Y., and Yuanhong, L. “Robust locally linear embedding algorithm for machinery fault diagnosis”. In: *Neurocomputing* vol. 273 (2018), pp. 323–332.

[15] Shen, F., Langari, R., and Yan, R. “Transfer between multiple machine plants: A modified fast self-organizing feature map and two-order selective ensemble based fault diagnosis strategy”. In: *Measurement* vol. 151 (2020), p. 107155.

[16] Amruthnath, N. and Gupta, T. “A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance”. In: *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*. Apr. 2018, pp. 355–361.

[17] Cerrada, M., Sánchez, R.-V., and Cabrera, D. “A semi-supervised approach based on evolving clusters for discovering unknown abnormal condition patterns in gearboxes”. In: *Journal of Intelligent & Fuzzy Systems* vol. 34 (2018), pp. 3581–3593.

[18] Wienke, J., Meyer zu Borgsen, S., and Wrede, S. “A Data Set for Fault Detection Research on Component-Based Robotic Systems”. In: *Towards Autonomous Robotic Systems*. Ed. by Alboul, L., Damian, D., and Aitken, J. M. Vol. 9716. Cham: Springer International Publishing, 2016, pp. 339–350.

[19] Wienke, J. and Wrede, S. *A Fault Detection Data Set for Performance Bugs in Component-Based Robotic Systems*.

[20] Wienke, J. and Wrede, S. “Autonomous fault detection for performance bugs in component-based robotic systems”. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 3291–3297.

[21] Wienke, J. “Framework-level resouce awareness in robotics and intelligent systems”. PhD dissertation. Bielefeld University, 2018.

[22] Basurto, N., Cambra, C., and Herrero, Á. “Improving the detection of robot anomalies by handling data irregularities”. In: *Neurocomputing* (2020).

[23] Basurto, N., Cambra, C., and Herrero, A. “AI-driven Visualizations for Performance Monitoring and Anomaly Detection in Robots”. In: *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2020, pp. 1–6.

[24] Demartines, P. and Herault, J. “Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets”. In: *IEEE Transactions on Neural Networks* vol. 8, no. 1 (Jan. 1997), pp. 148–154.

[25] Sammon, J. W. “A Nonlinear Mapping for Data Structure Analysis”. In: *IEEE Transactions on Computers* vol. C-18, no. 5 (May 1969), pp. 401–409.

[26] Maaten, L. van der and Hinton, G. “Visualizing Data using t-SNE”. In: *Journal if Machine Learning Research* vol. 9 (2008), pp. 2579–2605.

[27] Maaten, L. van der and Hinton, G. “Stochastic Neighbor Embedding”. In: *Advances in Neural Information Processing Systems* vol. 15 (2002), pp. 833–840.

[28] Wienke, J. and Wrede, S. “A middleware for collaborative research in experimental robotics”. In: *2011 IEEE/SICE International Symposium on System Integration (SII)*. Dec. 2011, pp. 1183–1190.

[29] Basurto, N. and Herrero, Á. “Data Selection to Improve Anomaly Detection in a Component-Based Robot”. In: *14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019)*. Ed. by Martínez Álvarez, F. et al. Cham: Springer International Publishing, 2020, pp. 241–250.

[30] Schubert, E. et al. “DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN”. In: *ACM Trans. Database Syst.* vol. 42, no. 3 (July 2017), 19:1–19:21.

[31] Sánchez, R., Herrero, Á., and Corchado, E. “Visualization and clustering for SNMP intrusion detection”. In: *Cybernetics and Systems* vol. 44, no. 6-7 (2013), pp. 505–532.

Paper II

Improving the Detection of Robot Anomalies by Handling Data Irregularities

Nuño Basurto, Carlos Cambra, Álvaro Herrero

Published in *Neurocomputing*, October 2021, volume 459. pages 419-431. DOI: 10.1016/j.neucom.2020.05.101.

II

Abstract

The ever-increasing complexity of robots causes failures of them as a side effect. Successful detection of anomalies in robotic systems is a key issue in order to improve their maintenance and consequently reducing economic costs and downtime. Going one step further in the detection of anomalies in robots, different mechanisms to deal with data irregularities are proposed and validated in present paper in order to increase detection rates. More precisely, strategies to overcome missing values and class imbalance are considered as complementary tools to get better one-class classification results. The effect of such strategies is evaluated through cross-validation when applying a standard supervised learning model, the Support Vector Machine. Experiments are run on an up-to-date and public dataset that contains some examples of different software anomalies that the middleware of the robot under analysis may experience.

II.1 Introduction

It is widely acknowledged that in present fourth industrial revolution, knowledge extraction from large volumes of data is a crucial task. There are different facilitators [1] to support the transfer to Industry 4.0 [2], that include Artificial Intelligence in general and Machine Learning (ML) in particular. Among all the resources associated to the smart and future factories, robots play a key role [3]. There has been a 31% increase of industrial robots, reaching 384,000 units in the world in just one year, as reported by the Industrial Federation of Robotics [4]. Furthermore, the annual sales volume of industrial robots have been increasing from last 6 years (2013-2018). In parallel to the increase of sale figures, the complexity of robots is constantly growing over time. Additionally, the demands for robustness and reliability are increasing as well. However, as any cyber-physical system, robots suffer from failures and finding anomalies is

required in order to allow recovery and continuous operation. Further effort must be devoted to anomaly detection in robots as little attention has been paid to it by the international research community until now [5].

Present paper addresses the detection of performance anomalies experienced by the software of a robotic system. Due to the widely acknowledged importance of data pre-processing, different such mechanisms (mainly data balancing and handling of missing values - MV) have been applied in order to better identify anomalies. Based on previous work on the same real-life dataset [6], [7], experiments are conducted by means of the One-Class Support Vector Machine (SVM), that is discussed in section II.4.

The successful detection (and identification in multiclass cases) of anomalies/faults is a challenging task that does not only apply to robots [8], [9], [10]. For the benefit of industrial companies in general [11], and for the automatic anomaly detection in particular, ML techniques have been successfully applied in different fields [12] up to now.

Among the vast amount of classifiers that exist, SVM is one of the most widely applied ones for anomaly/fault detection as it has proved to be a successful model. In [13] it is applied to a multi-sensor motor after preprocessing data by means of the FShort-Time Fourier Transform. The aim is reducing maintenance costs of the electro-mechanical system of the motor. More recently, Zidi et al. [14] proposed the use of SVM in the Wireless Sensor Networks field where anomalies could come from different sources, such as software, hardware or the communication system. SVM was benchmarked against some other well-known classifiers such as Naive Bayes (NB) or Hidden Markov Models, obtaining positive results.

Within the stage of data preprocessing, necessary before the pattern recognition one, data irregularities are usually found and must be overcame [15]. Present paper focuses on two of these irregularities, namely MV [16] and data imbalance [17]. There have been previous proposals to deal with the MV in robot data, such as the one proposed by Twala [18], in which a probabilistic approach is used, based on the a-priori probability of each value determined from the instances in that node which have specified values. Robot failures are detected by applying a well-known classifier: a Decision Tree (DT). The classifier is applied to all available data collected from the sensors of the robot; it does not matter the type of attribute (whether numeric or nominal). The author applied and compared different imputation techniques for handling the MV. As opposed to this previous work, present paper deals with the software of a robot and strategies for discarding MV rather than imputing them, due to the induced error of imputation.

The imbalanced class distribution is another important issue that must be addressed before applying supervised learning techniques. Several approaches to deal with this problem [17], [19] have been proposed up to now. Dataspace weighting [20] was proposed in order to balance the classes by assigning different weights to instances of different classes. As a result, classes have the same total weigh, with a positive impact on classification rate. On the other hand, Cerqueira et al. [21] adopted an approach for dealing with MV similar to the one in present research: deleting them. Additionally, they used the Synthetic Minority Over-

sampling Technique (SMOTE) to get a class-balanced distribution of data that improved the classification performance. The aim of such classification was carrying out a predictive maintenance (that is, detecting anomalies) on the air pressure system of heavy trucks. More recently, another study [22] has been published where SMOTE is applied for anomaly detection. In order to detect abnormal events in an assembly line, data are processed (to remove outliers) with DBSCAN and then SMOTE is applied for data balancing. Finally, Random Forest (RF) is used as the learning model for anomaly prediction. RF is also applied in [23] to detect and classify failures of a vehicle fleet. Additionally, a parameter tuning framework is proposed to overcome the class imbalance problem. Similarly, Luo et al. [24] considered the problem of imbalanced data and its implications in anomaly detection. In order to solve it, they generated new synthetic data samples by means of a technique called Triangle Syntethic Data, that is an extended version of SMOTE. They have used some standard classifiers, such as DT, Logistic Regression, SVM, and NB, so they can verify the universality of the algorithms. Rather than proposing the application of one balancing method, such as SMOTE, present paper is a comprehensive study of the application of different balancing methods to improve anomaly detection.

One-class classification has been also addressed before, together with data balancing techniques. In [25] authors analyze the effect that the imbalance of classes can have in seven one-class and two multiclass datasets. Six classification models, such as NB or SVM are compared when applying the Totem-Links undersampling technique. The model finally proposed by the authors, based on the SVM classifier, managed to improve the tendency of the minority class without affecting the majority class.

In the case of robotic systems, most previous work has been focused on the detection of hardware anomalies while few papers deal with software anomalies, which have been largely ignored. Software failures often occur in robotic systems and their automatic detection requires training data. The problem comes from the difficulty of obtaining the data either because of the lack of execution traces or because the existing registers do not refer to the exact moment in which they are produced. That is why it is difficult to find a dataset generated in a controlled environment where all the information is available. One of the pioneer works on the detection of software anomalies within the framework of component-based robots is [7]. In that paper, authors propose the only publicly-available dataset (further details in section II.3) that gathers data from different performance indicators of a robot. The dataset [26] has been used in present paper as a benchmark dataset due to its interest and novelty. Authors of the dataset applied [27] One-Class SVM (OCSVM) in order to compare its performance with that obtained when using another model. Thus, present paper focuses on the effect of data irregularities on classification by OCSVM.

In the doctoral dissertation [28] associated to this dataset, compiling all the previous papers by these authors, they explored two alternatives. Firstly, they considered methods to understand and systematize resource control, for which a set of tools was developed. On the other hand, they studied the topic that leads to present work: the use of different ML techniques to detect anomalies and allow automatic reactions in execution time, based on the use of component

resources.

The rest of this paper is organized as follows: the proposed framework for anomaly detection (comprising the applied classifier, the pre-processing strategies and the performance metrics) is described in section II.2 while the case study and its associated dataset is described in section II.3. The setup of performed experiments and the obtained results are presented in section II.4. Finally, section II.5 introduces the main conclusions derived from present research and points outs some proposals for future work.

II.2 Proposed Framework for Anomaly Detection

Detection of anomalies is known as the problem of finding certain patterns in the data that do not conform to a expected behavior [8]. This "anomalous" behaviour may be associated to failures or malfunctioning of any kind. Anomaly detection in the software of a robotic system is addressed in present paper by using the framework that is described in this section. The SVM classifier (see section II.2.1) is used as the learning method to be trained on the analyzed dataset (described in section II.3). The applied pre-processing techniques are then explained in section II.2.2 and the different metrics that have been observed in order to compare the performance are described in section II.2.3.

II.2.1 Learning Method

The SVM [29], [30] is a widely-applied classifier that implements the Statistical Learning Theory. The purpose of this shallow ML model is to identify the hyperplane that maximizes the separation margin of data, according to the defined classes in the training dataset. For generalization purposes, it tries to universalize the archetype that will be used to classify the new data samples. This is the Structural Risk Minimization perspective, as opposed to the Empirical Risk Minimization one, that is implemented in other models such as neural networks. SVM for one-class classification (as the anomaly detection in present paper) is a learning model whose loss function is the Hinge function, defined as:

$$L[y, f(x)] = max[0, 1 - yf(x)] \tag{II.1}$$

Being x one of the observations taken from the input data, and y is the class x belongs to. *f(x)* is the output of the SVM itself. During training, the SVM identifies the support vectors that are those data samples that maximize the separation of data. Being S the set of support vectors, α the coefficients of the classifier, and β the coefficients of the predictor, once trained the SVM can be defined as in equation II.2.

$$f(x) = \Sigma_{i \epsilon s} \alpha \cdot y_i \cdot \langle x_i, x \rangle + \beta_0 \tag{II.2}$$

In present paper, a SVM equipped with a sigmoidal kernel function has been used. This function is defined as:

$$k(x, y) = tanh(ax^T y + c) \tag{II.3}$$

II.2.2 Data Pre-processing

As previously stated, two data irregularities are addressed in present work. MV is an issue mainly when working in a field where sensors are involved (as present case study). It is even more important due to the fact that most supervised learning methods (SVM included) can not deal with MV. In present paper, MV are removed from the data in order to apply the mentioned classifier. There are mainly two ways of removing MV: deleting those data instances containing at least one of these values for any of the features or deleting those features containing at least one of these values for any of the data instances. The former causes a reduction in the number of data samples while the latter causes a reduction in the number of features, contributing the two of them to negative effects in the learning of the classifier. In order to find an equilibrium, present paper proposes establishing permissiveness ratios for MV when removing data features. That is, features containing a percentage of MV below the threshold are kept while all the others are removed. For those features that are below the threshold and contain any MV, data instances comprising such MV are subsequently removed. Consequently, by increasing the MV ratio more features are considered by the classifier but less instances and vice versa. To empirically set up such ratio, values of 0%, 10%, 25%, and 50% have been tested in the conducted experiments (see section II.4).

On the other hand, the imbalance of classes often appears in datasets for anomaly detection. It results on the majority class ("normal" status of systems) getting a benefit from the classifier and being prejudicial to the results for the minority class (anomalies/failures). In order to deal with this problem, different solutions (known as balancing methods) have been proposed so far [19]. They are aimed at ensuring that the different classes have a similar number of instances. The different methods to get such a class balance can be classified in 3 main categories [19] by taking into account how do they get a similar number of instances: undersampling, oversampling, and hybrid methods. The methods belonging to each one of these categories that have been applied in present research are:

- **Undersampling methods**: their strategy to get a balanced number of instances per class is creating a new subset by removing some instances. Usually, the data instances to be removed are from the majority classes, so their prominence is reduced in favor of minority classes. The most common and widely used method of undersampling is known as Random Under Sampling (RUS). It is a simple non-heuristic method that gets a class-balanced subset by randomly selecting those instances to be deleted.

- **Oversampling methods**: their strategy to get a balanced number of instances per class is creating a superset by artificially generating data instances. Usually, these new instances are from the minority classes, so their prominence is increased. As in the case of undersampling, there is a common and widely used method of oversampling, known as Random Over Sampling (ROS) that randomly selects the data instances to be duplicated.

A more advanced oversampling method is Synthetic Minority Oversampling TEchnique (SMOTE) [31]. It introduces synthetic data samples created by interpolating different minority-class instances. In order to select these reference instances, k-Nearest Neighbors (KNN) algorithm is applied, as graphically explained in Figure II.1. SMOTE initially selects an x_i minority class instance as a basis for creating new instances of the minority class. By considering Euclidean distance, multiple data samples (Nearest Neighbors) of the minority class (points from x_{i1} to x_{i4}) are chosen from the dataset. Finally, an interpolation is performed in order to obtain new instances ranging from r_1 to r_4.

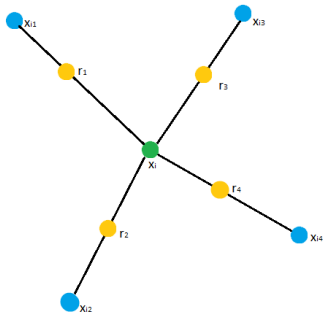

Figure II.1: Syntethic data creation with SMOTE. Adapted from [19].

- **Hybrid methods**: they combine the use of oversampling and undersampling techniques in order to reduce the impact in only one of the classes that the single methods have. One of the hybrid methods that have been applied in present work is ROS + RUS, that combines the two simplest methods of data balancing, previously introduced. Additionally, in keeping with the idea in [31], SMOTE is combined with RUS, generating synthetic instances of the minority class while randomly eliminating instances of the majority one at the same time.

II.2.3 Performance Metrics

The performance of SVM when detecting the anomalies is validated through the standard metrics for supervised learning methods. These metrics are described in this subsection and are calculated from the figures associated to a one-class classification. These figures are the ones usually presented in a confusion matrix, that in anomaly detection are:

- False Positives (FP): normal data that are mistakenly classified as anomalous.
- False Negatives (FN): anomalies that are mistakenly classified as normal data.

- True Positives (TP): anomalies that are correctly classified as such.
- True Negatives (TN): normal data that are correctly classified as such.

Based on these basic statistics that have been previously defined, some useful metrics can be calculated:

II.2.3.1 Accuracy

It can be seen as the global hit ratio, without taking into account whether the data is anomalous or not. It is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{II.4}$$

II.2.3.2 Precision

It is designed to reflect the proportion of data that the given classifier successfully labels as anomalous. This proportion is calculated by taking into account the total number of data labelled as anomalous (TP + FP). It is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{II.5}$$

II.2.3.3 True Positive Rate (TPR)

This metric, also known as Recall, focuses on the relevant data of the problem that, in anomaly detection, are the anomalies. It is similar to Precision but the proportion is now calculated by taking into account the total number of truly anomalous data (TP + FN). It is defined as:

$$TPR = \frac{TP}{TP + FN} \tag{II.6}$$

II.2.3.4 False Positive Rate (FPR)

This metric reflects the proportion of "normal" data that is mistakenly classified as anomalous. This proportion is calculated by taking into account the total number of "normal" data (FP + TN):

$$FPR = \frac{FP}{FP + TN} \tag{II.7}$$

II.2.3.5 F_1 Score

As there are strong dependencies between some of the metrics that has been introduced so far, a new one was conceived in order to reflect a balance between the different aspects to be evaluated. This is the reason to introduce the F_1 Score, that is defined as:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{II.8}$$

II.2.3.6 ROC Curve

The well-known Receiver Operating Characteristic (ROC) curve confronts TPR with FPR in a probabilistic way. It is used to depict the performance of a classifier in a 2D representation. Thus, it supports easily finding the best operating point in order to balance the two metrics (TPR and FPR). Based on this curve, the most important metric for present paper is calculated: the area under the curve (AUC) [32]. Although the previous metrics have also been calculated in all the experiments conducted in present study, it is AUC the top one as it is fair for evaluating classification results on imbalance datasets [17] [33] and it was the one used by authors of the dataset under analysis [27]. As AUC is calculated as a portion of the area of the perfect classification (unit square) it takes values in the range [0, 1]. As a result, the closer the AUC value is to 1, the better.

II.3 Real-life Case Study

As previously stated, present work addresses the detection of performance anomalies in the middleware of a component-based robot. It is done by analyzing a dataset [7] that was generated by researchers from the Bielefeld University (Germany) and is available at [26]. Data were recorded from the *ToBi* robot, whose base is PatrolBot, built upon the research platform GuiaBot, by MobileRobots. As a participant in the RoboCup@Home competition in 2015, its mission was to carry out different tasks related to a waiter's job, such as recognizing clients, asking them about the drink or serving. To complete these tasks, the robot has different components such as two RGBD cameras for person/object recognition, an arm for manipulating objects, and a speech recognition sensor, among others. Through a message-oriented, event-based middleware called Robotics Service Bus (RSB) [34], all the robot components are connected. Data from this RSB associated to different system executions have been captured at runtime thanks to a tool called *rsbag*.

For the induction of anomalies, the authors of the dataset firstly surveyed researchers, university students, and workers through a questionnaire. As a result, the most usual software anomalies for the platform were identified. Then, the anomalies were induced in *ToBi* and were activated through RSB middleware in order to know the precise moment they were produced. 11 anomalies were induced and are present in the dataset, namely: armServerAlgo, legDetectorSkippable, objectBuilderSkippable, clafuSleep, pocketSphinxLeak, btlAngleAlgo, bonsaiParticipantLeak, bonsaiTalkTimeout, facerecSkippable,

clockShift and SpreadLatency. Differentiating from previous work [6], where only one of them (armServerAlgo) was addressed, present paper addresses 9 of them. In order to set a common criteria, anomalies affecting more than one robot component (spreadLatency and clockshift) have been discarded for a fair comparison. Those anomalies analyzed in present work are shown in Table II.1. For the sake of brevity and clarity, a code has been assigned to each one of them, as shown in the first column.

Table II.1: Selected anomalies to be analyzed.

Code	Name	Description
A1	armServerAlgo	Certain movements of the arm are performed from known valid poses
A2	legDetectorSkippable	The 'legdetector' processed each scan multiple times
A3	objectbuilderSkippable	The person tracking performed transformations for each person multiple times
A4	clafuSleep	The results are returned only after a delay of 5 seconds
A5	pocketSphinxLeak	The speech recognition component accumulates memory for each sound
A6	btlAngleAlgo	Adds a mathematical error used to track people
A7	bonsaiParticipantLeak	Participants are not cleaned up properly
A8	bonsaiTalkTimeout	Configuring a wrong RSB scope for the text-to-speech engine
A9	fecrecSkippable	Temporarily removes a throttling of the main loop of the 'facerec' component

The dataset comprises 71 trials, being each one of them an attempt of the robot to perform some of the tasks. Not all trials are included in present work; analyzed trials are those that do not have undetected faults and that are considered valid by the dataset authors. Each trial comprises a file linked to each component of the robot and the different data associated to it. These data refer to the interaction of the robot with its environment during a given time interval and duration varies between trials. Each data instance from the dataset is a sampling of the different data sources at a certain time. Two data sources are used to get information about the performance of the robot software: Features and Counters. The authors of the dataset provided a third set, events, whose information is found in Features. It contains information about the relevant events that occurred in the component, including the size of sending and receiving information. On the other hand, Counters are the raw export of the performance counters for the component, whereas Features are a combination of performance counters and events with the timing of the counters. Further details on data sources are available at [26].

The dataset has the following structure, as described in Figure II.2: the information of each one of the 71 trials is available. There is the information gathered from the different components and (in a faults file) all the induced anomalies, with the starting and ending time as well as its type. For each one of the components, the three sources of data (Features, Counters and Events) are available. Additionally, there is another file that indicates if there is an anomaly

induced in this component and which is the affected time frame.

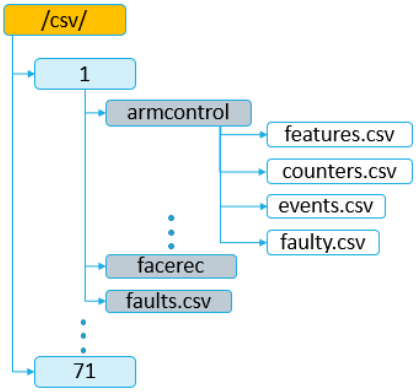

Figure II.2: Structure of the dataset under analysis.

Table II.2: Occurrences of each anomaly and distribution per trials. In bold, the trials selected for the one-trial experiments.

Anom.	1 time	2 times	3 times	4 times
A1	28, 32, 36, 41, 45, 57, 65, 71	21	**23**	
A2	19, 21, 24, 31, 36, 55, 57, 64, 66, 68, 70		**71**	
A3	20, 38, 41, 42, 54, 55, 57, 64, 65, 66	18	**63**	
A4	21, 27, 32, 41, 42, 51, 55, 56	49, 63, **65**, 66		
A5	28, 29, 31, 32, 35, 37, 38, 41, 45, 68			**69**
A6	19, 21, 39, 70, 71	29, 37, **45**		
A7	19, 20, 32, 35, 51, 56, 64	**24**, 68, 70		
A8	23, 24, 28, 35, 36, 38, 39, 42, 45, 49, 56	**18**, 51		
A9	19, 20, 27, 29, 31, 32, 36, 42, 49, 51, **54**, 64			

In order to evaluate the impact of the proposed strategies, experiments were conducted comprising only one trial and all the trials containing examples of a certain anomaly. The underlying idea was to know if the performance of the applied classifier will significantly vary depending on the amount of trials to be considered and the balancing of them. The number of occurrences (from 1 to 4) of each anomaly and the related trials are shown in Table II.2. In bold there can be identified those trials that have been selected for the one-trial experiments. Among all the trials containing examples of an anomaly, those that have a greatest relevance for each one of them were selected individually. The selection criteria of the trial has been firstly the one with the highest number of anomaly occurrences. Secondly, for those trials with the same number of occurrences, it has been selected the one with the longest period of time between each two

anomalies. The main reason is to maximize the time between occurrences in order to let the robot recover from the first occurrence.

Table II.3: Missing values in the dataset per anomaly and data source, with its percentage to total values.

Anomaly	Features	Counters	Features + Counters
A1	366166 (20.4%)	26025 (8.5%)	392191 (19.06%)
A2	17447(4.43%)	24163 (10.03%)	41610 (7.04%)
A3	16492 (3.59%)	23206 (11.78%)	39698 (6.48%)
A4	66308 (11.65%)	24630 (8.04%)	90938 (10.93%)
A5	67884 (6.74%)	25264 (8.25%)	93148 (7.33%)
A6	469662 (9.17%)	19308 (5.88%)	488970 (9.05%)
A7	469662 (9.17%)	19308 (5.88%)	488970 (9.05%)
A8	469662 (9.17%)	19308 (5.88%)	488970 (9.05%)
A9	175074 (21.62%)	48109 (14.65%)	223183 (20.39%)

Table II.4: Class distribution of data per anomaly and trial in the dataset.

Anomaly	All Trials			One Trial		
	Normal Class	Anomaly Class	Anomaly Percentage	Normal Class	Anomaly Class	Anomaly Percentage
A1	20832	1055	5.06%	462	233	50.43%
A2	20765	1127	5.43%	439	206	46.92%
A3	20515	1375	6.70%	445	209	46.97%
A4	20547	1345	6.55%	427	160	37.47%
A5	20738	1147	5.53%	316	320	101.27%
A6	20934	951	4.54%	553	186	33.63%
A7	20837	1048	5.03%	522	160	30.65%
A8	20685	1200	5.80%	554	160	28.88%
A9	20847	1036	4.97%	500	88	17.60%

Since data irregularities are important in present work, some figures about them are provided. Firstly, the total amount of MV in all the trials affected by each anomaly are shown in Table II.3 per anomaly and data source. As it can be seen from this table, the amount of MV significantly varies from one anomaly to the other ones. It is worth mentioning the case of A6, A7, and A8 anomalies as MV amount to 488,970 in all of them, when considering both Features and Counters. This is because they all affect the same robot component, namely statemachine. A study has been conducted to know whether the presence of MV adjusts to a given pattern, but no relevant evidences have been found.

Additionally, it can be observed in Table II.4 the distribution of data in the two classes (normal/anomaly) for the different anomalies and the percentage of data from the minority class (anomalous). These figures are calculated by

setting a MV ratio of zero and this is the reason why the number of instances is the highest one. In the right part of this Table II.4, it can be seen the data distribution in classes for the individual trials. As indicated, these subsets of the data are much more balanced (higher percentage of anomalies) that in the case of the whole dataset (all trials). Furthermore, in the case of A5 anomaly, there are few more instances of the anomaly class than that of the normal class. Then, the effect of applying the pre-processing techniques is checked for similar data in slightly and strongly unbalanced datasets.

Finally, figures about the size of the different datasets are provided in Table II.5. The row-wise datasets are presented per anomaly and data-source and the number of both rows and columns are included. The applied MV ratio (varying from 0% to 50%) is indicated in the case of all trials and in the case of the one trial, there is only one value associated to the 0% MV ratio.

II.4 Experiments and Results

In this section, the results obtained after the execution of the different experiments are shown. 30 of them have been carried out for each one of the anomaly, which amount to 270 experiments in total (9 anomalies are studied). They have been conducted on different subsets of the original dataset (see section II.3): on the one hand, the most significant trial for each one of the anomalies (see Table II.2) has been analyzed (results in section II.4.1) while all the trials have been also analyzed (results in section II.4.2). The best results are presented in each case, regardless the data source (Features, Counters, and both of them). At the end of the section, results associated to the different data sources are presented (see Figure II.5) and discussed.

For each one of these subsets of data, several experiments have been performed with different combinations of the data-preprocessing methods explained in section II.2.2. With regard to the MV issue, different values of the previously explained MV ratio have been applied when analyzing all trials. Once MV had been removed, different experiments have been carried out with a great variety of data balancing methods, namely ROS, RUS, both at the same time (ROS + RUS), SMOTE, and SMOTE with RUS. All in all, one undersampling, two oversampling, and two hybrid methods have been applied for data balancing. Additionally, the obtained performance results are also compared with that for the originally imbalanced dataset without applying any of data-balancing method (referenced as “None”).

All the results presented in this section have been obtained by training a SVM (see section II.2.1) on 75% of the available data while validating on the 25% remaining data. For the validation, the well-known technique of k-fold Cross Validation (with the value $k = 10$) has been applied. Additionally, 10 executions have been carried out per each experiment in order to obtain more statistically significant results. Average results for these 10 executions are shown in present section.

In order to validate the results obtained with the different models and datasets, the non-parametric Wilcoxon Signed-Ranks Test [35] [36] has been

Table II.5: Size of the different datasets per anomaly and data source.

		Rows					Columns				
		0	0.1	0.25	0.5	1 Trial	0	0.1	0.25	0.5	1 Trial
Features	A1	21887	20591	14429	7350	695	35	47	58	68	77
	A2	21892	21892	21892	21892	645	17	17	17	17	17
	A3	21890	21890	21890	21890	654	20	20	20	20	20
	A4	21892	21892	21892	21892	587	22	22	22	22	22
	A5	21885	21885	17175	17175	636	42	42	43	43	43
	A6	21885	18608	5772	-	739	175	189	207	-	213
	A7	21885	18608	5772	2583	682	175	189	207	212	214
	A8	21885	18608	5772	2583	714	175	189	207	212	214
	A9	21883	21883	21883	21883	588	28	28	28	28	29
		Rows					Columns				
		0	0.1	0.25	0.5	1 Trial	0	0.1	0.25	0.5	1 Trial
Counters	A1	21887	21887	18530	18350	695	11	11	13	13	11
	A2	21892	21892	18534	18534	645	8	8	10	10	10
	A3	21890	21890	18533	18533	654	6	6	8	8	8
	A4	21892	21892	18534	18534	587	11	11	13	13	13
	A5	21885	21885	17175	17175	636	12	12	13	13	13
	A6	21885	21885	21885	-	739	14	14	14	-	14
	A7	21885	21885	21885	21885	682	14	14	14	14	14
	A8	21885	21885	21885	21885	714	14	14	14	14	14
	A9	21883	21883	21883	21883	588	12	12	12	12	13
		Rows					Columns				
		0	0.1	0.25	0.5	1 Trial	0	0.1	0.25	0.5	1 Trial
Features + Counters	A1	21887	20591	11762	5974	695	44	56	69	79	86
	A2	21982	21892	18534	18534	645	23	23	25	25	25
	A3	21890	21890	18533	18533	654	24	24	26	26	26
	A4	21982	21892	18534	18534	587	31	31	33	33	33
	A5	21885	21885	17175	17175	636	52	52	54	54	54
	A6	21885	18608	5772	-	739	187	201	219	-	225
	A7	21885	18608	5772	2583	682	187	201	219	224	226
	A8	21885	18608	5772	2583	714	187	201	219	224	226
	A9	21883	21883	21883	21883	588	38	38	38	38	40

used, as it suits present work (compare different models on different dataset without a-priori assumptions). This test supports selecting the best methods on the varied situations under analysis (different datasets and combinations of methods).

II.4.1 One-Trial Experiments

As previously mentioned, experiments were carried out on one single trial per anomaly (that containing most examples of the anomaly as stated in Table II.2). The same pre-processing methods have been applied in the one-trial and all-trial

experiments. However, in the one-trial experiments only one MV ratio was tested: 0%. The reason for that is that none of the selected trials contains MV.

Obtained metrics, calculated on the different data and methods, are shown in the following tables. In Table II.6 it is shown the obtained F_1 values, while the AUC ones are shown in Table II.7.

Table II.6: Obtained F_1 values per anomaly and data-balancing method in the one-trial experiments.

	None	ROS	SMOTE	RUS	ROS + RUS	SMOTE + RUS
A1	0.3194	0.4128	0.4140	0.4229	0.4135	**0.4242**
A2	0.9415	0.9387	0.9492	**0.9424**	0.9397	0.9362
A3	0.8187	0.8226	0.7918	**0.8324**	0.8194	0.8061
A4	0.5486	0.5373	**0.5755**	0.5529	0.5656	0.5595
A5	0.6888	0.7149	**0.7152**	0.6981	0.6907	0.6805
A6	0.4680	0.5995	0.5837	0.6388	0.6114	**0.6628**
A7	0.5853	**0.5881**	0.5532	0.5789	0.5796	0.5396
A8	0.1296	0.2985	0.2774	0.2954	0.2825	**0.3225**
A9	0.7932	**0.9034**	0.8320	0.8093	0.8339	0.8909

Table II.7: Obtained AUC values per anomaly and data-balancing method in the one-trial experiments.

	None	ROS	SMOTE	RUS	ROS + RUS	SMOTE + RUS
A1	0.5354	0.5592	0.5712	**0.5808**	0.5515	0.5737
A2	0.9588	0.9622	**0.9656**	0.9646	0.9582	0.9558
A3	0.8738	0.8796	0.8524	**0.8806**	0.8800	0.8656
A4	0.6926	0.6904	**0.7084**	0.6916	0.7040	0.7061
A5	0.7019	**0.7188**	0.7086	0.6969	0.6986	0.6857
A6	0.6491	0.7584	0.7390	0.7781	0.7571	**0.7903**
A7	0.7353	0.7464	0.7277	**0.7544**	0.7444	0.7188
A8	0.4432	0.5828	**0.5894**	0.5762	0.5885	0.5486
A9	0.8878	**0.9594**	0.9414	0.9546	0.9360	0.9540

When analyzing the F_1-score metric (Table II.6), it can be clearly seen that the application of balancing techniques has greatly improved the obtained values. SMOTE (rather applied in isolation or in connection with RUS) has got the highest F_1 values in 5 cases (out of 9). On the other hand, none of the highest

values has been obtained with the original dataset (no balancing method applied). Results are greatly improved in the case of the A8 anomaly, varying from 0.1296 (no balancing method) to 0.3225 (obtained with SMOTE + RUS).

The AUC scores (Table II.7) are also provided. As it is the key metric (see section II.2.3), a bar graph (see Figure II.3) has been generated in order to ease the comparison of results. Thus, the best AUC scores for each one of the anomalies obtained by each one of the applied methods can be observed. Additionally, the results of statistical test for these two metrics are shown in Tables II.8 and II.9.

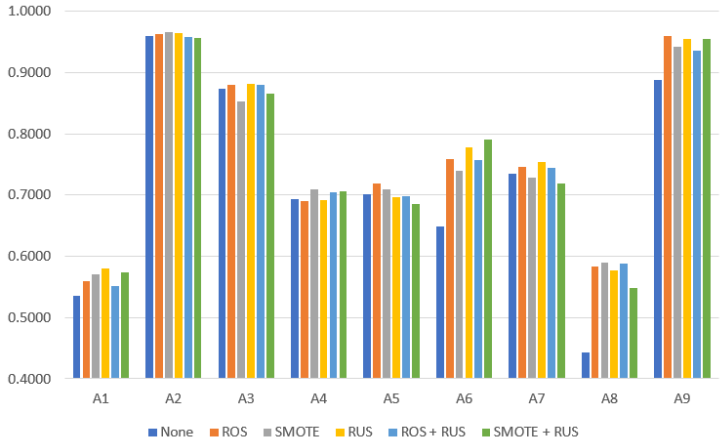

Figure II.3: AUC values per anomaly in the one-trial experiments.

When considering these results, some facts are worth highlighting:

- Analysis by anomaly: AUC values strongly vary from some anomalies to the other ones. While in the case of A1 and A8 anomalies the obtained values are close to 0.5, for most of the anomalies they are in the range [0.6 - 0.9]. In the case of A2 and A9, values higher than 0.9 have been obtained. The case of the A5 anomaly is worth mentioning; it is the only balanced anomaly and as a result, AUC scores are not improved with the balancing techniques. Actually, worst results have been obtained with all the undersampling combinations (RUS, ROS + RUS, and SMOTE + RUS) than that obtained with the original data.

- Analysis by balancing method: when comparing the results obtained with the balancing methods and those obtained from the original dataset, similar values have been obtained except in the anomalies A6, A8, and A9. For these three anomalies, the AUC values from the original dataset are much lower than those obtained with any of the balancing methods. As it can be seen in the Table II.4, these anomalies are three of the most unbalanced ones, except for the A7 anomaly. Going into more detail it has been observed that Features and Features + Counters follow this same pattern

but it is not followed in the case of Counters itself. This trial has fewer instances than the other two anomalies with the same component, A6 and A8. In all anomalies, the best AUC scores are obtained with a balancing method and not with the original data. However, there is not a balancing method that gets better AUC values in all anomalies.

As previously stated, the Wilcoxon Signed-Ranked Test has been carried out to get more statistically significant conclusions from the one-trial experiments. It has been applied to the obtained values for the F_1 score (Table II.8) and AUC (Table II.9) metrics.

Table II.8: p-values obtained by the non-parametric Wilcoxon Signed-Ranked Test pairwaise on the one-trial experiments per balancing method for the F_1 values.

	None	ROS	SMOTE	RUS	ROS + RUS	SMOTE + RUS
None	-	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2
ROS	0.0546	-	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2
SMOTE	0.1132	≥ 0.2	-	≥ 0.2	≥ 0.2	≥ 0.2
RUS	0.0195	≥ 0.2	≥ 0.2	-	≥ 0.2	≥ 0.2
ROS + RUS	0.0546	≥ 0.2	≥ 0.2	≥ 0.2	-	≥ 0.2
SMOTE + RUS	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	-

Table II.9: p-values obtained by the non-parametric Wilcoxon Signed-Ranked Test pairwaise on the one-trial experiments per balancing method for the AUC values.

	None	ROS	SMOTE	RUS	ROS + RUS	SMOTE + RUS
None	-	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2
ROS	0.0078	-	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2
SMOTE	0.0976	≥ 0.2	-	≥ 0.2	≥ 0.2	≥ 0.2
RUS	0.0195	≥ 0.2	≥ 0.2	-	≥ 0.2	≥ 0.2
ROS + RUS	0.0195	≥ 0.2	≥ 0.2	≥ 0.2	-	≥ 0.2
SMOTE + RUS	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	-

According to the values in the Table II.8 (F_1 values) and the values of R^+ and R^-, H_0 for None is not rejected just in the cases of SMOTE and SMOTE + RUS. For all the other balancing methods, it can be said that SMOTE and SMOTE + RUS do not outperform None. The same happens when considering both the R^+ and R^- values and p-values (Table II.9) related to AUC. As a result, it can be concluded that in the case of one-trial experiments, ROS, RUS, and ROS + RUS outperform all the other methods (None included) but any of them can be identified as the best one in all cases.

II.4.2 All-Trials Experiments

As it has been previously explained, same experiments have been run on data subsets containing all the trials for each one of the anomalies. That is, data from all the trials listed in each row of Table II.2 have been merged. As a result, highly unbalanced datasets have been generated, as can be seen in Table II.4.

Differentiating from the experiments on the one-trial datasets, for the all-trials ones 4 different MV ratios have been considered: 0%, 10%, 25%, and 50%. Obviously, with a 0% MV rate, the datasets with the highest number of instances and the lowest number of features (the ones in Table II.4) have been obtained. For the sake of brevity, in the case of all-trials datasets, only best AUC scores for each combination of parameters are shown. Obtained values are compiled in Table II.10 for the MV ratios of 0%, 10%, 25%, and 50% respectively, and all data sources.

From the obtained results, it can be stated that the worst performance has been obtained with the original datasets (no balancing method) for all the anomalies. On the other hand, from these results no clear conclusion can be drawn as to which balancing method works best when stating a 0% MV ratio. With this MV threshold, SMOTE and RUS stand out, being the best ones for 3 of the anomalies each, while the combination of them (SMOTE + RUS) has not obtained the best result for any of the anomalies.

When increasing the MV ratio to 10% (see second section of Table II.10), it can be observed with greater clarity that oversampling methods (ROS and SMOTE) stand out from the rest of balancing methods and the original datasets. Actually, the highest AUC value (0.9822) of all the performed experiments in present work has been obtained with the 10% MV ratio and ROS balancing method for the A2 anomaly. The AUC values for the A9 anomaly are very similar to those with the 0% MV ratio (they only vary for the undersampling methods) because there is no change in the number of MV and hence instances. That is, there is not any original feature containing less than 10% MV. In the case of the 10% MV ratio, it is ROS that stands out (best one for 4 anomalies). Once again, the combination of SMOTE and RUS (SMOTE + RUS) has never obtained the best results, as it has happened with the original data.

From the results with a 25% MV ratio (see third section Table II.10), as opposed to what was pointed our for the 10% MV ratio, the RUS method stands out from the other ones (best results in 5 out of 9 anomalies). As an exception, it has got the worst result of a balancing method in the case of the A6 anomaly. When combined with the SMOTE method (SMOTE + RUS) it has also obtained the highest value for the A1 anomaly. For this anomaly, this combination is the only method that has improved the AUC values when compared with that obtained from the original (unbalanced) data. SMOTE is the second best method, obtaining the highest AUC scores in 2 anomalies. The hybridization of ROS + RUS has not obtained the best value for any anomaly.

Finally, results obtained when applying a 50% MV ratio (see last section Table II.10) are discussed. No results are available for the A6 anomaly because when pre-processing it with that threshold of MV, many instances are eliminated. It causes that none from the anomaly class is kept and hence all the ones

Table II.10: Obtained AUC values per anomaly and data-balancing method. All-trial experiments with 0%, 10%, 25%, and 50% MV ratio.

		None	ROS	SMOTE	RUS	ROS + RUS	SMOTE + RUS
0%	A1	0.5255	0.5747	**0.5841**	0.5639	0.5798	0.5574
	A2	0.8855	0.9805	0.9797	0.9785	**0.9821**	0.9798
	A3	0.7268	0.7842	0.7765	**0.7919**	0.7700	0.7786
	A4	0.5067	0.5704	0.5667	0.5603	**0.5722**	0.5697
	A5	0.4993	0.6945	**0.7053**	0.6994	0.6968	0.6857
	A6	0.5041	**0.5354**	0.5149	0.5144	0.5202	0.5263
	A7	0.5946	0.6770	0.6837	**0.6871**	0.6842	0.6709
	A8	0.4879	0.5225	0.5424	**0.5479**	0.5319	0.5358
	A9	0.6557	0.9059	**0.9110**	0.9098	0.9054	0.9097
10%	A1	0.5155	**0.5531**	0.5368	0.5480	0.5415	0.5225
	A2	0.8855	**0.9822**	0.9797	0.9788	0.9810	0.9814
	A3	0.7268	0.7774	0.7765	**0.7947**	0.7852	0.7719
	A4	0.5067	**0.5775**	0.5667	0.5636	0.5683	0.5740
	A5	0.4993	0.6945	**0.7053**	0.7050	0.6927	0.7013
	A6	0.5067	0.5158	0.5226	0.5183	**0.5260**	0.5243
	A7	0.5701	**0.6806**	0.6712	0.6803	0.6751	0.6796
	A8	0.4884	0.5196	**0.5344**	0.5334	0.5329	0.5315
	A9	0.6557	0.9059	0.9110	**0.9210**	0.9104	0.9086
25%	A1	0.5423	0.5309	0.5289	0.5322	0.5118	**0.5524**
	A2	0.8719	0.9717	0.9702	**0.9745**	0.9702	0.9709
	A3	0.7056	0.7726	0.7711	**0.7928**	0.7835	0.7646
	A4	0.4874	**0.5765**	0.5701	0.5700	0.5693	0.5744
	A5	0.5082	0.6917	**0.7079**	0.6993	0.6891	0.6924
	A6	0.5063	0.7891	**0.8010**	0.7213	0.7704	0.7866
	A7	0.4962	0.6776	0.6744	**0.6837**	0.6823	0.6811
	A8	0.4726	0.5257	0.5386	**0.5825**	0.5438	0.5344
	A9	0.6557	0.9059	0.9051	**0.9210**	0.9104	0.9086
50%	A1	0.5379	0.5670	**0.5699**	0.5646	0.5479	0.5516
	A2	0.8719	0.9717	0.9702	**0.9745**	0.9702	0.9709
	A3	0.7056	0.7726	0.7711	**0.7928**	0.7835	0.7646
	A4	0.4874	**0.5765**	0.5701	0.5700	0.5693	0.5744
	A5	0.5082	0.6917	**0.7079**	0.6993	0.6891	0.6924
	A6	-	-	-	-	-	-
	A7	0.4967	0.9090	**0.9178**	0.7015	0.8458	0.8859
	A8	0.4784	0.6417	0.6472	**0.6671**	0.6439	0.6453
	A9	0.6557	0.9083	0.9095	**0.9161**	0.9159	0.9085

remaining in the dataset belong are from the normal class. For the other anomalies, it can be pointed out that, as it happened for the 25% MV ratio, RUS outperforms the other methods, being the best one 4 times. SMOTE has

obtained the best results 3 times and the hybrid methods have not obtained the highest AUC value for any anomaly.

Table II.11: p-values obtained by the non-parametric Wilcoxon Signed-Ranked Test pairwaise on the all-trials experiments per balancing method for the AUC values.

	None	ROS	SMOTE	RUS	ROS + RUS	SMOTE + RUS
None	-	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2
ROS	0.0039	-	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2
SMOTE	0.0039	0.1641	-	≥ 0.2	≥ 0.2	≥ 0.2
RUS	0.0039	≥ 0.2	≥ 0.2	-	≥ 0.2	≥ 0.2
ROS + RUS	0.0039	≥ 0.2	≥ 0.2	≥ 0.2	-	≥ 0.2
SMOTE + RUS	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	-

Table II.12: p-values obtained by the non-parametric Wilcoxon Signed-Ranked Test pairwaise on the all-trials experiments per MV ratio for the AUC values.

	0%	10%	25%	50%
0%	-	≥ 0.2	≥ 0.2	≥ 0.2
10%	≥ 0.2	-	≥ 0.2	≥ 0.2
25%	≥ 0.2	≥ 0.2	-	≥ 0.2
50%	≥ 0.2	≥ 0.2	≥ 0.2	-

Obtained p-values when applying the Wilcoxon Signed-Ranked Test are calculated per balancing method (Table II.11) and MV ratio (Table II.12). In the case of the balancing methods, the null hypothesis is rejected in all cases for None. Thus, we can conclude that all classifiers obtain a better rank than None. On the other hand, when comparing all the balancing methods, none of them can be designated as the best one for all the datasets in present study as there are not statistical differences.

When analyzing the results per MV ratio, in none of the cases the null hypothesis is rejected, either because of the R or p-value scores. As a result, it can be said that there are no significant statistical differences between the different MV ratios.

As in the case of the one-trial experiments, a bar plot has been generated showing the obtained AUC values for each anomaly, balancing method and MV ratio. In order to also contribute to easily interpret the obtained AUC results, several boxplots have been generated. They are presented, summarizing information according to different criteria: balancing method and MV ratio (Figure II.4), data sources (Figure II.5 a), b), and c)), and anomalies (Figure II.5 d)).

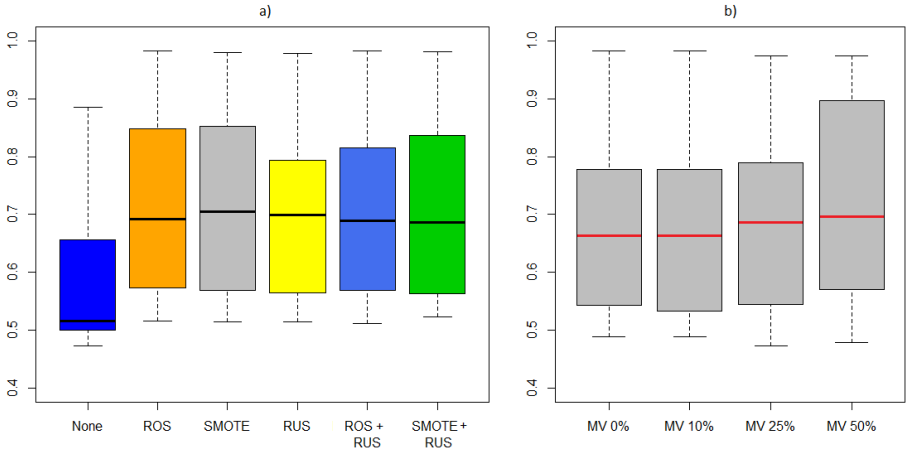

Figure II.4: Boxplot of the obtained AUC values in the all-trials experiments: a) per balancing method and b) per MV ratio.

From all these results and figures, it can be seen that the variance of AUC values is pretty similar for the results obtained with all the balancing methods. On the other hand, the median values are similar as well; those obtained by SMOTE and RUS are the two highest ones. Paradoxically, it is the combination of these two the one that has obtained the lowest variance.

In Figure II.4 it can be seen the effect of modifying the MV ratio. The median value is higher with a 50% Missing Values ratio. On this same boxplot it is seen as the third and the first quartile are considerably higher than on the rest, especially in the case of the third quartile. The general trend is that at a higher percentage of Missing Values in the data set both the first, the third quartile and the median increase. However, in the case of 10% of Missing Values the third quartile and the median reach values similar to 0% and on the other hand the first quartile gets a slightly smaller value.

A continuity in the values obtained in the different data sources is observed in Figure II.5, where the most different one from the other two is Counters. In the case of this data source, the variance is higher for the majority of anomalies. Additionally, the case of A6 anomaly is worth mentioning as the variance is drastically reduced for the Counters when compared to the other two data sources (Features and Features + Counters). After a thorough analysis, it has been identified the results causing this phenomenon, that have been obtained with the 25% MV ratio (note that no results are available for this anomaly when applying the 50% MV ratio). It can be seen in the Table II.5 that there is a big difference in the number of features between the 10% and 25% MV ratios in the case of the A6 anomaly for the Features and Features + Counters data sources.

Results are also discussed per anomaly, summarizing all the figures for the different MV ratios and balancing methods (boxplot d) in Figure II.5). Very concentrated AUC values (comprising few “abnormal” ones) have been obtained

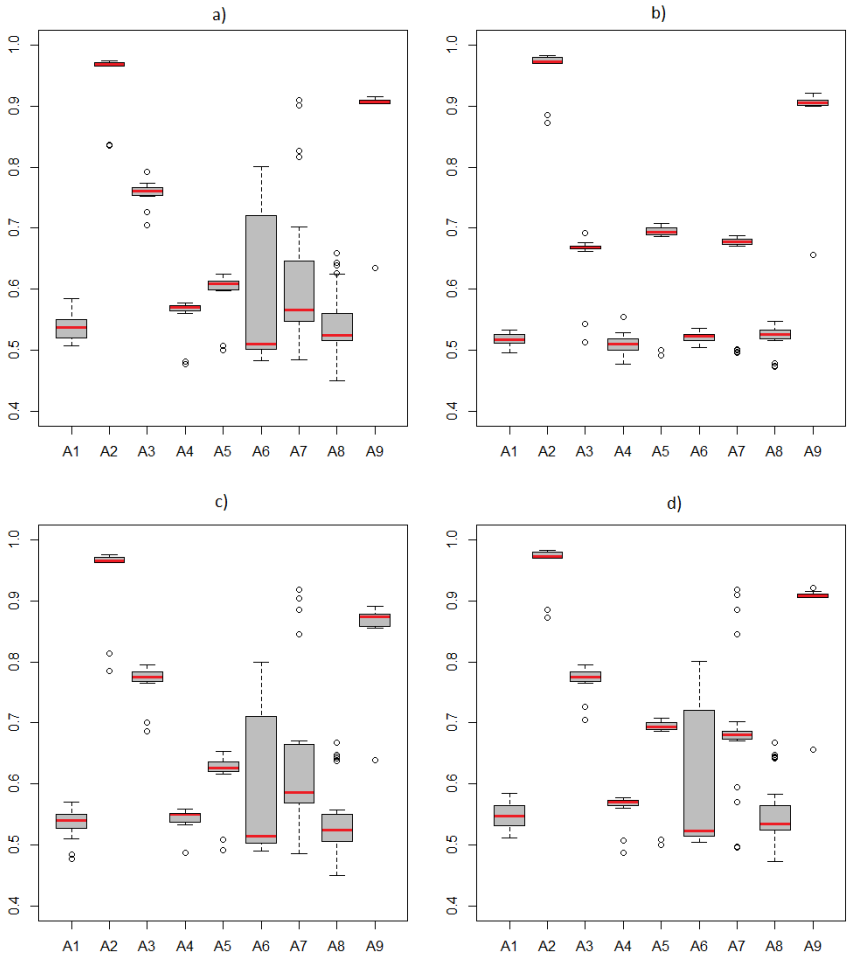

Figure II.5: Boxplot of the obtained AUC values per anomaly in the all-trials experiments in each data source. a) Features, b) Counters, c) Features + Counters, and d) All data sources.

for the A2, A3, A4, A5, A7, and A9 anomalies. Results associated to the A1 and A8 anomalies have a greater variance and lower median. The A6 anomaly is the one with the lowest median, very distant from the third quartile and close to the first quartile. This is mainly due to those results that has been previously highlighted from the last section of Table II.10: with a 50% MV ratio, no result can be obtained given the absence of instances of the anomalous class.

When analyzing results obtained from the different data sources, the following ideas can be observed:

- Features: the most prominent algorithm is RUS (best one for 4 of the

anomalies) followed by ROS and SMOTE (for 2 anomalies), whereas SMOTE + RUS ha attained the best results for 1 anomaly. With regard to the MV ratio, it should be noted that the same results for the A2 and A3 anomalies have been obtained with 50% and 25% values as the size of the dataset persists (see Table II.5). The 50% MV ratio is the one associated to most best results (5 out of 9 anomalies).

- Counters: the method of balancing with best results has been ROS (for 4 anomalies), followed by RUS (for 3 anomalies) and SMOTE (for 2 anomalies). There were 3 anomalies attaining the same best results with 2 different MV ratios: A3 with 0% and 10%, A5 with 25% and 50%, and A9 with 10% and 25%. As previously mentioned in the case of Features, the datasets are the same in all cases. What is different in this case is that the 0% MV ratio is associated to best results for 6 of the anomalies.

- Features + Counters: when combining the two data sources, the methods obtaining best results are RUS and SMOTE (for 4 anomalies each) while the combination of them (SMOTE + RUS) is the best one for an anomaly. In this case, a MV ratio is not clearly the best one for any anomaly: 50% for 3 anomalies, 25% for 2 anomalies, 10% for 3 anomalies, and 0% for 1 anomaly.

To sum up, a brief summary of the individual results is presented: the SMOTE algorithm has outperformed all the other ones for 4 anomalies, RUS for 2 and ROS for 2 as well. It is worth highlighting the fact that none of the hybrid balancing techniques have achieved the best AUC result for any anomaly. When taking into account the MV ratio, the 0% value is associated to the best results in 1 occasion, 10% and 25% in 3 occasions and 50% in 2 occasions. Finally, each one of the data sources is associated to the best results for 3 of the anomalies. On the other hand, the worst results (from those presented in previous tables) have always been associated to the A8 anomaly, being the lowest one that with a 0.5 MV ratio and Features + Counters as data sources.

II.5 Conclusions and Future Work

In present paper, different alternatives for data preprocessing (management of MV and class-balancing methods) have been validated for the detection of anomalies in a component-based robotic system. Obtained results when training and testing the same learning model (SVM) are presented and compared in section II.4 to validate the effect of pre-processing. All these figures have been obtained on a real-life and brand new dataset.

From the one-trial experiments it can be concluded that, as expected, the more balanced datasets are, the higher AUC values are obtained for the great majority of anomalies. However, in a real-life setting, anomalies are not frequent and unbalanced datasets are usual. Thus, experiments on more unbalanced and hence more real datasets have been also conducted. When analyzing the balancing methods by means of the statistical test, ROS and RUS, together

with its combination are the ones that outperforms the no-balancing alternative. However, none of them can be clearly identified as the best one.

From the all-trials experiments when considering the balancing method, it must be highlighted the results obtained with SMOTE (grey box in Figure II.4.a), which has stood out from the rest of methods. When individually analyzing the results (Table II.10), RUS is the balancing technique obtaining the highest AUC rates in most occasions (40%). Complementary, from the point of view of the the MV ratio, it can be concluded that the least restrictive value (a 50% ratio) means better AUC values in general terms (box in the right side of Figure II.4.b). As previously mentioned in section II.4.2 when discussing the results in Table II.4, the best results for the anomalies with a lower percentage of anomalous data (except the case of the A9) have been obtained with a high MV ratio (25% or 50%). For the other anomalies, best results have been obtained with the lowest ratios (0% and 10%).

As far as data sources are concerned, generally the best values are obtained with the combination of both (Features + Counters), whereas the highest median of values is the one associated to Counters. Results with Features greatly vary, depending on the MV ratio, while those with Counters are more constant.

All in all, it can be concluded that the proposed data-preprocessing techniques greatly contribute to increase the detection rate of anomalies, outperforming previous work [27]. However, the balancing method, MV ratio and data source must be carefully selected in each case as there is not a combination of them that outperforms the other ones in all cases, according to statistical tests.

Further work will be focused on covering all the anomalies and therefore analyzing those affecting more than one component. Additionally, due to the high number of features (columns) in the dataset, the application of feature selection techniques will also be explored. An alternative way of dealing with MV such as data imputation will also be considered, as well as benchmarking with additional learning models. The effect of the different strategies for handling data irregularities will also be considered for other supervised learning models in addition to SVM. Finally, some other alternatives for handling data imbalance (such as class weighting) will also be studied.

References

[1] Shrouf, F., Ordieres, J., and Miragliotta, G. “Smart factories in Industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm”. In: *2014 IEEE International Conference on Industrial Engineering and Engineering Management*. Dec. 2014, pp. 697–701.

[2] Muhuri, P. K., Shukla, A. K., and Abraham, A. “Industry 4.0: A bibliometric analysis and detailed overview”. In: *Engineering Applications of Artificial Intelligence* vol. 78 (2019), pp. 218–235.

[3] Aiman Kamarul Bahrin, M. et al. “Industry 4.0: A review on industrial automation and robotic”. In: *Jurnal Teknologi* vol. 78 (June 2016), pp. 137–143.

[4] IFR. *Summary - OUTLOOK on World Robotics Report 2019 by IFR*. en.

[5] Khalastchi, E. and Kalech, M. "On Fault Detection and Diagnosis in Robotic Systems". In: *ACM Comput. Surv.* vol. 51, no. 1 (Jan. 2018), pp. 1–24.

[6] Basurto, N. and Herrero, Á. "Data Selection to Improve Anomaly Detection in a Component-Based Robot". In: *14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019)*. Ed. by Martínez Álvarez, F. et al. Cham: Springer International Publishing, 2020, pp. 241–250.

[7] Wienke, J., Meyer zu Borgsen, S., and Wrede, S. "A Data Set for Fault Detection Research on Component-Based Robotic Systems". In: *Towards Autonomous Robotic Systems*. Ed. by Alboul, L., Damian, D., and Aitken, J. M. Vol. 9716. Cham: Springer International Publishing, 2016, pp. 339–350.

[8] Xu, X., Liu, H., and Yao, M. "Recent Progress of Anomaly Detection". In: *Complexity* vol. 2019 (2019).

[9] Ranshous, S. et al. "Anomaly detection in dynamic networks: a survey". In: *Wiley Interdisciplinary Reviews: Computational Statistics* vol. 7, no. 3 (2015), pp. 223–247.

[10] Jove, E. et al. "A fault detection system based on unsupervised techniques for industrial control loops". In: *Expert Systems* vol. 0, no. 0 (2019), e12395.

[11] Herrero, Á. and Jiménez, A. "Improving the Management of Industrial and Environmental Enterprises by means of Soft Computing". In: *Cybernetics and Systems* vol. 50, no. 1 (2019), pp. 1–2.

[12] Malhotra, R. "A systematic review of machine learning techniques for software fault prediction". In: *Applied Soft Computing* vol. 27 (2015), pp. 504–518.

[13] Banerjee, T. P. and Das, S. "Multi-sensor data fusion using support vector machine for motor fault detection". In: *Information Sciences* vol. 217 (2012), pp. 96–107.

[14] Zidi, S., Moulahi, T., and Alaya, B. "Fault Detection in Wireless Sensor Networks Through SVM Classifier". In: *IEEE Sensors Journal* vol. 18, no. 1 (Jan. 2018), pp. 340–347.

[15] Das, S., Datta, S., and Chaudhuri, B. B. "Handling data irregularities in classification: Foundations, trends, and future challenges". In: *Pattern Recognition* vol. 81 (2018), pp. 674–693.

[16] García-Laencina, P. J., Sancho-Gómez, J.-L., and Figueiras-Vidal, A. R. "Pattern classification with missing data: a review". In: *Neural Computing and Applications* vol. 19, no. 2 (Mar. 2010), pp. 263–282.

[17] López, V. et al. "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics". In: *Information Sciences* vol. 250 (2013), pp. 113–141.

[18] Twala, B. “Robot execution failure prediction using incomplete data”. In: *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. Dec. 2009, pp. 1518–1523.

[19] Fernández, A. et al. “Data Level Preprocessing Methods”. In: *Learning from Imbalanced Data Sets*. Cham: Springer International Publishing, 2018. Chap. 5, pp. 79–221.

[20] He, H. and Garcia, E. A. “Learning from Imbalanced Data”. In: *IEEE Transactions on Knowledge and Data Engineering* vol. 21, no. 9 (Sept. 2009), pp. 1263–1284.

[21] Cerqueira, V. et al. “Combining Boosted Trees with Metafeature Engineering for Predictive Maintenance”. In: *Advances in Intelligent Data Analysis XV*. Ed. by Boström, H. et al. Cham: Springer International Publishing, 2016, pp. 393–397.

[22] Syafrudin, M. et al. “An Affordable Fast Early Warning System for Edge Computing in Assembly Line”. In: *Applied Sciences* vol. 9, no. 1 (2018), pp. 84–102.

[23] Bergmeir, P. et al. “Classifying component failures of a hybrid electric vehicle fleet based on load spectrum data”. In: *Neural Computing and Applications* vol. 27, no. 8 (Nov. 2016), pp. 2289–2304.

[24] Luo, M. et al. “Using Imbalanced Triangle Synthetic Data for Machine Learning Anomaly Detection”. In: *Computers, Materials & Continua* vol. 58, no. 1 (2019), pp. 15–26.

[25] Devi, D., Biswas, S. K., and Purkayastha, B. “Learning in presence of class imbalance and class overlapping by using one-class SVM and undersampling technique”. In: *Connection Science* vol. 31, no. 2 (2019), pp. 105–142.

[26] Wienke, J. and Wrede, S. *A Fault Detection Data Set for Performance Bugs in Component-Based Robotic Systems*.

[27] Wienke, J. and Wrede, S. “Autonomous fault detection for performance bugs in component-based robotic systems”. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 3291–3297.

[28] Wienke, J. “Framework-level resouce awareness in robotics and intelligent systems”. PhD dissertation. Bielefeld University, 2018.

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

[30] Boser, B. E., Guyon, I. M., and Vapnik, V. N. “A Training Algorithm for Optimal Margin Classifiers”. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT ’92. Pittsburgh, Pennsylvania, USA: ACM, 1992, pp. 144–152.

[31] Chawla, N. V. et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* vol. 16 (2002), pp. 321–357.

[32] Jin Huang and Ling, C. X. “Using AUC and accuracy in evaluating learning algorithms”. In: *IEEE Transactions on Knowledge and Data Engineering* vol. 17, no. 3 (Mar. 2005), pp. 299–310.

[33] Mullick, S. S. et al. “Appropriateness of performance indices for imbalanced data classification: An analysis”. In: *Pattern Recognition* vol. 102 (June 2020), p. 107197.

[34] Wienke, J. and Wrede, S. “A middleware for collaborative research in experimental robotics”. In: *2011 IEEE/SICE International Symposium on System Integration (SII)*. Dec. 2011, pp. 1183–1190.

[35] Gehan, E. A. “A generalized Wilcoxon test for comparing arbitrarily singly-censored samples*”. In: *Biometrika* vol. 52, no. 1-2 (June 1965), pp. 203–224.

[36] Santafe, G., Inza, I., and Lozano, J. A. “Dealing with the evaluation of supervised classification algorithms”. In: *Artificial Intelligence Review* vol. 44 (June 1965), pp. 467–508.

Paper III

Imputation of Missing Values Affecting the Software Performance of Component-based Robots

Nuño Basurto, Ángel Arroyo, Carlos Cambra, Álvaro Herrero

Published in *Computers & Electrical Engineering*, October 2020, volume 87.
DOI: 10.1016/j.compeleceng.2020.106766.

Abstract

Intelligent robots are foreseen as a technology that would be soon present in most public and private environments. In order to increase the trust of humans, robotic systems must be reliable while both response and down times are minimized. In keeping with this idea, present paper proposes the application of machine learning (regression models more precisely) to preprocess data in order to improve the detection of failures. Such failures deeply affect the performance of the software components embedded in human-interacting robots. To address one of the most common problems of real-life datasets (missing values), some traditional (such as linear regression) as well as innovative (decision tree and neural network) models are applied. The aim is to impute missing values with minimum error in order to improve the quality of data and consequently maximize the failure-detection rate. Experiments are run on a public and up-to-date dataset and the obtained results support the viability of the proposed models.

III.1 Introduction

Wide attention has been devoted to the development of intelligent robots in recent years. Although significant contributions have been done, it still is a challenging field where further progress is required to satisfy present and future demands. One of such demands is the fluent interaction with non-expert humans, that is required for robotic systems to be widely integrated in a variety of homes and workplaces [1]. In order to get such fluency, performance of both hardware and software is a keystone. However, the ever-increasing complexity of robots leads to a parallel increase in chances of experiencing a failure. Accurate and

prompt detection of such events is required in order to improve performance and hence fluency. Full attention has been payed to advance in many subfields of the robotics arena but according to some authors [2], further effort must be devoted to anomaly detection in such systems. It is even more challenging when failures happen in a real-world context where complex phenomenon may interfere.

Accordingly, present paper focuses on the preprocessing of robot-performance data, whose importance is widely acknowledged. More precisely, the aim is the successful imputation of Missing Values (MV) in order to get as much data as possible for subsequent anomaly/failure detection. Thus, a wide variety of Artificial Intelligence (AI) models are applied in order to predict the MV of all the dataset components.

The successful detection of anomalies/faults is a challenging task that does not only apply to robots [3] [4] [5]. From a business perspective [6], AI in general, and Machine Learning (ML) in particular, can greatly contribute to anomaly detection and some other interesting tasks, maximizing companies benefits.

Since pioneer works [7] in the application of ML to robotics, unsupervised [8], supervised, and reinforcement [9] learning models have been previously applied. A variety of problems have been addressed so far such as control [10] [11] and communications [12] among others. In the case of anomaly detection, most ML previous work has been focused on the detection of hardware anomalies [13], while software anomalies have been scarcely investigated. Software failures often occur in robotic systems and their automatic detection requires training data. The problem comes from the difficulty of obtaining the data either because of the lack of execution traces or because the existing registers do not refer to the exact moment in which anomalies are produced. That is why it is difficult to find a dataset generated in a controlled environment where all the information is available. Furthermore, when data are gathered in a real-life environment, quite likely there will be some or many MV, that can not be processed by ML models.

One of the few works on the detection of software anomalies within the framework of component-based robots is [14]. In that paper, authors proposed the only publicly-available dataset (further details in section III.3) that gathers data from different performance indicators of a robot. The dataset [15] has been used in present paper as a benchmark dataset due to its interest and novelty. In this dataset, there are many MV associated to different data so a robust strategy must be followed in order to deal with them as most ML models can not process such data. One of the obvious preprocessing alternatives in order to solve such problem in the data is removing MV, either by deleting data instances or by deleting attributes. However, a more advanced proposal is to impute such values, keeping some information that could be useful for the subsequent anomaly detection. This approach is adopted in the present paper.

AI methods have been previously applied for imputation of MV [16]. However, scant attention has been devoted to the application of ML methods in order to solve such problem in robot datasets. One of the very few previous proposals is [17], where a probabilistic approach for classification using incomplete data was applied. The author performed a classification (for failure detection) of data samples by calculating the a priori probability of MV, determined from the data samples that are not missing. However, the author proposal was only applied to

outdated (1999) datasets containing hardware anomalies. Differentiating from previous work, the present paper is the first approach to impute MV in a dataset containing information about the performance of the software components of a robot. A comprehensive benchmark comprising a wide variety of methods has been performed and some of the methods are applied to this problem for the first time.

The methods applied for imputation of MV are introduced in section 2, while the analysed case study is described in section 3. The performed experiments, together with their associated results are compiled in section 4. Finally, the main conclusions and some proposals for further work are presented in section 5.

III.2 Imputation Methods

As previously stated, ML methods are applied in present study for imputation of MV. More precisely, experiments have been run with four regression techniques and two Artificial Neural Network (ANN) models with different training algorithms. The applied techniques are described in the following subsections.

III.2.1 Regression Techniques

Regression tries to model the relationship between two variables in the dataset by fitting a linear equation to the input data. One of the variables is the predictor variable and the other one is considered to be the criterion variable [18]. The general purpose of multiple regressions is to learn more about the relationship between several independent or predictor variables and a dependent or criterion variable. Such relationships can be linear or non-linear, leading to the two techniques that are described below.

III.2.1.1 Linear Regression

Linear Regression (LR) attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to the dataset [19]. Every value of the predictor variable (x) is associated with a value of the criterion variable (y). The regression line for p explanatory variables (x_1, x_2, ..., x_n) is defined as follows:

$$U_y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{III.1}$$

This line describes how the mean response U_y changes with the explanatory variables. The observed values for Y vary about their means U_y and are assumed to have the same standard deviation σ. The fitted values b_0, b_1, ..., b_n estimate the parameters β_0, β_1, ...,β_p of the population regression line. Since the observed values for y vary about their means U_y, the multiple regression models include a term for this variation. The model is expressed as DATA = FIT + RESIDUAL, where the "FIT" term represents the expression $\beta_0+\beta_1 x_1+\beta_2 x_2+ ... +\beta_n x_n$. The "RESIDUAL" term represents the deviations of the observed values (y) from their means U_y, which are normally distributed with mean 0 and variance σ.

The notation for the model deviations is ϵ. The model for linear regression, given n rows, is [19]:

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_n x_{in} + \epsilon_i \quad \text{for i=1, 2, ..., n} \tag{III.2}$$

III.2.1.2 Non-Linear Regression

Non-Linear Regression (N-LR) is a form of regression in which observational data are modeled by a function which is a non-linear combination of the input data and depends on one or more criterion variables . The parameters can take the form of an exponential, trigonometric, power, or any type of non-linear function. To determine the non-linear parameter values, an iterative algorithm is used. The model can be defined as:

$$y = f(X, \beta) + \varepsilon \tag{III.3}$$

Where B represents the non-linear parameter estimates to be computed, X is the dependent variables and ϵ represents the error terms.

III.2.1.3 Regression Trees

Regression Trees (RT) are usually shown growing upside down, with its root at the top. An observation passes down the tree through a series of splits (nodes). At them, a decision is made as to which direction (branch) to lead based on the value of one of the criterion variables. When a terminal node (leaf) is reached, a predicted response is given according to the end node. Trees are often built via binary recursive partitioning. In RT, values at the terminal nodes are assigned using the mean of cases in that node [20]. In present study, two variations of RT are applied:

- Fine Tree (FT). It usually comprises a reduced set of leaf nodes to optimize building time.
- The Boosting Ensemble (BE) algorithm iterative calls the regression-tree algorithm to construct an ensemble of trees. This ensemble combines results from many weak learners (least-squares boosting) with RT learners into one high-quality ensemble model. The response is calculated according to the Mean Squared Error (MSE) of the trained regression ensemble model that takes into account the results of boosting a high number (100) of trees.

III.2.2 Artificial Neural Networks

Artificial Neural Networks, also known as connectionist systems or adaptive networks, are simplified models of natural neural systems. The following definition, given by Hecht - Nielsen in 1988, formalizes the concept: *"An ANN is a parallel processing computer system distributed, consisting of a set of elementary processing units equipped with a small local memory and interconnected in a*

network through connections with associated weights. Each processing unit has one or more input connections and a single output connection that links to many collateral connections as desired. All processing associated with an elementary unit is a local, i.e. depends only on the values that take input signals from the unit and the internal state of the same". Two different models, adjusted to such definition, have been applied in the present study and are defined in the following subsections.

III.2.2.1 Multilayer Perceptron

The Multilayer Perceptron (MLP) consists of a system of simple interconnected neurons or nodes. They are connected by weights and output signals which are a function of the sum of the inputs to the node modified by a simple activation function. The architecture consists of several layers of neurons; the input layer serves to pass the input vector to the network. The terms called as "input vectors" and "output vectors" refer to the inputs and outputs of the MLP and can be represented as single vectors. The MLP may own al least one or more hidden layers and finally an output layer. MLP are fully connected, with each node connected to every node in the next and previous layer. One of the critical issues of such model is the training (update) of all the weights as the error can be calculated in the output but weights in all layers must be updated. To solve such problem, backpropagation was proposed and several different algorithms implement it. Due to their known advantages, two of them have been applied in present study:

- Levenberg-Marquardt (LM) [21]. It is derived from the Newton's technique that is designed for minimizing functions that are sums of squares of non-linear functions.
- Bayesian Regularization (BR) [22]. It aims to improve the model's generalization capability, expanding the objective function with the addition of the sum of squares of the network weights.

III.2.2.2 Radial-Basis-Function Networks

In a Radial-Basis-Function Network (RBFN), each neuron in the hidden layer has its own centroid, and for each input vector $x = (x_1, x_2, \ldots, x_n)$, it computes the distance between x and the centroid. As a result, the output of these neurons is calculated as a non-linear function of this distance. Assuming that there are r input nodes and m output nodes, the overall response function without considering non-linearity in an output node has the following form:

$$\sum_{i=1}^{M} W_i * K\left(\frac{x - z_i}{\sigma_i}\right) = \sum_{i=1}^{M} W_i * g\left(\frac{||x - z_i||}{\sigma_i}\right) \tag{III.4}$$

where x is an input vector, $M \epsilon \mathbb{N}$ is the number of units in the hidden layer, $\mathrm{W}_i \ \epsilon \mathbb{R}^m$ is the vector of weights linking the ith hidden-layer unit to the output nodes, K is a radially symmetric kernel function of a unit in the hidden layer,

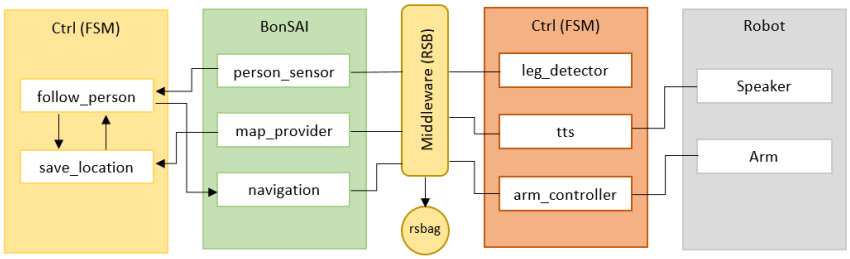

Figure III.1: Robot system architecture comprising analyzed modules. Adapted from [14].

z_i and σ_i, are the centroid and smoothing factor of the kernel node, and g: $[0, \infty) \rightarrow \mathbb{R}$ is the activation function, which characterizes the kernel shape.

III.3 Real-life Case Study

Present research focuses on the imputation of MV in order to optimize anomaly detection in robot software. As previously stated, researchers at the University of Bielefeld (Germany) developed the only publicly-available [15] dataset [14] containing software anomalies. The analyzed robot has different components from different manufacturers integrated in the GuiaBot platform, developed by Mobile Robots. This robot was developed to participate in the RoboCup@Home competition. RoboCup@Home aims to deploy technology to address future robot-service in domestic contexts. The obtained score in such contest depends on two main issues [23]: the degree of autonomy and the performance of the robot. In the dataset under analysis, the robot faced several problems similar to those addressed by a human waiter, such as the identification of customers, serving drinks, and interacting with people and objects. The diverse nature of such tasks requires the robot to have different components to be attached to the main platform. Examples of such components are a mechanical arm, a sensor to detect people from their legs, and a camera to recognize people.

The different software components of the robot communicate through the Robotics Service Bus (RSB) middleware [24], using an event-based system and whose information is stored in a tool that incorporates the middleware called rsbag. The transferred information is encrypted as a notification. There is a framework called BonSAI whose responsibility is the combination between the sensors (that receive external information) and the actuators. For the representation and control of the execution flows, a Finite State Machine (FSM) is used. The general architecture of the robot is depicted in Figure III.1, where it can be seen how the flow associated to the arm is different from that associated to the detection of legs because the second one does not lead to a physical action of the robot.

The two components analyzed in present research are the one for controlling

the arm (ArmController) and that for the detection of legs (LegDetector). The first one performs two actions: to control the movement of the arm in different directions and to open and close the grip. The induced anomaly (ArmServerAlgo) increases the amount of movements to carry out the target tasks and hence penalizes the execution time. As a consequence, the robot performs a series of unnecessary actions that have a negative effect on the performance counters. On the other hand, the LegDetector component is responsible for recognizing and detecting the legs of human beings in order to avoid colliding with them. The associated anomaly (LegDetectorSkippable) causes the robot to perform the scan of legs a greater number of times than needed. These components have been selected because, in the preliminary experiments carried out by the authors of the dataset [25], ArmController achieved the worst results while LegDetector achieved the best results when trying to detect the anomalies. Hence, the present study involves datasets that a priori have differences when performing classification tasks for failure detection. As it can be seen in Figure III.1, these components are of different nature since one involves a physical response in the robot, while the other does not.

The whole dataset comprises data from several trials, being each one of them an attempt of the robot to perform a target task. The authors who gathered the data [14], detected that in some of these trials there were undetected anomalies, that's the reason why some of them are discarded. The analyzed data for each one of the components come from the performance counters, measured every second. Further details about the attributes comprised in the component datasets can be found in Table III.1. This dataset is described in greater detail in [15].

Table III.1: Explanation of the dataset attributes.

Variables	Description
vsize	The current size of virtual memory used by a task.
open_fds	The current number of file descriptors opened by a process.
rchar	Number of bytes that a process has read since the beginning.
open_connections	Number of network connections opened by a process.
stime	Amount of time of the process in user mode.
wchar	Number of bytes that a process has write since the beginning.
utime	Amount of time of the process in kernel mode.
num_threads	Number of threads that a process has in operation.
rss	Current RSS of a process
received_bytes	Number of bytes received by the interface.
sent_bytes	Number of bytes sent by the interface.
write_bytes	Number of bytes written by the device.

The induction of anomalies is not constant in the whole dataset, as anomalies are induced at a varying rate in the different trials. Table III.2 shows the distribution of anomaly occurrences for the analyzed components. There are 10 trials associated to the ArmController and 12 to the LegDetector components.

All of the trials have been fused in present study to generate 2 different datasets, one per component.

Table III.2: Occurrences of each anomaly and distribution per trials.

Anomaly	1 time	2 times	3 times
ArmServerAlgo	28, 32, 36, 41, 45, 57, 65, 71	21	23
LegDetectorSkippable	19, 21, 24, 31, 36, 55, 57, 64, 66, 68, 70		71

There is a similar rate of MV in the two component datasets. In the ArmController one there are 26025 (8.5%) data instances containing missing values while there are 24163 (10.03%) in the case of the LegDetector.

III.4 Experiments and Results

The regression techniques described in Section III.2 have been applied to the datasets (ArmController and LegDetector) detailed in Section III.3, in order to evaluate their imputation capability on all the attributes of each dataset. To get more significant results, they are validated by the well-known n-fold Cross-Validation (CV) scheme. CV is a technique that splits the data, in order to measure the performance (MSE in the present study) of each technique in different subsets of data. The number (n) of data partitions has been set to 10 for all the experiments in the present study, as it is a standard value.

In order to do the regression on all the attributes of the 2 datasets, different variations have been generated for each dataset, one per each attribute. As a result, 11 variant datasets have been generated for the ArmController dataset and 8 for the LegDetector. In each case, the attribute on which the regression is applied is stated as the target column while the remaining ones are the predictor variables.

In the case of the ANN models, the training process has been repeated 10 times for each training algorithm (LM and BR for the MLP). The main purpose of this repetition is to reduce the effect of randomness and therefore obtaining more representative results. A sigmoid activation function in the hidden layer and a linear activation function in the output layer have been applied in the case of the MLP. A radial-basis transfer function was applied in the case of RBFN. Additionally, different parameters have been tested for some of the techniques. However, for the sake of brevity, only the results obtained for the following parameters are shown in Sections III.4.1 and III.4.2:

- FT: minimum leaf size (4).
- BE: minimum leaf size (8), and number of learners (30).
- RBFN: spread (40) and maximum number of neurons (10).

- MLP (both LM and BR): neurons in the hidden layer (10) and training epochs (70).

The average MSE and execution time (for the 10 folds) have been calculated for all the experiments and are shown in Tables III.3 to III.7. In the case of the MLP, the mean and standard deviation for the 10 executions are shown. MSE is calculated when trying to impute 25% of the data samples (considered as the MV) for each one of the CV folds, while 75% of the data samples are used to build/train the methods.

III.4.1 ArmController Component

Tables III.3 and III.4 show the results obtained for the ArmController component. More precisely, Table III.3 shows the average MSE obtained by each one of the regression techniques when predicting values for each one of the attributes in the original dataset. After analyzing results in this table, it is worth mentioning that figures significantly vary depending on the applied method and the target attribute. The N-LR method gets the best results (minimum error) for 9 out of the 11 components, while 3 other methods get the best result only in one case; FT for the open_fds component, RBFN for the num_threads component, and MLP with the BR training algorithm for the vsize component.

Table III.3: Average MSE value per method and dataset attribute on the ArmController component.

	LR	N-LR	FT	BE	RBFN	MLP_LM		MLP_BR	
						Mean	Std Dev	Mean	Std Dev
vsize	1.67E-06	1.64E-06	1.42E-05	3.12E-04	1.67E-06	3.49E-09	9.25E-26	2.89E-09	0
open_fds	2.80E-23	2.76E-23	3.52E-24	2.43E-12	2.79E-23	7.97E-23	1.97E-39	1.28E-22	5.26E-39
rchar	5.35E-19	5.28E-19	4.83E-10	5.67E-10	5.35E-19	1.50E-15	0	8.54E-16	2.20E-32
open_connections	2.80E-23	1.74E-28	1.31E-14	2.21E-12	2.79E-23	2.31E-22	2.63E-39	2.43E-22	0
stime	1.26E-17	1.26E-17	2.65E-09	2.78E-09	1.26E-17	3.32E-14	0	3.89E-14	0
wchar	6.63E-19	6.52E-19	3.55E-10	4.19E-10	6.63E-19	8.67E-16	0	1.15E-15	4.69E-32
utime	2.17E-14	2.16E-14	4.28E-08	4.97E-08	2.17E-14	6.31E-12	2.82E-30	2.01E-11	2.26E-29
num_threads	5.59E-22	1.43E-22	5.03E-23	1.23E-11	5.59E-22	2.99E-21	7.36E-38	5.66E-21	0
rss	7.44E-11	2.48E-11	1.93E-06	5.82E-06	7.44E-11	6.73E-10	8.67E-27	9.39E-10	0
received_bytes	1.67E-17	1.57E-17	9.95E-10	2.10E-09	1.67E-17	5.19E-14	5.29E-31	1.04E-13	8.82E-32
sent_bytes	5.13E-18	4.83E-18	6.38E-10	1.22E-09	5.13E-18	3.02E-14	1.76E-31	1.61E-14	7.61E-31

The differences in the MSE values are quite big, being the FT (except for the open_fds component) and BE methods the ones with the worst performance in terms of MSE.

When considering the attribute which obtain a lower MSE, the open_fds achieves the best results while the open_connectins is the second one. On the other hand, the vsize is the attribute that obtain the higher error for all the methods.

For the MLP neural model (2 training algorithms), the standard deviation of the MSE is really low (zero in eight times). It means that obtain results are robust and consistent for the 2 training algorithms (LM and BR).

Complementary to the above-shown errors, Table III.4 shows the average execution times for each method and attribute.

Table III.4: Average execution time per method and dataset attribute on the ArmController component.

	LR	N-LR	FT	BE	RBFN	MLP_LM		MLP_BR	
						Mean	Std Dev	Mean	Std Dev
vsize	8.13E-02	1.57E-01	1.44E-01	5.33E-01	2.00E-02	1.35E-01	1.20E-01	1.40E-01	1.17E-01
open_fds	3.08E-02	1.12E-01	2.08E-01	6.28E-01	3.34E-02	9.81E-02	2.57E-02	9.37E-02	1.30E-02
rchar	3.47E-02	1.75E-01	3.51E-01	1.03E+00	1.44E-02	9.68E-02	2.03E-02	9.60E-02	1.45E-02
open_connections	4.29E-02	1.74E-01	1.33E-01	5.48E-01	2.47E-02	1.04E-01	3.40E-02	9.90E-02	1.80E-02
stime	3.29E-02	1.45E-01	1.21E-01	5.57E-01	1.90E-02	9.76E-02	1.37E-02	1.06E-01	1.88E-02
wchar	3.33E-02	1.67E-01	1.66E-01	5.41E-01	3.88E-02	1.07E-01	2.20E-02	1.08E-01	2.42E-02
utime	3.65E-02	1.37E-01	1.34E-01	5.03E-01	1.32E-02	1.00E-01	1.23E-02	1.08E-01	1.91E-02
num_threads	3.43E-02	1.58E-01	9.89E-02	4.50E-01	1.45E-02	9.57E-02	1.23E-02	1.03E-01	1.52E-02
rss	3.41E-02	1.59E-01	1.87E-01	4.62E-01	1.29E-02	9.81E-02	1.63E-02	1.05E-01	1.17E-02
received_bytes	3.35E-02	1.39E-01	1.33E-01	5.36E-01	1.29E-02	9.78E-02	1.18E-02	1.11E-01	2.85E-02
sent_bytes	3.87E-02	1.49E-01	1.37E-01	5.83E-01	1.38E-02	1.06E-01	2.72E-02	9.61E-02	8.66E-03

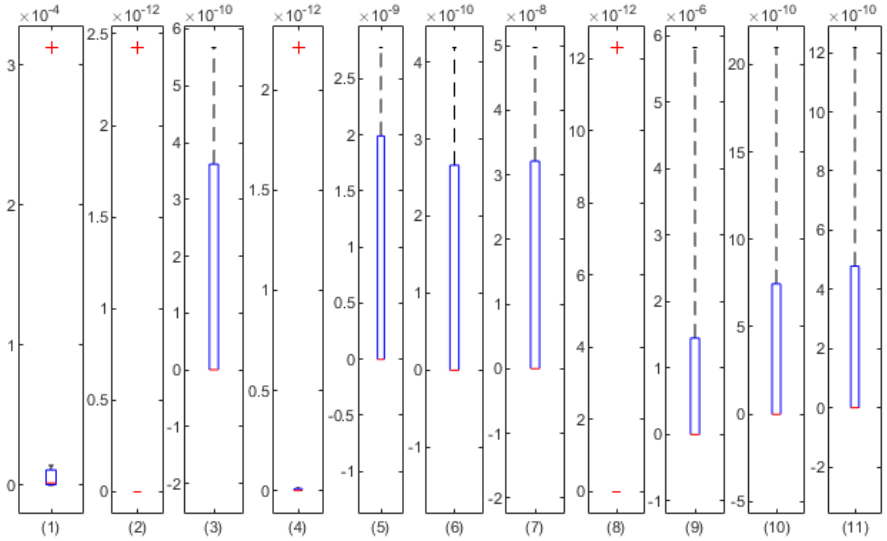

Figure III.2: Boxplot of the MSE values (all imputation methods) on the ArmController component per attribute. (1) vsize, (2) open_fds, (3) rchar, (4) open_connections, (5) stime, (6) wchar, (7) utime, (8) num_threads, (9) rss, (10) received_bytes, (11) sent_bytes

As far as execution times are concerned, it can be said that the results slightly vary. The RBFN neuronal model manages to be the fastest method for all the attributes. On the other hand, BE is the slowest method, as the highest execution times are obtained by this method. Analyzing the table by attributes, no big differences are appreciated in the time. However, the fastest results are obtained for the rss and received_bytes attributes.

In order to ease a visual analysis of these obtained results, Figure III.2 and Figure III.3 show the boxplots for the values that are summarized in Table III.3 and Table III.4 respectively. In Figure III.2, all the results obtained by all

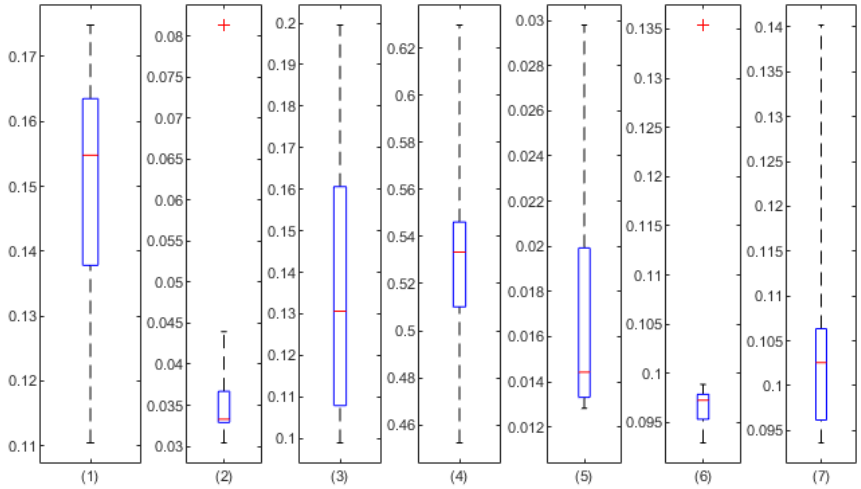

Figure III.3: Boxplot of the execution time (all components) on the ArmController component per method. (1) LR, (2) N-LR, (3) FT, (4) BE, (5) RBFN, (6) MLP_LM, (7) MLP_BR

the applied imputation methods are included for each one of the components. Similarly, in Figure III.3, all the results obtained on all the dataset components are included for each one of the imputation methods.

It is worth highlighting from Figure III.2 the great variability in the results when applying the different imputation methods to each one of the components. This is specially visible for the components open_fds (2), open_connections (4), and num_threads (8). For such components, the high error rates obtained by the BE method for the component open_fds and those obtained by the FT and BE methods for the other two components can be easily identified. The remaining error rates obtained for these components are shown in a small box, with no variance. Something similiar (to a certain extent) can be seen for the vsize (1) component. For the other components, error rates are pretty similar, slightly varying between the different methods.

As previously mentioned, it can be seen in Figure III.3 that there is a small variability in the execution times. In the case of the N-LR and MLP_LM methods, some outlier values can be observed, while the other values are very similar. All in all, average values are quite centered in the (25^{th} and 75^{th}) percentiles shown in the boxplot.

III.4.2 LegDetector Component

Similarly to what is shown above for the ArmController component (Subsection III.4.1), MSE and execution times are shown for the experiments run on the LedgeDetector component. Therefore, Table III.5 and Table III.6 shows

the MSE and execution times respectively obtained in the experiments on the LegDetector component.

Table III.5: Average MSE value per method and dataset attribute on the LegDetector component.

	LR	N-LR	FT	BE	RBFN	MLP_LM		MLP_BR	
						Mean	Std Dev	Mean	Std Dev
write_bytes	1.07E-14	1.07E-14	8.42E-08	9.28E-08	1.07E-14	3.68E-11	0	2.8099E-11	4.06E-28
rchar	3.71E-12	3.71E-12	1.68E-06	1.73E-06	3.71E-12	6.87E-09	0	5.5068E-09	0
stime	1.83E-14	1.83E-14	1.02E-07	1.17E-07	1.82E-14	6.20E-11	9.03E-28	1.9076E-11	4.06E-28
wchar	9.66E-17	9.66E-17	1.70E-09	2.29E-09	8.74E-17	5.25E-14	1.76E-31	6.1249E-14	0
utime	1.11E-12	1.11E-12	4.12E-07	5.01E-07	1.08E-12	2.45E-09	4.62E-26	5.0123E-09	1.39E-25
rss	1.79E-05	1.79E-05	2.44E-03	3.48E-03	1.79E-05	1.31E-05	7.19E-08	1.2559E-05	4.68E-07
received_bytes	5.98E-14	5.98E-14	7.72E-08	1.25E-07	5.98E-14	5.83E-12	0	1.5439E-11	2.71E-28
sent_bytes	1.58E-14	1.57E-14	1.82E-08	2.71E-08	1.42E-14	8.29E-13	0	1.0276E-12	1.16E-29

Table III.5 shows quite different results from those observed in Table III.3 for the ArmController component (Subsection III.4.1). In the case of the LegDetector component, the model that returns the best results (in terms of the MSE) is the RBFN neural model, for 7 out of 8 dataset attributes that have been imputed. For 3 of these attributes, namely write_bytes, rchar, and received_bytes, the LR and N-LR regression techniques have obtained similar error rates. The MLP with the BR training algorithm has obtained the best result for the rss component, while the FT and BE trees have performed poorly (similarly to what is shown in TableIII.3). As for the previous component, the standard deviation for the two MLP training algorithms (10 executions each) are really low; it amounts to zero in 6 out of the 16 cases that are shown. Analyzing the MSE per components (Table III.5), wchar is the one for which the lowest error rate has been obtained. Secondly, similar MSE values have been obtained for the write_bytes, stime, received_bytes and sent_bytes attributes. Oppositely, rss is by far the one with the highest MSE value.

Table III.6: Average execution time per method and dataset attribute on the LegDetector component.

	LR	N-LR	FT	BE	RBFN	MLP_LM		MLP_BR	
						Mean	Std Dev	Mean	Std Dev
write_bytes	7.15E-02	1.84E-01	5.36E-01	8.45E-01	1.43E-02	9.43E-02	1.36E-02	1.06E-01	2.43E-02
rchar	3.49E-02	9.30E-02	1.34E-01	9.68E-01	1.18E-02	9.53E-02	2.87E-02	9.93E-02	2.88E-02
stime	3.15E-02	1.33E-01	6.87E-01	7.49E-01	1.11E-02	5.52E-01	1.06E+00	8.84E-02	1.63E-02
wchar	2.98E-02	9.52E-02	5.22E-01	6.90E-01	1.09E-02	9.97E-02	3.29E-02	8.35E-02	9.87E-03
utime	2.91E-02	9.33E-02	7.95E-01	6.75E-01	1.66E-02	8.30E-02	1.08E-02	8.90E-02	1.14E-02
rss	3.09E-02	8.09E-02	7.40E-01	6.53E-01	1.08E-02	4.49E-01	7.53E-01	9.78E-01	6.57E-01
received_bytes	2.83E-02	8.23E-02	6.52E-01	6.04E-01	1.07E-02	8.18E-02	9.15E-03	8.69E-02	1.16E-02
sent_bytes	2.97E-02	8.49E-02	4.28E-01	6.14E-01	1.06E-02	7.98E-02	7.18E-03	8.77E-02	1.09E-02

As far as average execution times are concerned, the results shown in Table III.6 are very similar to those shown in Table III.4, the RBFN model

outperforms the other methods, being the fastest one in the calculation of the missing values for the 8 parameters that have been regressed. On the other hand, those methods based on regression trees (FT and BE) are the ones with highest execution times for all the components. Although more trees are generated in the case of the BE, their execution times are higher that those of the FT only for 5 of the attributes.

Additionally, it is noted that the execution times are somewhat smaller than those shown in Table III.4 because the the number of predictors (attributes) for this component (7) is smaller than those for the ArmController (10).

In order to ease a visual analysis of these obtained results, Figure III.4 and Figure III.5 show the boxplots for the values that are summarized in Table III.5 and Table III.6 respectively.

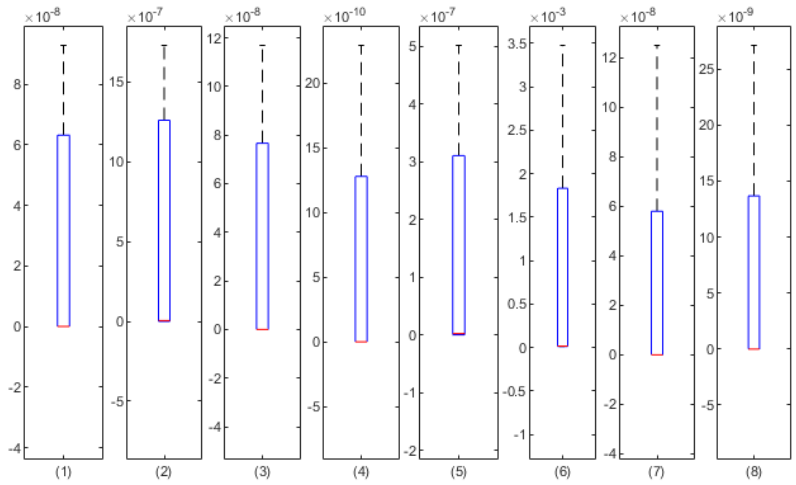

Figure III.4: Boxplot of the MSE values (all imputation methods) on the LegDetector component per attribute. (1) write_bytes, (2) rchar, (3) stime, (4) wchar, (5) utime, (6) rss, (7) received_bytes, (8) sent_bytes

Figure III.4 shows more compact results than those associated to the ArmController component (shown in Figure III.2). In the case of the LegDetector component, there is no outliers in any of the attributes and all the error rates for the 8 attributes are within the 25^{th} and 75^{th} percentiles. This is a fact that deserves attention, as it means that the imputation methods perform in a regular and smooth way for all the attributes.

On the other hand, when considering the boxplot of the execution time (Figure III.5), it can be said that similar results are shown to those for the ArmController component (shown in Figure III.3). Once again, in the case of N-LR and the MLP with the BR training algorithm, an outlier is identified outside the percentiles. For all the other methods it can be said that time results are similar to those of the previous component (ArmController).

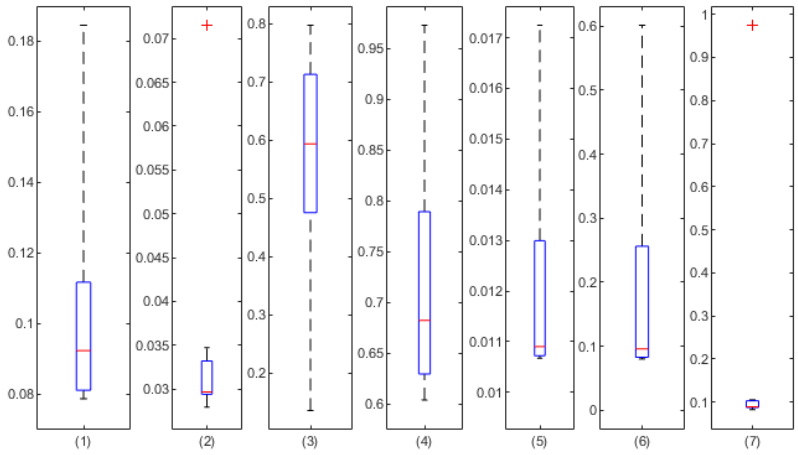

Figure III.5: Boxplot of the execution time (all components) on the LegDetector component per method.(1) LR, (2) N-LR, (3) FT, (4) BE, (5) RBFN, (6) MLP_LM, (7) MLP_BR

All in all, it can be said that for the two components that are analyzed in present research, error rates vary according to the regression method that is applied. As the aim of present work is to validate which one of these methods outperforms the other ones in order to impute missing values, MSE and time results (shown in Tables III.3 to III.6) are summarized in Table III.7.

Table III.7: Summary of the best-performing imputation method per component attribute in terms of both error and execution time.

Dataset	ArmController		LegDetector	
Attribute	MSE	Time	MSE	Time
vsize	MLP_BR	RBFN	-	-
open_fds	FT	RBFN	-	-
open_connections	N-LR	RBFN	-	-
num_threads	N-LR	RBFN	-	-
rchar	N-LR	RBFN	LR, N-LR, RBFN	RBFN
stime	LR, N-LR, RBFN	RBFN	RBFN	RBFN
wchar	N-LR	RBFN	RBFN	RBFN
utime	N-LR	RBFN	RBFN	RBFN
rss	N-LR	RBFN	MLP_BR	RBFN
received_bytes	N-LR	RBFN	LR, N-LR, RBFN	RBFN
sent_bytes	N-LR	RBFN	RBFN	RBFN
write_bytes	-	-	LR, N-LR, RBFN	RBFN

By means of Table III.7 it is possible to analyze at a glance the obtained results that are presented in this section. In nutshell, the methods that have obtained

the best results in terms of MSE are RBFN and N-LR. N-LR outperforms the other ones for most of the attributes in the ArmController component while the same happens with RBFN in the case of the LegDetector component. For 4 attributes LR, N-LR and RBFN have obtained similar error rates, that are the lowest ones. In addition to this general perspective, it can be seen that for some of the attributes, MLR_BR and FT are the best performing methods. This means that the selection of the regression method must be considered case by case and several methods must be applied in order to impute missing values with the minimum error. In terms of execution times, the RBFN model is the fastest one in all cases.

III.5 Conclusion

In the present study the imputation methods detailed in Section III.2 have been applied to the two datasets explained in Section III.3. These two datasets correspond to the ArmController and LegDetector components of the robot. After preparing the data and applying the CV scheme to obtain more reliable results, a regression has been performed on the 11 and 8 attributes of the dataset. From the obtained results (Section III.4) it can be concluded that:

- For the ArmController component (Section III.3), the N-LR method is the one that obtains the best results in terms of MSE and the neuronal model RBFN is the second best. The worst error rates are obtained by the FT and BE techniques. The attributes with lowest error rates have been open_fds and open_connections, so the imputation of missing values on them can be reliably performed. As for execution times, the RBFN method obtains the lowest times on all attributes.

- For the LegDetector component (Section III.5), the RBFN method is the one that obtains the best results in terms of MSE for the 8 attributes. In many cases, similar results are obtained by LR and N-LR. Additionally, RBFN is the fastest method on all attributes.

- It can be observed that there is no single technique that is best in all cases. Even on the same attribute in the 2 different datasets, the best results can be obtained with different techniques.

Taking into account all the above mentioned, it can be concluded that imputation of missing values can be successfully performed. One of the regression methods that are compared can be selected to impute values of each one of the attributes from the given components.

As a future line of work, imputation will be combined with some other data preprocessing techniques (such as data balancing algorithms) to improve anomaly detection.

References

[1] Hoffman, G. “Evaluating Fluency in Human–Robot Collaboration”. In: *IEEE Transactions on Human-Machine Systems* vol. 49, no. 3 (June 2019), pp. 209–218.

[2] Khalastchi, E. and Kalech, M. “On Fault Detection and Diagnosis in Robotic Systems”. In: *ACM Comput. Surv.* vol. 51, no. 1 (Jan. 2018), pp. 1–24.

[3] Xu, X., Liu, H., and Yao, M. “Recent Progress of Anomaly Detection”. In: *Complexity* vol. 2019 (2019).

[4] Ranshous, S. et al. “Anomaly detection in dynamic networks: a survey”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* vol. 7, no. 3 (2015), pp. 223–247.

[5] Jove, E. et al. “A fault detection system based on unsupervised techniques for industrial control loops”. In: *Expert Systems* vol. 0, no. 0 (2019), e12395.

[6] Herrero, Á. and Jiménez, A. “Improving the Management of Industrial and Environmental Enterprises by means of Soft Computing”. In: *Cybernetics and Systems* vol. 50, no. 1 (2019), pp. 1–2.

[7] Dorigo, M. and Schnepf, U. “Genetics-based machine learning and behavior-based robotics: a new synthesis”. In: *IEEE Transactions on Systems, Man, and Cybernetics* vol. 23, no. 1 (Jan. 1993), pp. 141–154.

[8] Jayaratne, M., de Silva, D., and Alahakoon, D. “Unsupervised Machine Learning Based Scalable Fusion for Active Perception”. In: *IEEE Transactions on Automation Science and Engineering* vol. 16, no. 4 (Oct. 2019), pp. 1653–1663.

[9] Kober, J., Bagnell, J. A., and Peters, J. “Reinforcement learning in robotics: A survey”. In: *The International Journal of Robotics Research* vol. 32, no. 11 (2013), pp. 1238–1274.

[10] Zhao, D., Ni, W., and Zhu, Q. “A framework of neural networks based consensus control for multiple robotic manipulators”. In: *Neurocomputing* vol. 140 (2014), pp. 8–18.

[11] Xiao, B. and Yin, S. “Exponential Tracking Control of Robotic Manipulators With Uncertain Dynamics and Kinematics”. In: *IEEE Transactions on Industrial Informatics* vol. 15, no. 2 (Feb. 2019), pp. 689–698.

[12] H. Alsamhi, s., Ma, O., and Ansari, M. S. “Survey on artificial intelligence based techniques for emerging robotic communication”. In: *Telecommunication Systems* vol. 72, no. 3 (Nov. 2019), pp. 483–503.

[13] Lu, H. et al. “Motor Anomaly Detection for Unmanned Aerial Vehicles Using Reinforcement Learning”. In: *IEEE Internet of Things Journal* vol. 5, no. 4 (Aug. 2018), pp. 2315–2322.

[14] Wienke, J., Meyer zu Borgsen, S., and Wrede, S. “A Data Set for Fault Detection Research on Component-Based Robotic Systems”. In: *Towards Autonomous Robotic Systems*. Ed. by Alboul, L., Damian, D., and Aitken, J. M. Vol. 9716. Cham: Springer International Publishing, 2016, pp. 339–350.

[15] Wienke, J. and Wrede, S. *A Fault Detection Data Set for Performance Bugs in Component-Based Robotic Systems*.

[16] Arroyo, A. et al. “Neural Models for Imputation of Missing Ozone Data in Air-Quality Datasets”. In: *Complexity* vol. 2018 (Mar. 2018).

[17] Twala, B. “Robot execution failure prediction using incomplete data”. In: *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. Dec. 2009, pp. 1518–1523.

[18] Yale, U. of. *Linear Regression*. 2017.

[19] Yale, U. of. *Multiple Linear Regression*. 2017.

[20] Moisen, G. G. *Classification and Regression Trees*. 2018.

[21] Lv, C. et al. “Levenberg–Marquardt Backpropagation Training of Multilayer Neural Networks for State Estimation of a Safety-Critical Cyber-Physical System”. In: *IEEE Transactions on Industrial Informatics* vol. 14, no. 8 (Aug. 2018), pp. 3436–3446.

[22] Doan, C. D. and Liong, S.-y. “Generalization for multilayer neural network bayesian regularization or early stopping”. In: *Proceedings of Asia Pacific Association of Hy drology and Water Resources 2nd Conference*. 2004, pp. 5–8.

[23] Jumel, F. “Advancing Research at the RoboCup@Home Competition [Competitions]”. In: *IEEE Robotics Automation Magazine* vol. 26, no. 2 (June 2019), pp. 7–9.

[24] Wienke, J. and Wrede, S. “A middleware for collaborative research in experimental robotics”. In: *2011 IEEE/SICE International Symposium on System Integration (SII)*. Dec. 2011, pp. 1183–1190.

[25] Wienke, J. and Wrede, S. “Autonomous fault detection for performance bugs in component-based robotic systems”. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 3291–3297.

Paper IV

A Hybrid Machine Learning System to Impute and Classify a Component-Based Robot

Nuño Basurto, Ángel Arroyo, Carlos Cambra, Álvaro Herrero

Accepted in *IGPL*, July 2021, volume In Press. DOI: Not available

Abstract

In the field of cybernetic systems and more specifically in robotics, one of the fundamental objectives is the detection of anomalies in order to minimize loss of time. Following this idea, this paper proposes the implementation of a Hybrid Intelligent System in four steps to impute the missing values, by combining clustering and regression techniques, followed by balancing and classification tasks. This system applies regressions models to each one of the clusters built on the instances of data set. Subsequently, a variety of balancing techniques are applied to improve the classifier's ability to discern whether it is in an error or a normal state. These techniques support to obtain better classification ratios in which a robot is close to error and allow us to bring the behavior back to a normal state. The experimentation is performed using a modern and public data set, which has been extracted from a component-based robotic system, in which different anomalies are induced by software in their components.

IV.1 Introduction

In recent years, the production systems have been including more and more robotic systems in production lines to automate processes and improve efficiency in productivity terms. Robotics has been expanding in a variety of ways, such as quality control, assembly or loading and unloading [1]. The robotic systems cover different disciplines like cinematic, mechatronics, electronics, and Artificial Intelligence (AI). Regarding the latter, the European Commission recently reported that 42% of enterprises use at least one technology related to this field [1]. This percentage will increase in the coming years, showing a clear need for companies to adopt these technologies, which are already consolidated in the

[1]European Commission. European enterprise survey on the use of technologies based on artificial intelligence (July 2020). URL: https://ec.europa.eu/digital-single-market/en/news/european-enterprise-survey-use-technologies-based-artificial-intelligence

market. However, everything depends on adapting current systems and hiring qualified personnel to carry out this work.

In the field of robotics, Machine Learning (ML) algorithms can be applied to detect errors in executions, in order to reduce the impact of these errors and return to normal production. To prevent them and reduce downtime, anomaly detection [1] is required, although not enough efforts are devoted to it from the scientific community so far [2].

The difficulty of processing real data that has been generated from sensors to detect anomalies is a challenge due to the presence of Missing Values (MV) [3], that must be overcame [4]. The need to impute values is enormous in order to minimize the loss of information when working with a robotic data set [5]. The relevance of having a complete data set to work with in order to obtain the best possible results is evident. In the case of not being able to impute the MV, it would be necessary to work with a less complete data set in terms of instances or attributes. In order to address this problem, a novel Hybrid Artificial Intelligence System (HAIS) is proposed.

The paper is distributed as follows: the previous work is discussed in IV.2 while the novel HAIS is described in Section IV.3. The component-based robot dataset is described in Section IV.4 and the experimental results are presented in Section IV.5. Finally, Section IV.6 deals with the conclusions reached from this research and future lines of work to be pursued.

IV.2 State of the art

To solve the MV problem, very different techniques have been applied up to now [5]. These Imputation Methods (IM) are classified as single imputation, where the method fills in one value for each missing one and multiple imputation where multiple values are tried at the same time. In this work, single imputation techniques have been applied to make the resulting imputation data set more easily usable for classification purposes. To achieve the best possible MV imputation a nonlinear regression technique together with an Artificial Neural Network (ANN), more precisely the Radial Basis Function Network (RBFN) taking advantage of their regression capability [6], are applied to fulfill the data set.

Other approaches that have been adopted concerning the presence of MV are carried out by Cerqueira et al. [7], who are committed to the elimination of MV. Likewise, these authors deal with the use of balancing techniques such as Synthetic Minority Oversampling Technique (SMOTE) to achieve a balanced distribution between classes. Following this line of use of balancing techniques, Syafrudin et al. [8] also relies on the use of SMOTE, in this case, the target is to detect possible anomalies in an assembly line. The approach used is to differentiate between two classes (one-class classification), one being normal and the other abnormal. Another recent study that combines balancing techniques with a one-class approach is the work related on Debarshhree et al. [9], where the authors investigated the impact that unbalancing can have on a variety of data sets. To test the effectiveness of the applied balancing techniques they

use a variety of classifiers such as Extreme Learning Machines, Naive Bayes, or Support Vector Machines (SVM). It was possible to improve the trends of the minority class for the majority class employing SVM.

In the case of the anomaly dataset that is analysed in the present paper [10], scant attention has been devoted to it by the research community, mainly due to its novelty. One of the very few papers addressing this topic is . [11]. The auhtors (those of the benchmark dataset itself) developed an error detection model, where they compare their own generated model by relying on a One-Class-SVM classifier, they act on the total set of anomalies, (detailed in Section IV.4). Their model obtained improved results in the vast majority of components, but not in the one analyzed in the present study. In a sequel paper [12], the authors discuss how to carry out individual performance tests on the different components of a robotic system, analyzing possible changes in the different revisions. Subsequently, Wienke et al. [13] further explored the idea of analyzing resource utilization in the different components, proposing a model that performs tests aimed at detecting regressions in resource utilization. The claims of this new model is to reduce the complexity in the creation of performance tests. Finally, Wienke published his thesis [14] in which he applied the methods seen in his earlier papers and extended them by emphasizing his component-based robotic data set and resource utilization improvement techniques.

The problem associated in this paper deals with the imputation of values through the combined use of regression and clustering techniques. Failure detection has already been discussed previously on this dataset [15]. However, the previous work on this data proposed the elimination of MV [16] rather than their imputation. The problem addressed in this paper have already been dealt with in less depth [17], where the work focused only on cluster regression but without performing classification tests on the data resulting from the imputation. Likewise, the study was conducted only with the k-means clustering technique and in present work it has been extended with additional techniques.

IV.3 Hybrid Intelligent System

The main goal of the hybrid system presented in this research is to establish a new type of MV imputation through the data set clustering, where the columns with MV have been previously extracted. Once the clusters have been obtained, within each one the columns with MV added again, the regression is applied on them to perform the imputation of the values later on. The Figure IV.1 shows the steps of each of the stages of this hybrid System.

The hybrid ML solution proposed in the present work is divided into the following four steps:

1. Clustering: three clustering techniques are applied to the data set to obtain more homogeneous groups of data.

2. MV Imputation: two regression techniques are applied to each of the clusters generated in the previous step to impute the MV, therefore all the single attributes are valued.

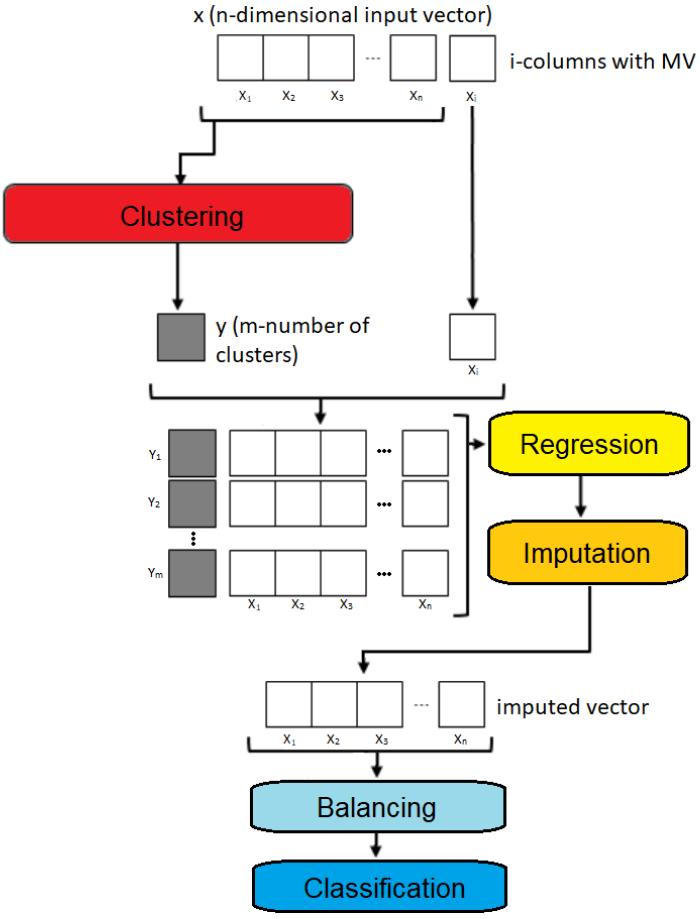

Figure IV.1: Hybrid system novel formulation.

3. Data Balancing: a series of balancing techniques are applied to already imputed data set to achieve a greater balance between the minority or anomalous class and the majority or normal class.

4. Classification: given the large number of sets obtained from the previous steps developed, the well-known Support Vector Machine (SVM) [18] classifier is used. For a clear interpretation of the results obtained by this process, several metrics are used to compare the results.

For the selection of different techniques in each of the steps of the present hybrid proposal, techniques previously applied to the data set have been used. For the selection of clustering techniques based on [17], only k-means and clustering techniques were used, although as detailed in the future work, the inclusion of hierarchical and density-based methods could be interesting. Secondly, for imputation techniques in [19] was concluded that, in general, the regression

techniques that worked best were those used in the present investigation. Finally, for balancing techniques in the research carried out [16], all the techniques presented here were used with the exception of Borderline-SMOTE (BLSMOTE), whose adhesion is due to the fact that it is derived from SMOTE, a technique that stood out from the rest and could be of interest.

IV.3.1 Clustering

IV.3.1.1 K-means.

Cluster analysis [20] organizes data by grouping data samples according to a criterion distance. Two individuals in a valid group will be much more similar than those in distinct groups. The k-means clustering algorithm [21] groups data samples into a predefined number of groups. It requires two input parameters: the number of clusters (k) and their initial centroids. Initially, each data sample is assigned to the cluster with the closest centroid. Once the clusters are defined, the centroids are recalculated and samples are reassigned. These steps are repeated until there is no further change in the centroids. The quality criterion to measure the grouping is the Sum of Squared Errors (SSE). The algorithm to minimize it can be defined as follows:

$$SSE = \sum_{j=1}^{k} \sum_{x \epsilon G_i} p(x_i, c_j)/n \tag{IV.1}$$

where, k is the number of groups, p is the proximity function, c_j is the centroid of group j, and n is the number of samples. Different measures of distance have been tested to obtain the best results, with the Euclidean distance being the chosen one. In this distance, each centroid is the mean of the points in its cluster. Is defined as:

$$d_{st}^2 = (x_s - y_t)(x_s - y_t)' \tag{IV.2}$$

where d is the distance from point x to centroid c. In the run experiments, the Means++ algorithm has been used for the initialization of centroids.

IV.3.1.2 Hierarchical.

Hierarchical clustering algorithms are top-down or bottom-up implementations. Bottom-up approaches treat each sample as a single cluster at the beginning, and then successively merge pairs of clusters until it merges all clusters into a single cluster containing all the samples. The bottom-up Hierarchical clustering, also called Hierarchical Agglomerative Cluster (HAC), generates a cluster tree or dendrogram by using heuristic techniques. A dendrogram [22] comprises many U-shaped lines connecting data points in a Hierarchical tree. The height of each U represents the distance between the two connected data points. The most popular algorithms that use merging to create the cluster tree are called agglomerative. There are many implementations of HAC [23]. Similar to the IV.3.1.1 Section, the Euclidean distance (equation IV.2) is chosen.

IV.3.1.3 Density-based Spatial Clustering of Applications with Noise (DBSCAN).

This clustering algorithm is based on density, i.e. it analyzes regions whose points have a higher density separated by others with a lower density [24]. In DBSCAN, each point sets a radius around itself, counting the number of points that fall within it. A minimum number of points that must be within this radius is established to know if they are part of the same group as the initial point. This algorithm does not follow centrality hypotheses as in the case of k-means, but produces complex groups. There are three types of points:

- Core points: the points that are in the interior of a group near the center.
- Border points: those located at the edge of the radius.
- Noise points: those located neither one nor the other and are not part of any group.

IV.3.2 Regression Techniques

In the proposed ML hybrid system, once data are clustered, two regression techniques (statistic and ANN) are applied to the defined clusters to get more accurate results.

Regression attempts to model the relationship between two or more variables in the data set by fitting a linear equation to the input data. One or more of the variables are the predictor ones, and the other variable is considered the criterion variable [25]. The goal of multiple regressions [26] is to learn more about the relationship between the independent or predictor variables and a dependent or criterion variable(s). These relationships can be linear or non-linear.

IV.3.2.1 Non-Linear Regression.

Non-Linear Regression (N-LR) is a regression algorithm which models observational data by a function that is a non-linear combination of the input data and depends on one or more criterion variables [27]. The parameters can take the form of an exponential, trigonometric, power, or any type of non-linear function. To determine the non-linear parameter values, an iterative algorithm is usually used. The model is defined as:

$$y = f(X, \beta) + \varepsilon \tag{IV.3}$$

Where β is a nonlinear parameter estimates to be computed, X is the dependent variables and ϵ represents the error term.

IV.3.2.2 Radial Basis Function Network.

An ANN is a simplified model of natural neural systems. The neurons are connected by weights and output signals which are the sum of the inputs to the node modified by an activation function. Different ANN models have been

tested to achieve the best imputation values, the one with the best results has been the Radial Basis Function Network (RBFN) which is defined as:

In the RBFN [28] each neuron in the hidden layer has its own n-dimensional centroid, and for each input vector $x = (x_1, x_2, \ldots, x_n)$, it computes the distance between x and the centroid of the network. A nonlinear Gaussian function distance is used to calculate the output of the neurons.

The overall output function has the form [29]:

$$\sum_{i=1}^{M} W_i * K\left(\frac{x - z_i}{\sigma_i}\right) = \sum_{i=1}^{M} W_i * g\left(\frac{||x - z_i||}{\sigma_i}\right) \tag{IV.4}$$

Where x is the input vector,$\mathrm{W}_i \ \epsilon\mathbb{R}^m$ are the weights connecting the ith neuron in the hidden-layer to the output neurons, $M\epsilon\mathbb{N}$ is the number of hidden neurons, K is a radially symmetric kernel function of a unit in the hidden layer, z_i is the centroid and σ_i is the smoothing factor of the kernel node, g: $[0, \infty) \rightarrow \mathbb{R}$ is the activation function of the output neurons.

IV.3.3 Balancing Techniques

Balancing techniques are a highly utilized resource when dealing with unbalanced data sets. When the classes of the data set are due to anomalous states, these data sets are highly unbalanced. The study of the data set, detailed in Section IV.4, shows that out of 21892 instances, only 1125 belong to the anomalous class, i.e. 5% of the total number of instances. There are mainly three types of approaches used to carry out the balancing:

IV.3.3.1 Oversampling.

This strategy tries to achieve a similar number of instances of both classes by increasing the number of instances of the minority class. In this case, it generates new instances of the anomalous class to obtain a similarity between the number of instances of both classes. The simplest technique used is Random Over Sampling (ROS), which generates new instances by duplicating existing instances of the minority class. Another method widely used in this field, which is more advanced and has a higher complexity than ROS, is Synthetic Minority Oversampling TEchnique (SMOTE) [30], which generates new artificial instances from the existing ones. It achieves this by relying on the well-known k-Nearest Neighbors (KNN) algorithm, performing an interpolation of a minority instance with other neighboring instances. Finally, another technique used in this research is the Borderline-SMOTE (BLSMOTE) [31]. As its name suggests, it is based on the SMOTE method, in which an oversampling is performed only on those instances that are on the borderline.

IV.3.3.2 Undersampling.

The balancing algorithms that follow this strategy work in a completely opposite way to what was observed in the above mentioned methods. They try to reduce the instances of the majority class in order to achieve a number of similar

instances. In the case addressed here, they eliminate the instances of the normal class. The algorithm used is Random Under Sampling (RUS), which selects completely randomly the instances of the majority class to be eliminated.

IV.3.3.3 Hybrids.

Hybrid techniques are those which use both, undersampling and oversampling algorithms at the same time. This reduces the impact of using only one of them. One of the methods presented in this research is ROS + RUS, that combines the two algorithms based on the random selection of instances above mentioned. Another technique used is SMOTE + RUS, which generates new synthetic instances of the minority class and randomly eliminates those of the majority class.

IV.3.4 Classifiers and metrics

The One-class SVM [32] [33] is one of the best known classifiers in general terms and specifically in the problems associated with one-class classification [34]. This classifier aims at identifying a hyperplane that maximizes the separation of the data instances sent to the algorithm from the training data set. In this way, once new data instances are used, it will be able to discern which class each one is, due to the universal archetype generated.

Working on unbalanced data sets, the well-known metric accuracy is not used on it's own, because it can lead to a high error in interpretation of its values, for example, the model may have the ability to distinguish only the majority class obtaining a good value, but on the other hand, the model does not have the capacity to detect the minority class. In this research, the results are shown with a wide variety of metrics, which are detailed below:

- Precision: shows the proportion of minority or anomalous class data that have been successfully labeled out of the total data labeled as anomalous.
- Recall: also known as True Positive Rate (TPR), shows the proportion of anomalous class well classified out of the real number of anomalous instances.
- F-Score: it is a metric that seeks a harmonic mean between Precision and Recall, given the difficulty in maximizing the values of both at the same time.
- AUC: this is the Area Under the Curve resulting from the visual ROC tool. It is used as an indicator of the model's ability to distinguish between classes.
- G-mean: it seeks to maximize the accuracy of both the minority and majority classes, while looking for a balance between them.

IV.4 Component-based Robot

A data set of a component-based robotic system [10] is used for this experimentation. Component-based means that the robot is made up of a variety of components that may have been manufactured by different companies, but thanks to middle-ware they are interconnected and they can work as a unified system. The middle-ware used by this robotic system is RSB Middleware [35], which is an event-based system. Within the middle-ware, a tool called rsbag is located, which is in charge of gathering the information that circulates through the middle-ware. This tool is key for the data extraction used in this paper.

The data set has been created by researchers at the University of Bielefeld (Germany) and is available in the public domain [36]. The robot was developed for its participation in the RoboCup@Home competition in 2015, where it had to carry out different tasks similar to a waiter. Some of these tasks are greeting a customer or serving a glass at the table. These actions are developed by relying on different components, for example the action of leaving a glass on the table uses a robotic arm with a gripper to pick up a glass.

Trying to detect possible errors in the robot's behavior, the authors of this data set decided to induce anomalies by software, implying that these affect the system's performance counters without penalizing the task from being carried out. For example, in the case of the robotic arm mentioned above, the anomaly causes the arm to move several times instead of once, thus penalizing these counters.

Among the various available components and given the great complexity of the experimentation, the LegDetector component has been selected, with its associated anomaly LegDetectorSkippable. This component is in charge of detecting the legs of a person in front of the robot thanks to a laser sensor. The anomaly in this case affects the counters by performing the reading attempt a number of times.

The selection of this component is due to a current problem: the need for visual processing and object recognition [37]. As the good values that were originally obtained in the experiments carried out by the authors of the data set [11] and by us in [16].

Among the attributes that constitute this component, two of them contain MV:

- received_bytes: amount of data in unit of bytes that are hosted on the interface.
- sent_bytes: amount of data in unit of bytes that are dispatched by the interface.

IV.5 Experiments and results

This section shows the results of applying the set of techniques discussed in Section IV.3 to the data set described in the Section IV.4.

The first step of the hybrid system described above is the application of three clustering techniques to the data set. To do this, the number of desired clusters (parameter k) must be provided. Estimating this value for k is not always straightforward, many techniques help us to estimate this parameter [38, 39, 40, 41]. All of them have been applied with different ranges of values for parameter k but no satisfactory results have been achieved, since the techniques return values that diverge from from each other.

Another option to estimate this value for the parameter k is the use of dendrograms. The Figure IV.2 shows the dendrogram for the original normalised data set.

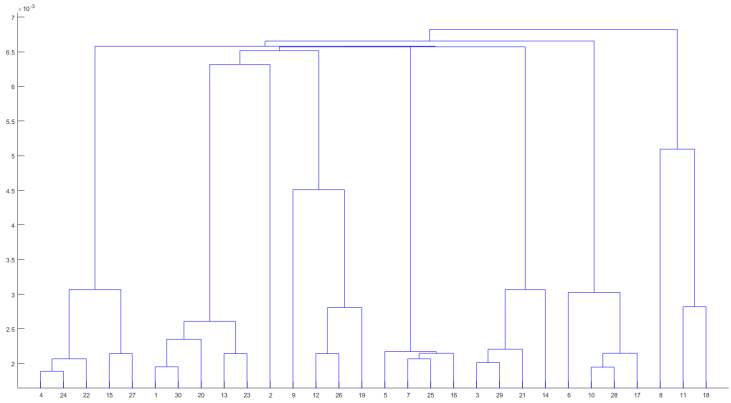

Figure IV.2: Dendrogram with 30 leaf nodes ('Euclidean' distance, 'Complete' linkage method).

Figure IV.2 depicts two clusters of data, one on the right-hand side with a few samples and the other on the left-hand side which groups most of the samples. This graphical result induces as most approximate values for the parameter k = (2, 3). The value of 3 because in the majority group on the right two subsets of data are distinguished.

Once the number of clusters has been selected, the different algorithms described in the Section IV.3.1 are applied. The clustering algorithms are applied in the data set omitting the attributes where the MV were located and described in Section IV.4. From these generated clusters, the attributes with MV are added again and the regressions are carried out with the methods detailed in IV.3.2. The imputation of MV is done in each cluster. After the classification is carried out in order to compare which combination of regression and clustering algorithms works best.

For a better understanding of the tests performed, for each of the two regression techniques, two different clustering distributions (2 and 3) have been used, and these distributions have been obtained from 3 clustering algorithms, giving a total of 12 different runs. In each of these runs, 6 balancing algorithms

and the execution of the data set without any type of treatment have been used, this last one has been denominated as “None”.

1. In Section IV.5.1 the results are analyzed from the perspective of the regression algorithms used.
2. In Section IV.5.2 from the perspective of the regression methods used.
3. In Section IV.5.3 from the balancing techniques used.

IV.5.1 Regression methods approach

An interesting comparison is the one between the two regression techniques used in this research (N-LR and RBFN) in each of the three clustering methods applied (K-means, HAC and DBSCAN). The Figure IV.3 shows a comparison of the different values achieved by each regression method in metrics. The first Figure shows the F-Score (a), where the general trend is that N-LR performs better than RBFN in most of the generated models, but taking special attention to the value achieved in DBSCAN with 3 clusters where it is observed that RBFN is slightly better. In the case of AUC (b), a generalized growth of the values is observed, with very good results in general. The values of both regression techniques reach similar values, making it difficult to conclude the best approach. Finally, the g-mean metric again shows greater variability in the results, where the general trend is that N-LR performs better in general terms, highlighting the values obtained by DBSCAN.

IV.5.2 Clustering techniques approach

A more global approach is chosen for the analysis of the clustering results, considering a wider range of metrics. It has been subdivided according to the number of clusters (k= 2, 3). Following this general approach, the results are displayed with radar plots IV.4, by this, the differences in the metrics can be recognized more intuitively. The Figure IV.4 for the two clusters(a) shows how in the general trend, Hierarchical and DBSCAN algorithms perform better than K-means, especially outperforming in g-mean and F-Score. Finally, the similarities achieved in AUC are remarkable. On the other hand, in the values achieved with 3 clusters(b), these are much more similar highlighting K-means standing out slightly in Recall.

IV.5.3 Balancing methods approach

The last approach adopted to analyze this novel hybrid system focuses on the study of the results obtained by each of the balancing methods applied. Table IV.1 shows the best values obtained for each balancing method. The values obtained by the Accuracy and Precision metrics are very good, with the peculiarity of two techniques coinciding in value, ROS + RUS in both cases. This trend continues with F-Score and g-mean where it stands out again over

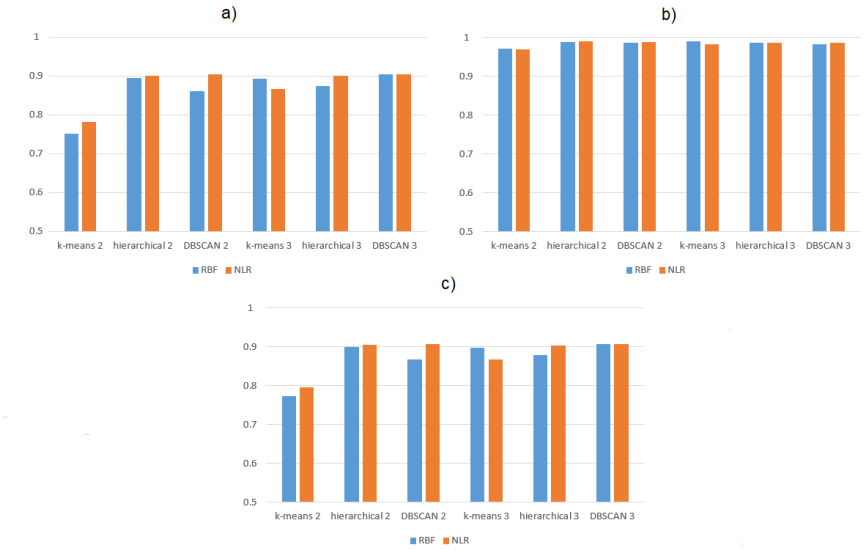

Figure IV.3: Bar plot showing the differences between RBFN and N-LR in the different metrics. a) F-Score, b) AUC and c) g-mean.

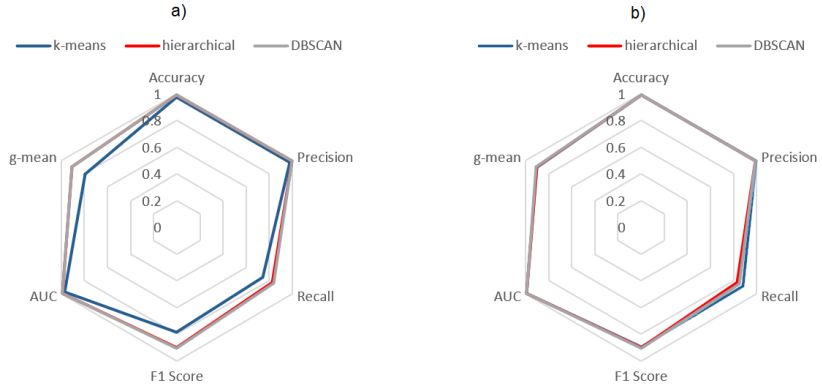

Figure IV.4: Radar diagrams are obtained from the results for each clustering technique with its different algorithms. Each one shows the results with different clusters a) k=2 and b) k=3.

the rest, although BLSMOTE results are also very good. Unexpectedly, Recall technique overcomes the other ones, where the non-application of any kind of balancing technique stands out from the rest so far.

After analyzing the different possibilities offered by this data set, there is a

Table IV.1: Metrics values for each of the balancing methods.

	None	ROS	SMOTE	BLSMOTE	RUS	ROS + RUS	SMOTE + RUS
Accuracy	0.9867	0.9846	0.9889	**0.9896**	0.9889	**0.9896**	0.9881
Precision	0.9751	**0.9964**	0.9893	0.9893	0.9929	**0.9964**	0.9929
Recall	**0.8824**	0.7790	0.8293	0.8478	0.8254	0.8457	0.8176
F-Score	0.8680	0.8696	0.9008	0.9055	0.9015	**0.9058**	0.8953
AUC	0.9694	0.9902	0.9874	0.9860	**0.9908**	0.9884	0.9903
g-mean	0.8681	0.8767	0.9042	0.9076	0.9053	**0.9081**	0.8994

general tendency that the data sets treated by the DBSCAN clustering algorithm provide better results than the other techniques, see Sections IV.5.1 and IV.5.2. The difference between RBFN and N-LR has not been very marked, although, as mentioned above, N-LR has generally performed better than RBFN. In terms of balancing techniques, the hybrid ROS+RUS technique performed better than the rest, followed closely by the BLSMOTE oversampling technique.

IV.6 Conclusions and Future Work

In this paper, a novel alternative in data imputation has been addressed, together with a set of techniques which seek to perform a complete data treatment. The proposed ML hybrid system has been validated on a data set of a component-based robotic system. The experimentation has been divided into different stages and the results obtained have been analyzed in Section IV.5.

The data set offered by the robot with anomaly information had some MV in two of their attributes (received_bytes and sent_bytes). In order to estimate these MV and have available a complete data set, the ML Hybrid System proposed in Section IV.3 was applied. A detailed analysis of each of the techniques of this hybrid system has led to the following conclusions. The three clustering techniques described offered good results, but DBSCAN stands out on the positive side and the Hierarchical technique on the negative side due to its slowness. Regarding to regression, has been performed on each of the clusters, both N-LR and RBFN obtained very good results with really low MSE values, highlighting slightly N-LR. These good results in the regression have allow to achieve a very reliable imputation on the MV and have at its disposal higher quality data set. In terms of balancing techniques, whose have been applied to the new dataset, the good performance of the hybrid ROS+RUS technique stands out overall.

The main objective of this paper was to demonstrate a novel system and the different alternatives with which to execute it, in terms of combining techniques of Machine Learning. Satisfactory results have been achieved in the implementation of the proposed hybrid model, which leads us to conclude that that the Hybrid Machine Learning System is a valid alternative for future researchers in this topic.

As future work, it would be interesting to combine new regression and clustering methods, but without losing the target of continuing with this modern hybrid system. The application of this approach to more data sets is undoubtedly an option to be considered.

References

[1] Jove, E. et al. "Anomaly detection based on one-class intelligent techniques over a control level plant". In: *Logic Journal of the IGPL* (Jan. 2020).

[2] Khalastchi, E. and Kalech, M. "On Fault Detection and Diagnosis in Robotic Systems". In: *ACM Comput. Surv.* vol. 51, no. 1 (Jan. 2018), pp. 1–24.

[3] García-Laencina, P. J., Sancho-Gómez, J.-L., and Figueiras-Vidal, A. R. "Pattern classification with missing data: a review". In: *Neural Computing and Applications* vol. 19, no. 2 (Mar. 2010), pp. 263–282.

[4] Das, S., Datta, S., and Chaudhuri, B. B. "Handling data irregularities in classification: Foundations, trends, and future challenges". In: *Pattern Recognition* vol. 81 (2018), pp. 674–693.

[5] Pigott, T. D. "A review of methods for missing data". In: *Educational research and evaluation* vol. 7, no. 4 (2001), pp. 353–383.

[6] García-Laencina, P. J., Sancho-Gómez, J.-L., and Figueiras-Vidal, A. R. "Pattern classification with missing data: a review". In: *Neural Computing and Applications* vol. 19, no. 2 (2010), pp. 263–282.

[7] Cerqueira, V. et al. "Combining Boosted Trees with Metafeature Engineering for Predictive Maintenance". In: *Advances in Intelligent Data Analysis XV*. Ed. by Boström, H. et al. Cham: Springer International Publishing, 2016, pp. 393–397.

[8] Syafrudin, M. et al. "An Affordable Fast Early Warning System for Edge Computing in Assembly Line". In: *Applied Sciences* vol. 9, no. 1 (2018), pp. 84–102.

[9] Devi, D., Biswas, S. K., and Purkayastha, B. "Learning in presence of class imbalance and class overlapping by using one-class SVM and undersampling technique". In: *Connection Science* vol. 31, no. 2 (2019), pp. 105–142.

[10] Wienke, J., Meyer zu Borgsen, S., and Wrede, S. "A Data Set for Fault Detection Research on Component-Based Robotic Systems". In: *Towards Autonomous Robotic Systems*. Ed. by Alboul, L., Damian, D., and Aitken, J. M. Vol. 9716. Cham: Springer International Publishing, 2016, pp. 339–350.

[11] Wienke, J. and Wrede, S. "Autonomous fault detection for performance bugs in component-based robotic systems". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 3291–3297.

[12] Wienke, J. and Wrede, S. “Performance regression testing and run-time verification of components in robotics systems”. In: *Advanced Robotics* vol. 31, no. 22 (Sept. 2017), pp. 1177–1192.

[13] Wienke, J. et al. “Model-based performance testing for robotics software components”. In: *Proceedings - 2nd IEEE International Conference on Robotic Computing, IRC 2018*. Vol. 2018-January. Institute of Electrical and Electronics Engineers Inc., Apr. 2018, pp. 25–32.

[14] Wienke, J. “Framework-level resouce awareness in robotics and intelligent systems”. PhD dissertation. Bielefeld University, 2018.

[15] Basurto, N. and Herrero, Á. “Data Selection to Improve Anomaly Detection in a Component-Based Robot”. In: *14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019)*. Ed. by Martínez Álvarez, F. et al. Cham: Springer International Publishing, 2020, pp. 241–250.

[16] Basurto, N., Cambra, C., and Herrero, Á. “Improving the detection of robot anomalies by handling data irregularities”. In: *Neurocomputing* vol. 459 (Oct. 2021), pp. 419–431.

[17] Arroyo, Á. et al. “Clustering and Regression to Impute Missing Values of Robot Performance”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 12344 LNAI. Springer Science and Business Media Deutschland GmbH, Sept. 2020, pp. 86–94.

[18] Suykens, J. A. and Vandewalle, J. “Least squares support vector machine classifiers”. In: *Neural processing letters* vol. 9, no. 3 (1999), pp. 293–300.

[19] Basurto, N., Cambra, C., and Herrero, Á. “Improving the detection of robot anomalies by handling data irregularities”. In: *Neurocomputing* (2020).

[20] Jain, A. K., Murty, M. N., and Flynn, P. J. “Data Clustering: A Review”. In: *ACM Comput. Surv.* vol. 31, no. 3 (Sept. 1999), pp. 264–323.

[21] MacQueen, J. et al. “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. Oakland, CA, USA. 1967, pp. 281–297.

[22] Sokal, R. R. and Rohlf, F. J. “The comparison of dendrograms by objective methods”. In: *Taxon* vol. 11, no. 2 (1962), pp. 33–40.

[23] Day, W. H. and Edelsbrunner, H. “Efficient algorithms for agglomerative hierarchical clustering methods”. In: *Journal of classification* vol. 1, no. 1 (1984), pp. 7–24.

[24] Ester, M. et al. “A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[25] Yale, U. of. *Linear Regression*. 2017.

[26] Pearson, K. and Lee, A. “On the Generalised Probable Error in Multiple Normal Correlation”. In: *Biometrika* vol. 6, no. 1 (1908), pp. 59–68.

[27] Neter, J. et al. *Applied linear statistical models*. Vol. 4. Irwin Chicago, 1996.

[28] Lippmann, R. P. “Pattern classification using neural networks”. In: *IEEE Communications Magazine* vol. 27, no. 11 (Nov. 1989), pp. 47–50.

[29] Park, J. and Sandberg, I. W. “Universal Approximation Using Radial-Basis-Function Networks”. In: *Neural Computation* vol. 3, no. 2 (June 1991), pp. 246–257.

[30] Chawla, N. V. et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* vol. 16 (2002), pp. 321–357.

[31] Han, H., Wang, W. Y., and Mao, B. H. “Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning”. In: *Lecture Notes in Computer Science*. Vol. 3644. Springer Verlag, 2005, pp. 878–887.

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

[33] Boser, B. E., Guyon, I. M., and Vapnik, V. N. “A Training Algorithm for Optimal Margin Classifiers”. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT ’92. Pittsburgh, Pennsylvania, USA: ACM, 1992, pp. 144–152.

[34] Shin, H. J., Eom, D. H., and Kim, S. S. “One-class support vector machines - An application in machine fault detection and classification”. In: *Computers and Industrial Engineering* vol. 48, no. 2 (Mar. 2005), pp. 395–408.

[35] Wienke, J. and Wrede, S. “A middleware for collaborative research in experimental robotics”. In: *2011 IEEE/SICE International Symposium on System Integration (SII)*. Dec. 2011, pp. 1183–1190.

[36] Wienke, J. and Wrede, S. *A Fault Detection Data Set for Performance Bugs in Component-Based Robotic Systems*.

[37] Kasaei, S. H. et al. “Towards lifelong assistive robotics: A tight coupling between object perception and manipulation”. In: *Neurocomputing* vol. 291 (2018), pp. 151–166.

[38] Caliński, T. and Harabasz, J. “A dendrite method for cluster analysis”. In: *Communications in Statistics-theory and Methods* vol. 3, no. 1 (1974), pp. 1–27.

[39] Davies, D. L. and Bouldin, D. W. “A cluster separation measure”. In: *IEEE transactions on pattern analysis and machine intelligence* vol. PAMI-1, no. 2 (1979), pp. 224–227.

[40] Rousseeuw, P. J. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* vol. 20 (1987), pp. 53–65.

[41] Tibshirani, R., Walther, G., and Hastie, T. “Estimating the number of clusters in a data set via the gap statistic”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* vol. 63, no. 2 (2001), pp. 411–423.