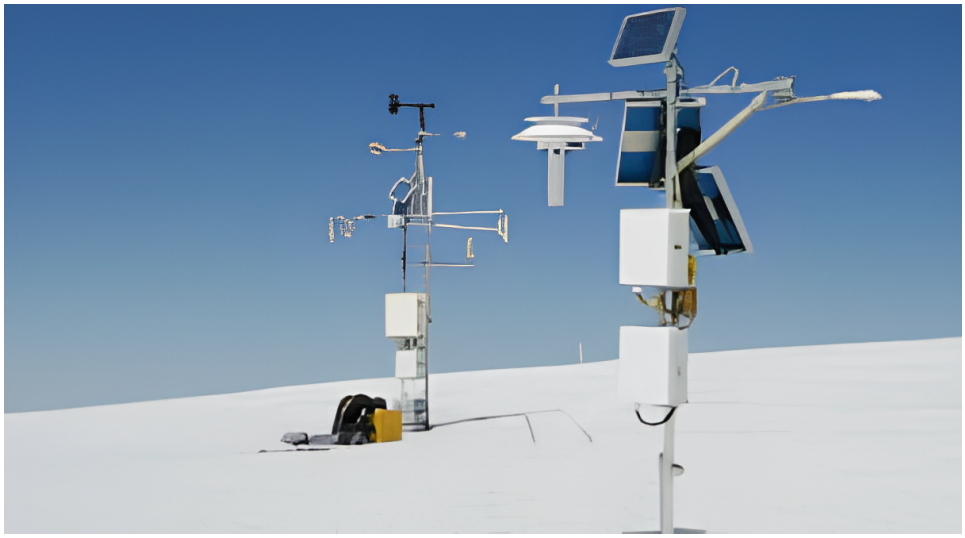
Data analysis and machine learning approaches for time series pre- and post- processing pipelines

Amaia Gil Lerchundi
Universidad del País Vasco/Euskal Herriko Unibertsitatea
April 2022

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea



UNIVERSITY OF THE BASQUE COUNTRY
Faculty of Informatics

Data analysis and machine learning approaches for time series pre- and post- processing pipelines

Supervised by:
Basilio Sierra Araujo
and
Marco Quartulli

Graduation thesis submitted by:
Amaia Gil Lerchundi
For the academic degree of Doctoral Programme in Informatics Engineering

April 2022

Acknowledgements

The research presented in this thesis has been done at Vicomtech and University of the Basque Country. I would like to thank these organisations for giving me the opportunity to carry out the research.

I would like to thank Igor G. Olaizola for introducing me to the field of research and for his continuous encouragement.

I wish to express my sincere gratitude to my supervisors. Thanks to Professor Basilio Sierra for being my advisor, for his interesting discussions, his optimistic point of view and his ability to bring an external perspective to our daily work. Thanks to Dr. Marco Quartulli for his support, guidance and patience during the whole research period, not only from a professional point of view but also on a personal level and always with a touch of humour. The assistance of these mentors and their confidence in my work have been very important.

I am grateful to my colleagues at the Data Intelligence for Energy and Industrial Processes of Vicomtech, as well as to all my other co-workers in the technology research centre. From the first coffee in the morning, continuing with the break and lunchtime, your friendship and encouragement have been fundamental to finish this thesis.

I would like to thank my friends for their unconditional support, especially Amaia and Sandra for always blindly believing in me.

I wish to thank all my family. Great thanks to my parents and sisters for instilling in me the spirit of sacrifice to achieve my objectives. Their unwavering support has been essential.

Finally, I would like to thank Urko, for being so understanding and for all the love he gives me. Thank you for always being there. Last but not least, to the new soon coming member to our family, we cannot wait to see you.

Amaia Gil Lerchundi
April 2022
Donostia-San Sebastián

Contents

Acknowledgements **v**

List of Figures **x**

List of Tables **xi**

Glossary **xiii**

Abstract **xvi**

I Body of the dissertation **1**

1 Background **3**
1.1 General introduction 3
1.2 Research environment 4
1.2.1 Vicomtech 4
1.2.2 Projects 5
1.3 Structure of the dissertation 7

2 Motivation **9**
2.1 Where to push the boundaries: context 9
2.2 An illustrative example 12
2.3 Research questions 13
2.4 Research outcomes 14

3 Theoretical concepts **17**
3.1 Time series definition and properties 17
3.2 Machine learning concept definitions 19

4 Compression **25**
4.1 Background . . . 25
4.2 Case study . . . 26
4.3 Innovation . . . 27
4.4 Conclusions and future work . . . 32

5 Joining **35**
5.1 Background . . . 35
5.2 Case study . . . 36
5.3 Innovation . . . 37
5.4 Conclusions and future work . . . 41

6 Imputation **43**
6.1 Background . . . 43
6.2 Case study . . . 44
6.3 Innovation . . . 46
6.4 Conclusions and future work . . . 48

7 Surrogate models **53**
7.1 Background . . . 53
7.2 Case study . . . 54
7.3 Innovation . . . 56
7.4 Conclusions and future work . . . 57

8 Conclusions and future work **59**
8.1 Conclusions . . . 59
8.2 Future work . . . 60

Bibliography **69**

II Appended Papers **71**

9 Summary of the appended papers **73**
9.1 Paper 1 . . . 73
9.2 Paper 2 . . . 74
9.3 Paper 3 . . . 74
9.4 Paper 4 . . . 75

10 Appended papers **77**

List of Figures

2.1 General idea schema. 11
2.2 Meteorological global observing system schema 12

3.1 Time series descomposition. 18
3.2 Scatter showing covariance relation between time series. 19
3.3 Examples showing the difference between missing data and outliers. . 20
3.4 The three pillars of learning in data science: clustering, classification and regression. 21
3.5 Ensemble architecture. 21
3.6 Confusion matrix example and evaluation metrics definition. 22
3.7 K-fold cross-validation method schema. 23

4.1 Schema of the smart compression method. 28
4.2 Compression example. 29
4.3 Smart compress methodology schema. 30

5.1 Most common joining strategies applied to a piecewise constant example. 38
5.2 Smart joining application schema. 39
5.3 Smart join methodology schema. 40

6.1 Three types of amputation strategies applied to a common example curve. 45
6.2 Imputation strategy application schema. 47
6.3 Imputation model generation. 49
6.4 Uncertainty in two signal examples depending on missing observation distribution. 50
6.5 Imputation methods comparison to two different signals and three amputation techniques. 51

7.1 Lithium-ion battery schema 55
7.2 The general structure of the creation of the surrogate model 57

List of Tables

4.1 Grouped sum of mean errors of MeAE values. 31

5.1 Error values for the different join methods in the application example. 41

Glossary

bfill Backward fill, using next valid observation. 37, 40

ECG electrocardiograph. 31

EOG electrooculography. 31

ffill Forward fill, propagating the last valid observation. 37, 40

IIoT Industrial Internet of Things. xv, 3–5, 35

Li-ion Lithium ion. 54–56

ltob largest-triangle-one-bucket. 27, 31

lttb largest-triangle-three-buckets. 27, 31, 32

m4a M4 algorithm. 27, 31

MAR Missing at random. 44, 45

MCAR Missing completely at random. 44, 45

MeAE Mean Absolute Error. xi, 31, 32

mmb mode-median-bucket. 27, 31

MNAR Missing not at random. 44, 45

NA Non-Available. 37, 40, 41

NMS National Meteorological Services. 13

oen one each n. 27, 31

SQL Structured Query Language. 36, 37, 41

Abstract

In the industrial domain, different kinds of sensor devices capture data continuously and constantly monitor the operation of the machines in real-time. The cost and size of sensor have reduced dramatically in recent years, making the digitisation of machines and processes affordable. In the context of the Industrial Internet of Things (IIoT) concerns come from both the large quantity and the sometimes low quality of data that it is typically messy, as observations can be noisy, missing or lost in communications. These limitations can lead to results that negatively impact business decisions.

This research focuses on time series data, which poses unique challenges due to the need to properly take into account autocorrelation, trends, seasonality, and gaps. Moreover, as time series data is often continuously generated, it is important that cleaning algorithms support near real-time operation. Furthermore, as the data evolves, the cleaning strategy needs to change in an adaptive and incremental way, in order to avoid having to start the cleaning process from scratch each time.

The objective of this thesis is to verify the possibility of applying machine learning-inspired process flows to data pre-processing steps. For that purpose, this work proposes methods that are capable of selecting optimal pre-processing strategies and providing insight into the errors made. The proposed methods generate pre- and post-processing models which are trained using available historical data, by minimising empirical loss functions.

Specifically, this dissertation studies time series compression, feature joining, observation imputation and surrogate model generation processes. In each of them, the optimal selection and combination of multiple strategies is pursued. This approach is defined according to data characteristics and user-defined system properties and limitations.

The general results indicate that the proposed approach identifies optimal pre- and post-processing strategies for univariate time series on a window-by-window basis, showing its capability to adapt to the current signal window. Specific details of the data generation process, of the dependency on other internal or external variables, or even of noise can affect the pre-processing selection results. Controlling the error in

the process is critical in order to detect model drifts and, as a consequence, to retrain the generated model to maintain data quality. Implementation results have allowed ensuring data quality control in real-world project scenarios.

Part I

Body of the dissertation

CHAPTER 1

Background

This chapter starts with a general introduction describing the Industry 4.0 environment and then continuous with the presentation of the research centre and projects where this work has been realised. Finally, the structure of the dissertation is detailed.

1.1 General introduction

The cost and size of sensor technology has been reduced dramatically in recent years, making the digitisation of the instrumentation of machines and processes technically affordable. Therefore, thanks to the Industrial Internet of Things (IIoT) both the discrete manufacturing industry (machine tools, parts and components, electronics, etc.) and the process industry have produced a rapid increase in data volumes [67]. In order to obtain knowledge from the collected data, this data needs to be analysed by domain experts and data analysts. Combining their knowledge and experience in the domain, they are capable of developing tools to discover non-evident patterns and relationships in production data, thus boosting their decision-making capacity.

However, in IIoT, the concern comes in handling vast quantities of unstructured data as well as sensor data from multiple devices. Therefore, in order to gain value from these data there has to be an alternative way to handle and manage it. The rapid increase over the last decade in data volumes, cloud storage, rental computing power, and network connectivity has enabled analysis of operational data that was previously impossible. Furthermore, streaming data velocity in the IIoT context, requires real-time or as close to real-time as possible handling and analysis. This constraint puts additional pressure on data storage and handling systems [73]. Another characteristic of this collected data is that it is typically messy, as observations from sensor data are often missing or lost in communications and therefore requires considerable tidying up before processing.

Through the latest advances in sensor technologies, sensors instead of just being

precise, can have self-awareness and can even predict their remaining useful life. Similarly, machine sensors through their controllers can be self-conscious, self-predict and self-compare [27]. Combined with this new breed of self-informed and self-predicting components, analytic processes can provide accurate predictive maintenance schedules for machinery and assets, keeping them in productive service longer and reducing the inefficiencies and costs of unnecessary maintenance. Monitoring for productivity and diagnosis are a valuable suite of functions that permit the detection and prediction of the occurrences of faults or problems. Having the ability to detect and proactively redress problems is very important in industrial environments.

Summarising, the IIoT provides a closer insight into company's operations and assets through integration of machine sensors, cloud computing and storage systems. Once the collected data is processed by means of advanced analytic processes, it provides a method of transforming business operational processes. Finally, business gains come from operational efficiency gains and accelerated productivity, with expected results in reduced unplanned downtime and optimised efficiency [65].

1.2 Research environment

This section describes the research centre and some of the projects in which this work has been done.

1.2.1 Vicomtech

Activities described in this work have been carried out at Vicomtech[1], an applied technological research centre located in Donostia - San Sebastián (Basque Country, Spain) that combines basic and applied research with the aim of transferring technology to the industry. More specifically, one of the main missions of Vicomtech involves meeting the applied research, technology development and innovation requirements of local companies and institutions in artificial intelligence, visual computing and interaction to enhance their competitiveness. To this end, Vicomtech promotes the development and creation of product prototypes and applications in collaboration with the industry. Besides, Vicomtech aims at contributing to universal knowledge by the publication of scientific results.

Nowadays, Vicomtech is composed of seven technological departments, each seeking to develop and apply technology in different fields and industries, as listed below:

- industry and advanced manufacturing
- digital media
- speech and natural language technologies
- data intelligence for energy and industrial processes
- intelligent transport systems and engineering

[1] https://www.vicomtech.org/en

- digital security
- ehealth and biomedical applications

This dissertation work corresponds to data intelligence for energy and industrial processes department and reflects the evolution and growth of data pre-processing and analysis in the IIoT domain. This evolution was given mainly by (1) the expertise acquired during the execution of several local and European projects, (2) the strong collaboration with other fields and departments, (3) the knowledge of the market and the industry in the corresponding technologies, and (4) all the research and development activities that aspects described above have originated overall.

The Data Intelligence for Energy and Industrial Processes department has extensive experience in the modelling and probabilistic characterisation of systems. In this sense, its experience covers disciplines as diverse as remote sensing, image characterisation, thermography and, above all, data processing through machine learning and Artificial Intelligence techniques. The department applies all these capabilities in the sectors it addresses (industrial activity, energy, agriculture, environmental management, etc.) increasing the competitiveness of these sectors through the prediction and optimisation of processes as well as improving their interpretability. In addition, the Data Intelligence department works on aspects of integration and interoperability of services from the capture of the data itself (through open industrial protocols) to its deployment in edge/cloud infrastructures.

Furthermore, the department supports teams for analysis and the decision making of experts in the productive process by means of the development of visual interaction systems. An important line of work in Visual Analytics is that related to Business Intelligence. By means of interactive visualisation of the information by experts, supported by Data Intelligence algorithms, we can achieve a major improvement in the comprehension of the current situation in business decision-making, such as giving instructions and allowing the definition of alternative scenarios in the quest for better solutions or as means of a tool for causality analysis.

The main strength of the group lies in the fact that, thanks to its experience in R&D projects, it has been trained in the modelling of highly complex systems and in turn implements, integrates and deploys the algorithms designed in a first phase of analysis and experimentation with data in which the final modules of prediction, optimisation, recommendation, etc. are implemented in conjunction with the business logic of each case.

1.2.2 Projects

The following are some projects, in which I have participated, that have taken place during the development of this thesis project.

Sparta - Strategic programs for advanced research and technology in Europe[2] (H2020 2019-2022)

With the real-world applications of AI came the realisation that its security requires immediate attention. Malicious users can skillfully influence the inputs fed to the AI algorithms in a way that changes the classification or regression results. The role of the program lies in conducting a thorough analysis of the threats and risks for AI. This is followed by providing mechanisms and tools to counter the deteriorating effects of recognised dangers in a variety of critical AI applications, making them safe and secure from the possibility of being compromised. The evaluation of machine learning models robustness in adversarial settings is not a trivial task. If the model is robust to a particular kind of attack, it is not a sufficient measure of its robustness. In order to have confidence in the predictions made by the model, it is needed to check its robustness against a variety of attack techniques.

Petroscopio (Industrial 2016-2019)

Generation of tools for analysis, diagnostics and monitoring of the operation by means of visualisation tools. This system also should integrate the complete process from data queries and table joining to model generation. With that objective in mind, the system should allow selecting, filtering and integrating data from multiple databases and data input characteristics. Once the data is integrated, the domain expert would use the tools for comparing and relating different operating variables in order to construct machine learning models for predicting and classifying future events. Summarising, the tools should provide domain experts the capability of training models ensuring all the pre-processing and integration steps that happen in between.

DataPump (Industrial 2016-2019)

Generation of a tool that allows downloading selected variables from multiple databases in user determined format. By the use of an interface, a user without any knowledge of query language should be capable of generating a data table that contains user selected variables. Apart from the variables, the user can indicate desired time windows and data sampling. With this tool, production process engineers and domain experts have the accessibility to data in order to carry out their investigations.

Mainwind+ (HAZITEK 2015-2018)

The aim of the project was offering specific solutions to wind turbine manufacturers who demand, above all, to know the real-time behaviour of components in order to test their reliability and reduce costs. The challenge was to exploit the potential offered by the information generated by the components developed so far, providing intelligent sensorisation, communication, storage and data exploitation technologies, and integrating them into the entire value chain of the wind business. This would

[2]https://www.sparta.eu/

make it possible to predict the behaviour of parts during use, reduce the risk of failure and optimise the logistics of spare parts. For that purpose, different visualisation tools were generated that allowed the integration of multiple data sources. As visualisations were integrated in a unique online platform, the different data sources should be joined and the data volume reduced before introducing them in the system.

Vix3D (Industrial 2019-2022)

Industrial process managers need to investigate the status in their production processes for resolving problems and optimise the processes. As big amounts of data are produced daily, it is not possible to revise all data manually in detail. The objective of the project was enabling the user through the use of synchronised visualisations to select the time windows of interest. Due to the volume of data, the visualisation were generated from reduced number of points capable of representing the general behaviour of the selected signals.

1.3 Structure of the dissertation

This dissertation is divided in two parts: Part I containing the body of the dissertation; and Part II first starts with a summary of the significant papers that are later included. Part I is organised in nine main chapters considering this introduction. A brief description of the content of each chapter follows.

Chapter 1: Background. This chapter starts with a general introduction. Then, the research environment is described introducing the research centre and the main projects where this research has been carried out. Finally, the structure of the thesis is detailed.

Chapter 2: Motivation. The main context of the research is presented, together with the research questions and outcomes.

Chapter 3: Theoretical concepts. This chapter presents definition and properties of the theoretical concepts which are used in the dissertation.

Chapter 4: Compression. The proposed solution to obtain an adaptative observation selection methodology is presented.

Chapter 5: Joining. This chapter details the optimisation methodology proposed to select a SQL joining method that satisfies user defined properties in the results.

Chapter 6: Imputation. A methodology to deal with observation imputation strategy selection is described in this chapter.

Chapter 7: Surrogate models. This chapter explains how to apply machine learning based ensemble models as surrogate models in order to substitute a Lithium ion battery performance simulation model.

Chapter 8: Conclusions and future work. Finally, this parts ends with general conclusions and future work of the dissertation.

CHAPTER 2

Motivation

This chapter introduces the context of the research, describing the specific problem to be solved and indicating the objectives. Later, the specific research questions to be resolved are detailed. Finally, the scientific contributions generated during the research done are listed.

2.1 Where to push the boundaries: context

Uncontrolled data quality can often result in erroneous upstream results that could impact business decisions negatively. In view of that, one of the critical challenges is to maintain large data warehouses and, at the same time, to ensure that the quality of the data remains high [22]. Data cleaning process is in charge of maintaining high data quality. Furthermore, this transformation is typically done as batch process, operating on the whole dataset without any feedback. This leads to long, frustrating delays during which users have no information of the effectiveness of the transformation [58]. Finally, data quality is evaluated in a data profiling process, which usually involves calculating several aggregate data statistics that form the data profile.

Incorporating the user into the cleaning process is critical to achieve high quality results. Generally, this process is studied as a separated problem and there are no guides on options available for building or choosing an adequate solution. The main advantage of domain-specific solutions is that they can incorporate knowledge of the domain while developing the solutions. As the domain is known, the decisions taken can often be comprehensive, are easier to deploy and validate the results. In general, a significant portion of the data cleaning and data transformation is done manually or by low-level programs that are difficult to write and maintain [57].

The data cleaning process should be supported by tools to limit manual inspection and programming efforts and be extensible to easily cover additional sources. In addition, data cleaning and other data transformations should not be performed in

isolation, and be specified in a declarative way for its later use in other data sources as well as for query processing. Specially for data warehouses, a workflow infrastructure should be supported to execute all data transformation steps for multiple sources and large data sets in a reliable and efficient way.

This research is centered in time series data. Time series data can be defined as a sequence of random variables, $x_1, x_2, \cdots, x_n$, where the random variable x_1 denotes the value taken by the series at the first time point, the variable x_2 denotes the value for the second time period, x_n denotes the value for the n-th time period, and so on. Time series have been widely used in many fields such as financial economy, meteorology and hydrology, signal processing and manufacturing industry. In the particular case of industry, time series data are specially important. In this domain, different kinds of sensor devices capture data continuously and constantly monitor the operation of the machine in real-time. However, these sensor devices often save erroneous values, thus generating low quality data [77].

In recent works, the data quality issues in time series data are studied, which pose unique challenges due to the presence of autocorrelations, trends, seasonality, and gaps in the time series data. Moreover, as time series data are often uninterruptedly generated, it is important that cleaning algorithms support online operation (real-time operations). Furthermore, as the data evolves, the data cleaning strategy needs to change in an adaptivity way. The incremental changes should be identified and updated, in order to avoid having to start the cleaning process from scratch each time [76]. This online cleaning algorithm can monitor the data quality, detect the problem and then promptly alarm or perform a reasonable cleaning. Finally, this process needs to avoid changing the original correct data and be based on the principle of minimum modification, that is, the smaller the change, the better [14, 1, 20].

The data generated by numerical models are capable of representing physical problems with high precision. However, each new prediction requires excessive computational time, making them not suitable for near real-time applications. In order to include the valuable information proportioned by the numerical models in the system, surrogate models can be used. When these models are trained with sufficient data from simulation results, thus are capable of mimicking the physical relations between the input and output parameters. By the use of these models, each evaluation result can be integrated directly together with the rest data in the data warehouse.

The objective of this research is providing guidelines to the pre-processing steps described in this section and tools to control and monitor the data availability and quality on the system. A general overview of the data quality process is described in the Figure 2.1.

Input data from multiple sources should be integrated in a common data warehouse. Previous to the integration, first, the quality of the data is evaluated and cleaned. Then, the cleaned data is compressed before saving it in the data warehouse. For the simulated data, depending on the data availability needed, surrogate models are generated to ensure that data can be provided in near real-time. On the one hand, the data in the warehouse is monitored by the system manager to ensure the quality of data is maintained. On the other hand, database manager uses the query tool to revise the data available in the warehouse. Finally, the desired data is

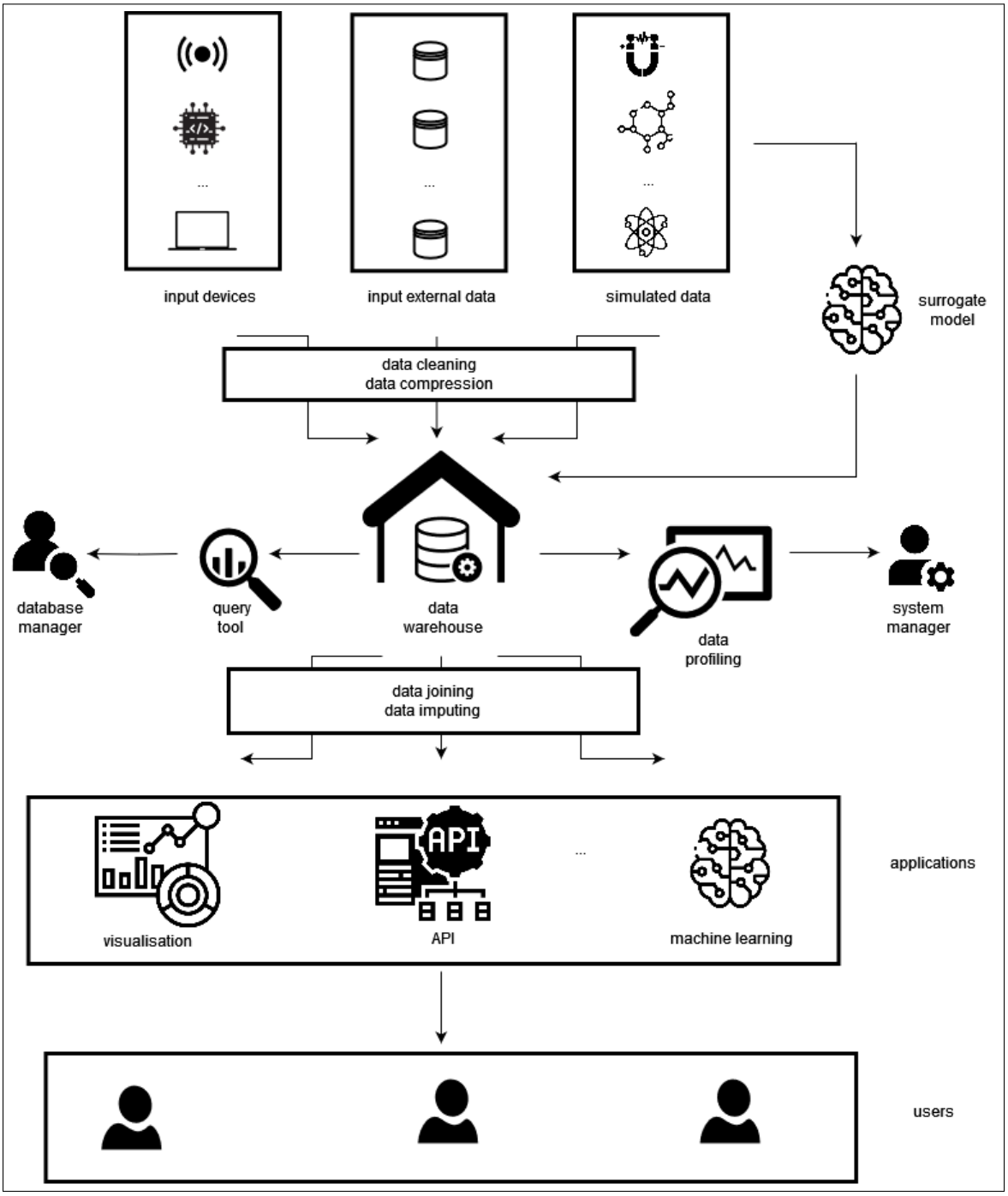


Figure 2.1: General idea schema. Data from multiple sources is collected and cleaned a compressed previous saving it in the warehouse. Simulation results are integrated by surrogates models in the data collection process. Later, the desired data is joined and the gaps imputed for its integration in previously defined services.

joined and the gaps imputed for its integration in previously defined services in which final users are connected.

2.2 An illustrative example

Networks of meteorological sensors are essential to monitor atmospheric processes, and to assess both long-term climate change and short-term weather events. Sensor network nodes typically have a set of design goals including sensor integration, data quality, size, cost, robustness and power management. Due to the harsh nature of most of the host environments, they are designed to be robust but there may be difficulties with transmitting through the environment being sensed, e.g. through ice, water or forests. Different types of data are collected by the sensor nodes, but some of the variables such as precipitation, ground water and surface water chemistry, vegetation and animal observations continue being measured manually.

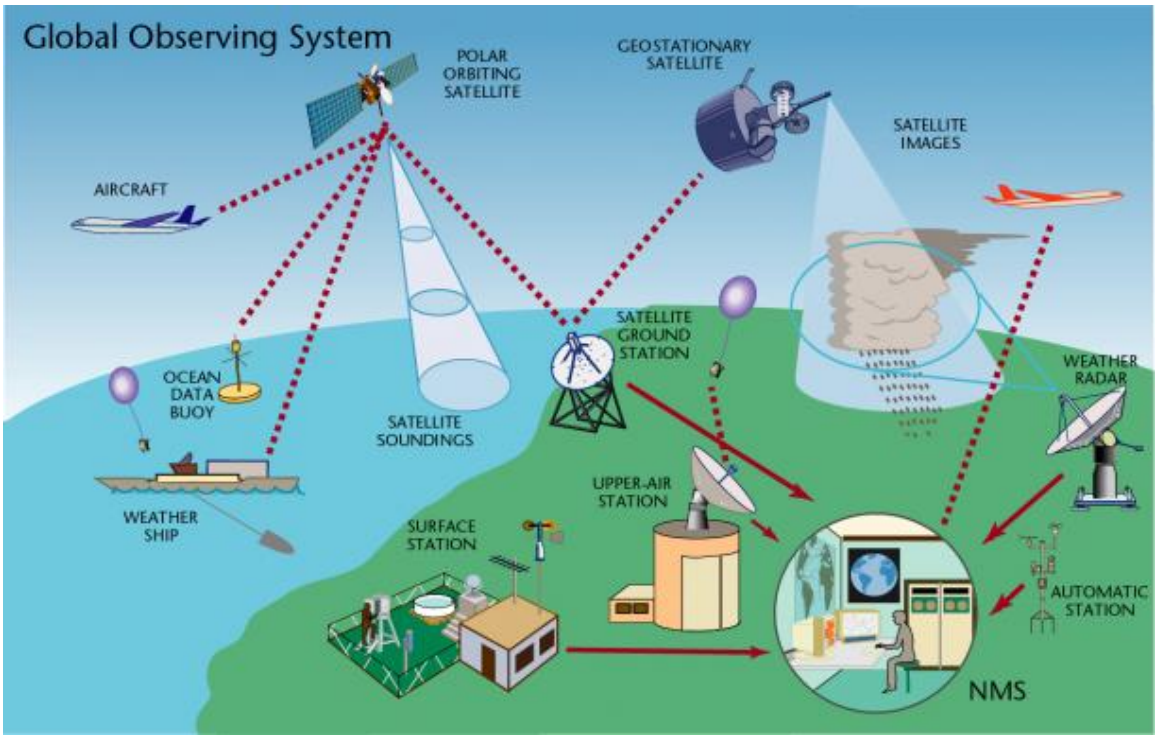


Figure 2.2: Meteorological global observing system schema[1]

Figure 2.2 shows the different elements that provide meteorological information and are part of the global observing system. Surface station and automatic stations located all around the world provide at least every three hours and often hourly meteorological parameters such as atmospheric pressure, wind speed and direction, air temperature and relative humidity. Upper-air stations, radiosondes, attached to free-rising balloons, make measurements of pressure, wind velocity, temperature and

[1]https://public.wmo.int/en/programmes/global-observing-system

humidity from just above ground to heights of up to 30km. Over the oceans the Global Observing System relies, in addition to satellites, on ships, moored and drifting buoys and stationary platforms. They comprise much the same variables as at surface land stations with the important additions of sea surface temperature, wave height and period. Aircraft-based observation provide reports of pressure, winds, temperature, humidity, turbulence and other parameters during flight. Polar orbiting and geostationary satellites are normally equipped with visible and infra-red imagers and sounders from which one can derive many meteorological parameters. Several of the polar-orbiting satellites are equipped with sounder instruments that can provide vertical profiles of temperature and humidity in cloud-free areas. Geostationary satellites can be used to measure wind velocity in the tropics by tracking clouds and water vapour. Finally, weather radars have been used in the detection of precipitating water droplets and the derivation of rainfall rates within clouds for decades. All the information collected in different sources is received in the national meteorological services (NMS) where data is processed and integrated in the forecasting systems.

For all these reasons, meteorological forecasting remains challenging due to the strong heterogeneity of land cover, the data management and access difficulties and the multiscale meteorological dynamics. Data pre-processing steps are essential in order to construct accurate forecasting models. Large data volumes from multiple origins should be integrated in a single system with difficulties such as different data samplings and missing observations. Furthermore, sensor network data should be combined with numerical model results that provide information that cannot be measured directly or from locations where data coverage is not available.

This thesis present a set of novel methods that can be integrated in a pipeline aiming at composing such heterogeneous, streaming, uncompressed, incomplete data from sensors, potentially integrating them with simulators via surrogates capable of generating samples in near-real time.

2.3 Research questions

Based on the previous experience of Vicomtech and its closeness to real industrial applications a set of common scenarios is identified. These scenarios include:

- no supervised sampling / compression strategy
- erroneous values stored and no unified strategy for indicating missing values
- misalignment in data

This context identified by Vicomtech is quite frequent. That is, clients and databases can vary, but the decisions previously taken of pre-processing steps or their supervision are similar. For that reason, this thesis formulates and tries to answer the following Research Questions, that provide solutions to the data cleaning processes that fall in the previous scenarios:

1. How to automatise or select pre-processing depending on system derived properties?

2. How can these pre-processing steps adapt to new characteristics without needing to change completely the framework?

3. How can the user manage / monitor the pre-processing steps?

This work is an attempt to provide answers to this set of research questions.

2.4 Research outcomes

Regarding the scientific contributions generated in the framework of this research, the following articles have been published and include contributions of the author. Journal articles:

(a) A. Gil, M. Quartulli, I. G. Olaizola, B. Sierra, **Learning Optimal Time Series Combination and Pre-Processing by Smart Joins**, in Applied Sciences, vol. 10, no. 18:6346, 2020.

(b) A. Gil, M. Quartulli, I. G. Olaizola and B. Sierra, **Towards Smart Data Selection From Time Series Using Statistical Methods**, in IEEE Access, vol. 9, pp. 44390-44401, 2021.

(c) M. Quartulli, A. Gil, A.M. Florez-Tapia, P. Cereijo, E. Ayerbe, and I.G. Olaizola. **Ensemble Surrogate Models for Fast LIB Performance Predictions**. Energies 2021, 14, 4115.

(d) X. Echeberria-Barrio, A. Gil-Lerchundi, J. Egana-Zubia and R. Orduna-Urrutia. Understanding Deep Learning defenses Against Adversarial Examples Through Visualizations for Dynamic Risk Assessment. Neural Computing & Applications. Springer, 2022.

(e) X. Echeberria-Barrio, A. Gil-Lerchundi, R. Orduna-Urrutia and Iñigo Mendialdua. Optimized Parameter Search Approach For Weight Modification Attack Targeting Deep Learning Models. Applied Sciences. (Accepted)

(f) A. Gil, M. Quartulli, I. G. Olaizola and B. Sierra, **ASSIST: Automatic Smart Selection of the Suitable Imputation Technique**. (Submitted)

Conference articles:

(a) A. Gil, E. Ayerbe, I. Urdampilleta, O. Miguel, H. J. Grande, F. Varas and L. Saavedra. Li-ion Cell design optimization based on 3D electric-thermal model. In Symposium for Fuel Cell and Battery Modeling and Experimental Validation (ModVal13), 2013.

(b) G. Nico, A. Gil, M. Quartulli, P. Mateus and J. Catalao. Merging InSAR and GNSS meteorology: How can we mine InSAR and GNSS datasets to extract and visualize information on atmosphere processes? In Proceedings of the 2017 conference on Big Data from Space (BIDS' 2017), pp. 375-378, 2017

(c) X. Echeberria-Barrio, A. Gil-Lerchundi, I. Goicoechea-Telleria and R. Orduna-Urrutia. Deep Learning Defenses Against Adversarial Examples for Dynamic Risk Assessment. 13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020). Advances in Intelligent Systems and Computing, vol. 1267, pp. 316-326, 2020.

(d) J. Franco, A. Garcia and A. Gil. Multivariate Adaptive Downsampling Algorithm for Industry 4.0 Data Visualization. 16th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2021). Advances in Intelligent Systems and Computing, vol 1401. Springer, 2021.

From the previous, only those publications with major contributions (remarked in bold) on the author and answering the research questions are presented in Part II.

CHAPTER 3

Theoretical concepts

This chapter introduces time series data and machine learning concepts that will be used in the following chapters.

3.1 Time series definition and properties

When a variable is measured sequentially in time over or at a fixed interval, known as the sampling interval, it forms a *time series*. The term *univarite time series* refers to a time series that consists of a single observation recorded sequentially in time and *multivariate time series* is used when multiple dependent variables observations are received each time. A time series of length n can be represented by $\{x_t : t = 1, \ldots, n\} = \{x_1, x_2, \ldots, x_n\}$, which consists of n values sampled at discrete times $1, 2, \ldots, n$. When all the observations between specific start and end time are extracted from a time series, the term *time (series) window* is used to refer to it.

The special structure of time series produces unique challenges for machine learning researchers. A consideration due to the special nature of time series is the fact that individual observations are typically highly related with their neighbours in time. Indeed, it is this property that makes most time series excellent candidates for dimensionality reduction by compression. However, for learning algorithms that assume the independence of features, this lack of independence must be countered or mitigated in some way.

The main features of many time series are trends and seasonal variations that can be modelled deterministically with mathematical functions in time. A systematic change in a time series that does not appear to be periodic is known as *trend* and the repeating pattern within any fixed period is called *seasonality*. Figure 3.1 shows a descomposition of a time series using its trend and seasonality patterns. *stationary* time series is one whose properties are constant.

The *covariance* is a measure of the linear association between two variables. Being

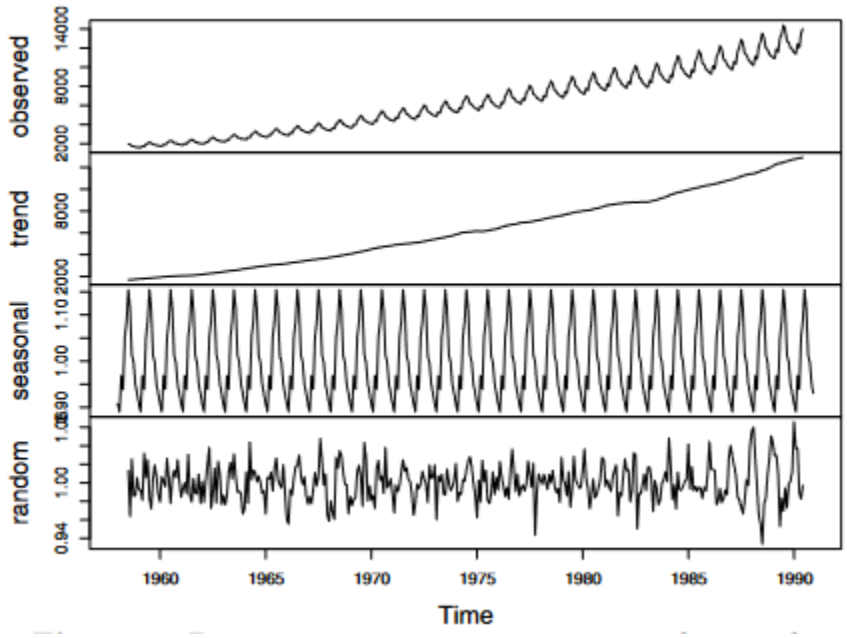


Figure 3.1: Time series example and its decomposition using trend and seasonality [49].

$\overline{x}$ and $\overline{y}$ the sampling estimates of mean values of variables x and y, the covariance is defined as follows:

$$Cov(x, y) = \sum (x_i - \overline{x})(y_i - \overline{y})/(n-1)$$

If the data pairs are plotted, the lines $x = \overline{x}$ and $y = \overline{y}$ divide the plot into quadrants. Points in the lower left quadrant and in the upper right contribute to the covariance in positive manner. In contrast, points in the upper left and lower right quadrants make a negative contribution to the covariance. Thus, if y tends to increase when x increases, most of the points will be in the lower left and upper right quadrants and the covariance will be positive. Conversely, if y tends to decrease as x increases, the covariance will be negative. If there is no such linear association, the covariance will be small relative to the standard deviations of $\{x_i\}$ and $\{y_i\}$. Figure 3.2 shows two time series samples that have a positive covariance value between them.

Correlation is a dimensionless measure of the linear association between a pair of variables (x, y) and is obtained by standardising the covariance by dividing it by the product of the standard deviations of the variables. Correlation takes a values between -1 and +1, with a value of 0 indicating no linear association. The sampled correlation is calculated using the following equation:

$$Cor(x, y) = \frac{Cov(x, y)}{sd(x)sd(y)}$$

where $sd(x)$ and $sd(y)$ are the sampling estimations of standard deviation values of variables x and y. A correlation of a variable with itself at different times is known as *autocorrelation*.

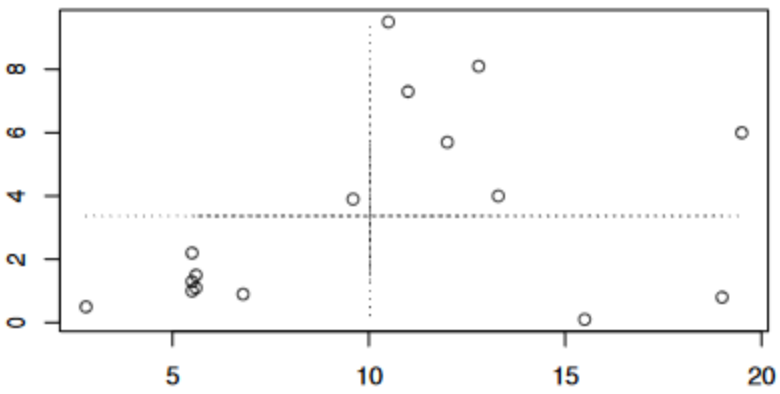


Figure 3.2: Scatter plot generate from two time series samples that have a positive covariance value associated [49].

3.2 Machine learning concept definitions

A *dataset* is a collection of data used for some specific machine learning purpose, which at least is divided in two separated sets: training and test set. A *training set* is a data set that is used as input to a learning system, which analyses it to learn a model. A *test set* (or evaluation set) is a data set containing data that are used to evaluate the model learned by a learning system.

Attributes (or features) are properties of things, actions or physical magnitudes that we, as humans, use to describe them. An *instance* is an individual object description, which is often represented as a vector of attribute values, each position in the vector corresponding to a unique attribute. Attribute-value pairs are standard way of describing things within the machine learning community. Traditionally, values have come in one of three types: binary, nominal and real.

Before the data can be analysed, they must be organised into an appropriate form. *Data pre-processing* (or preparation) is the process of manipulating and organising the data prior to the analysis. Missing values are a common problem to be resolved in this pre-processing step. *Missing values* is equivalent to unknown attribute values. First, the source of "unknowness" should be investigated; there are several such sources: A value is missing because it was forgotten or lost, a certain attribute is not applicable for a given instance, an attribute value is irrelevant in a given context, for a given observation the designer of a training database does not care about the value of a certain attribute. Strategies to work with missing values: ignore the example with missing values, consider the missing value as an additional regular value, substitute the missing value for matching purposes by a suitable value. Other common process that is applied in the data pre-processing steps is the identification of outliers instances and erroneous observations. The *outliers* are instances which are markedly different from their nearest neighbours. It is important to distinguish them from erroneous observations as one cause of the latter is that missing data are sometimes coded using a specific value. Such values need to be handled as missing values in the analysis and must not be included as observation values when fitting a model to data. Outlying

values that cannot be attributed to some coding should be checked carefully. If they are correct, they are likely to be of particular interest and should not be excluded from the analysis. However, it may be appropriate to consider robust methods of fitting models, which reduce the influence of outliers. Figure 3.3 shows the difference between missing data observations and outliers.

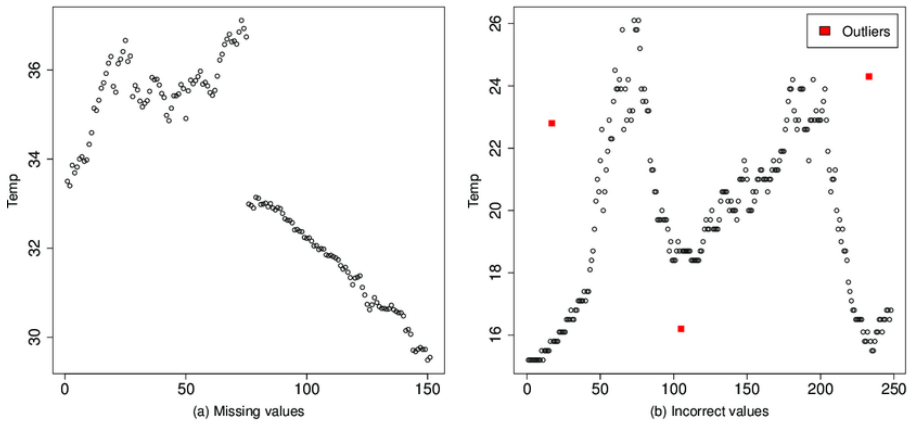


Figure 3.3: Examples showing the difference between missing data and outliers [47].

On one hand, *unsupervised learning* refers to any machine learning process that seeks to learn structure in the absence of either an identified output. *Clustering* is a type of unsupervised learning in which the goal is to partition a set of examples into groups called clusters. Intuitively, the examples withing a cluster are more similar to each other than to examples from other clusters. In order to measure the similarity between examples, clustering algorithms use various distortion or distance measures.

On the other hand, *supervised learning* refers to any machine learning process that learns a function from an input type to an output type using data comprising examples that have both input and output values. Two typical examples of supervised learning are regression and classification learning. In *regression* learning, we are typically interested in inferring a real-valued function (called a regression function) whose values correspond to the mean of a dependent (or response or output) variable conditioned on one or more independent (or input) variables. *Classification* means to put things into categories, group them together in some useful way. A *class* is a collection of things that might reasonably be grouped together. Data are said to suffer the *class imbalance* problem when the class distributions are highly imbalance. In this context, many classification learning algorithms have low predictive accuracy for the infrequent class.

Figure 3.4 shows graphical examples of clustering, classification and regression models.

The *cost* (or loss) of a prediction y', when the correct value y, is a measure of the relative utility of that prediction given that correct value. A common loss function used in classification learning is zero-one loss. Zero-one loss assigns 0 to loss for a correct classification and 1 for a incorrect classification. A common loss function

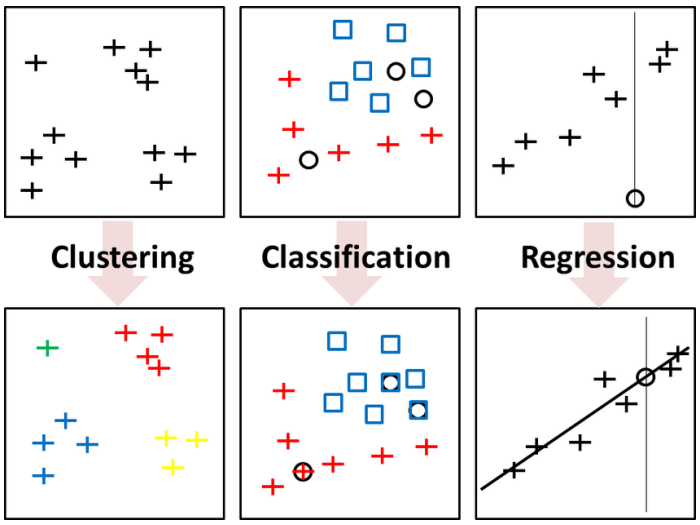


Figure 3.4: The three pillars of learning in data science: clustering, classification and regression [51].

used with regression is error squared. This is the square of the difference between the predicted and true values. In the model training, selected model's parameter values are optimised with the training set in order to minimise the cost function value.

Ensemble learning refers to the procedures employed to train multiple learning machines and combine their outputs, treating the as a "committee" of decision makers. The principle is that the decision for the committee, with individual predictions combined appropriately, should have better overall accuracy, on average, than any individual committee member. Two common strategies used in ensemble learning are bagging and stacking. In *bagging*, each member of the ensemble is constructed from a different training set and the models are combined by a uniform average or vote. However, when *stacking* strategy is applied, a set of models are constructed from bootstrap samples of a dataset, then their outputs on a hold-out dataset are used as input to a "meta"-model. Figure 3.5 shows stacking and bagging techniques architectures diagrams.

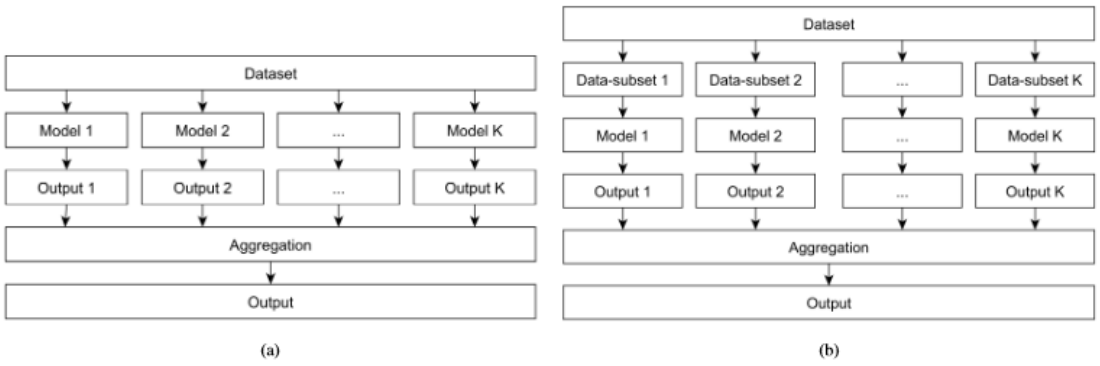


Figure 3.5: Ensemble architecture diagram for (a) stacking and (b) bagging techniques [39].

When it comes to machine learning model evaluation, *accuracy* refers to a measure of the degree to which the predictions of a model match the reality being modeled. For the classification models, the *confusion matrix* summarises the classification performance of a classifier respect to some test data. It is a two-dimensional matrix, indexed in one dimension by the true class of an object and in the other by the class that the classifier assigns. Figure 3.6 shows how the confusion matrix is calculated in a binary classification example, together with the equation of evaluation measures.

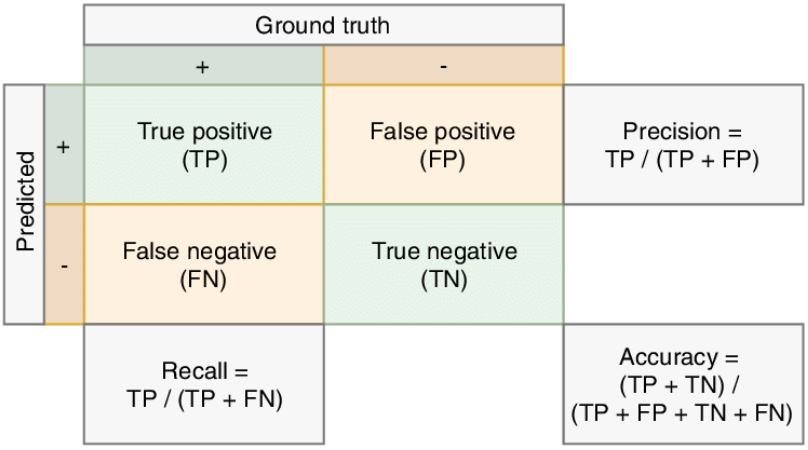


Figure 3.6: Confusion matrix of a binary classification example and evaluation metrics calculation definition [32].

In order to obtain a more robust evaluation of the model that is not so dependent of the initial dataset division, cross-validation strategy can be used. *Cross-validation* is a process for creating a distribution of pairs of training and test sets out of a single data set. In cross validation the data are partitioned into k subsets each called a fold. The folds are usually of approximately the same size. The learning algorithm is the applied k times, each time using the union of all subsets other than the one selected for training set and using the selected as the test set. Figure 3.7 shows the k-fold cross-validation methodology.

Finally, the model evolution should be evaluated after being introduced in the application for which it was generated. *Concept drift* occurs when the values of hidden variables change over time. That is, there is some unknown context for concept learning and when that context changes, the learned concept may no longer be valid and must be updated or relearned.

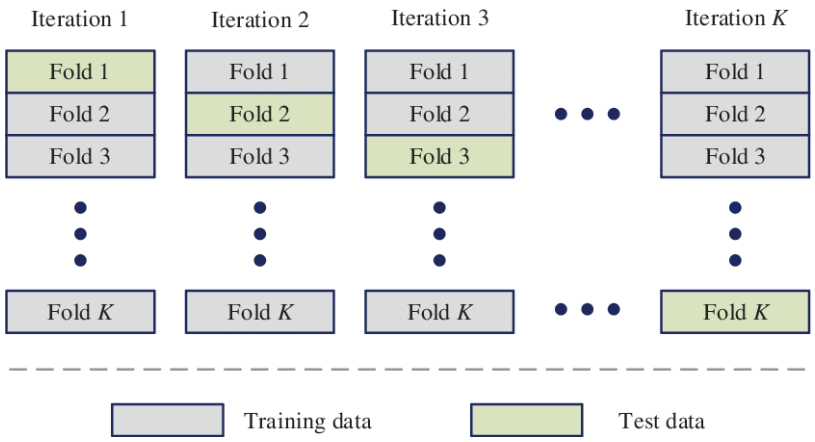


Figure 3.7: K-fold cross-validation method schema [59].

CHAPTER 4

Compression

Dynamic data from fine grained, high temporal resolution sensors is often useful for short-term forecasting and visualization [19]. However, communication latency, bandwidth constraints, high energy consumption and storage requirements for such data can be problematic [6]. Reducing the amount of data to be transmitted can help control latency time and save in energy consumption and storage [5]. A natural solution is to compress the data at the sensing devices, monitoring in real time the error introduced by this process. Point selection is a possible strategy for lossy compression. A key challenge in the setup of point selection methodologies is reducing the size of the transmitted data without sacrificing its quality.

4.1 Background

Blalock et al. [8] describe desirable properties of the compression algorithms:

1. Minimal buffering: on devices with small memory capacities, only small time windows can be used before data is compressed.

2. High decompression speed: decompression of data in order to recover the time series for such as visualization and machine learning applications needs to be quick.

3. Losslessness: using lossless compression algorithms ensure that the data could not depend on previously defined pre-processing strategies.

On the one hand, most work on compressing time series has focused on lossy techniques, i.e., after compression the original data cannot be recovered. Classical approaches for data compression include Fourier transforms [9], wavelets transforms [29], symbolic representation [41] and piecewise regression [19, 40]. Classical compression techniques reduce the volume of data by using transformations, regression

models or aggregations functions. The result of the transformations, the parameters of the regression models or the symbolic representation of the aggregated values are stored to represent the signal. None of the data points measured are transmitted and the signal representation is dependent on the efficacy and adequateness of the compression methodology used.

On the other hand, lossless compresssion techniques [54, 8, 75] help reducing the volume and storage of data to be transmitted without loosing any information and satisfy all the properties above mentioned. However, the compressed version of the signal cannot be used for visualization, control or analytic applications directly, as the encoding of data has been optimized to save storage.

A lossy compression technique is based onS point selection algorithms to reduce the data volume. These techniques aim to select the most significant or representative points. One option apart from selecting points from a regular sampled signal uses adaptive sampling methods [31]. These approaches study the level of variance between the collected data over a certain time frame and dynamically adjust the sampling frequency of the device. Adaptive sampling approaches work well in applications where the collected time series is stationary. In the case of quickly varying data, these approaches often perform poorly.

4.2 Case study

Consider a time series signal which is sampled with a constant frequency. Due to system limitations, for example with respect to memory, not all captured points can be stored, and therefore a data point selection methodology needs to be applied.

One possible classification of point selection algorithms considers the way in which those are applied [11]. On the one hand, algorithms can work in batch mode, i.e., the data is processed when the batch or group is complete. On the other hand, there are algorithms that work in online or streaming mode, in other words, when a new point arrives to the system, the data points selection methodology is applied directly.

Other possibility was proposed by Keogh et al. [35], a classification for data point selection algorithms based on the point selection strategy they adopt. Some of them select observations guaranteeing that the maximum allowed error is met in the time series representation. Other methods, select the best representation of the time series using limited quantity of memory, i.e., $k-1$ segments (or equivalently k points).

Classical data point selection methodologies are based on a maximal error value by point of the time series [78]. Therefore, these strategies work in online mode and depend on a maximum error threshold that would decide if the new received observation is saved or not. Among the classical procedures are the following ones: boxcar algorithm, backward slope methodology and swinging door strategy. The problem of these methodologies is that the compression level is dependent on the signal and as they work in online mode, latency is not controlled either.

The following algorithms work in batch mode. In the case of data points selection using maximum number of segments, a fixed window in time series data is used as a batch, and from there, a maximum number of points k is selected to be part of the

compressed signal.

The simplest way to select data points from a time series with a constant sampling frequency is to pick them using a lower frequency value than the original one, i.e, selecting a point each n points (*oen*).

The different versions of the largest triangle algorithm [71] are based on the use of the effective area of the data points: the significance of a point is indicated by the area of the triangle formed with its two adjacent points. Depending on how the adjacent points are selected or how the buckets are constructed, three different algorithms are generated: largest-triangle-one-bucket (*ltob*) starts from equal size buckets, selecting one point from each bucket regardless of which points are selected in nearby buckets; in the largest-triangle-three-buckets version (*lttb*) similar strategy to *ltob* is used but this time takes into account the point selected in the previous bucket; and in largest-triangle-dynamic, buckets are generated dynamically depending on the linearity of the inner buckets points. The mode-median-bucket (*mmb*) [71] algorithm uses the mode and the median values of data points in each bucket in order to select a point from it. The M4 (*m4a*) strategy was defined by Jugel et al. [34]. First, n buckets are generated containing approximately equal number of points. Then, from each bucket the minimum and maximum values from both axis (time index and data values) are selected.

4.3 Innovation

It is desirable to be able to compress time series from stochastic processes into streams with constant or limited length in order to meet memory capacity limitations. To the best of our knowledge, there has been no reported work on time series compression with rate adaptability and the ability to flexibly preserve different characteristics of interest of a given time series. In this sense, the contributions this thesis puts forward are:

1. The idea of combining different data points selection methodologies taking into account their adequateness in the current signal window.

2. A definition of errors for the determination of the optimal data points selection methodology, in each moment and for different characteristics of interest, depending on the envisaged application

In Figure 4.1 the application of the idea is shown. The compress steps receives as inputs the data to be processed, the maximum number of data points to select from the signal and the definition of the error function to be used to determine the quality of the compression. The smart compress model evaluates the input data and responds with the "optimal" compression method used, the selected observations of the signal and the error obtained in the compression. These three outputs are saved in a data storage, the selected points in order to reduce the data volume, the identifier of the method for decompressing adequately afterwards the data when it is needed, and the error for being capable of monitoring the error generated in this process.

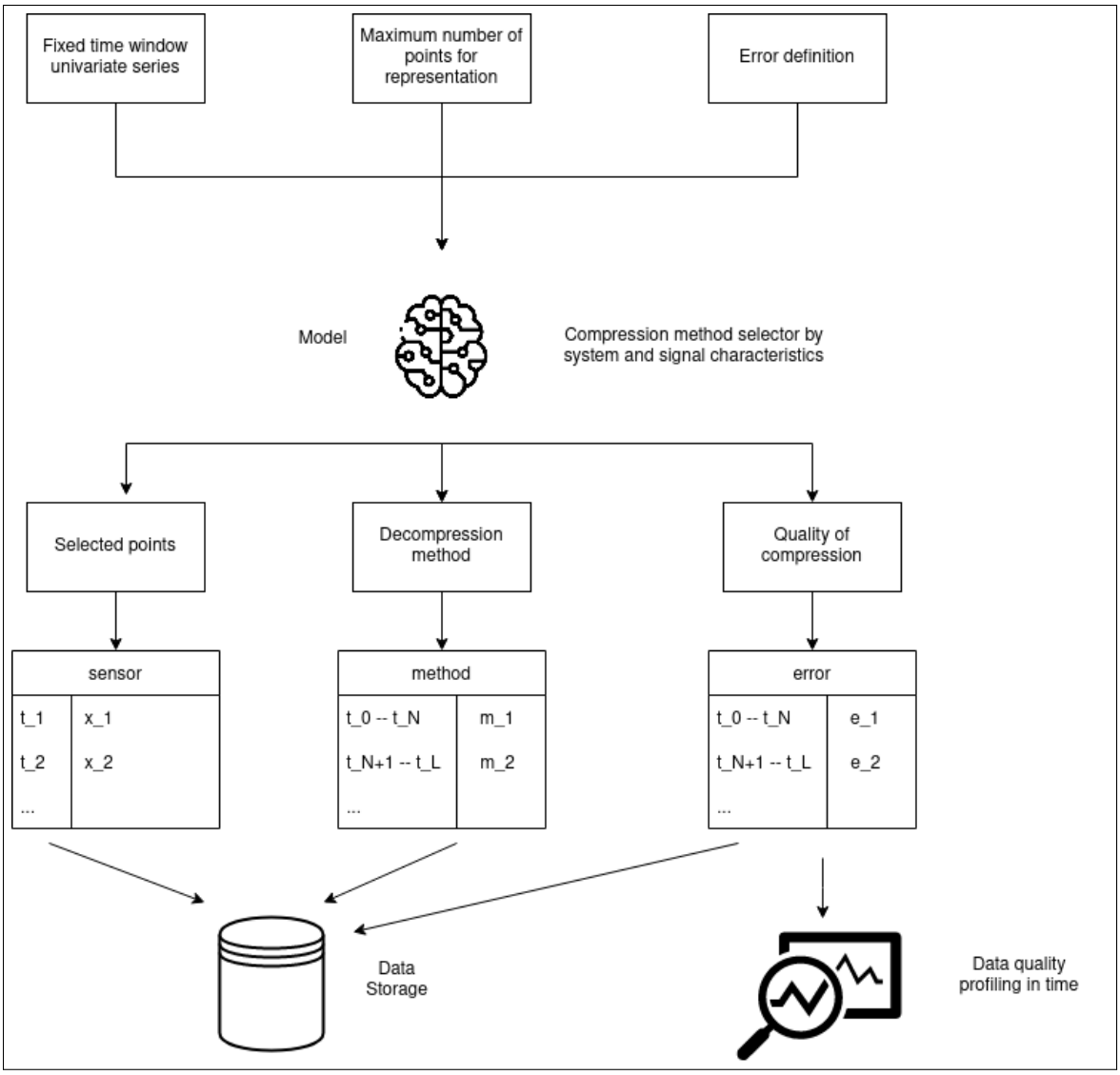


Figure 4.1: Schema of the smart compression method. The input data, the maximum number of data points and the error function are received in the compression system. Then, this inputs are evaluated by the smart compress model and responds with the "optimal" compression method, the selected observations and the error value. Finally, all information is saved and compression quality can be visualised.

Furthermore, this work proposes a smart data selection method based on a optimization process. The aim of this optimization problem is to select the method that minimizes the errors of the point selection process for each feature. Figure 4.2 shows two compression methods applied to a piecewise constant signal.

The general method pursued for generating the smart compress model is shown in Figure 4.3 and can be described as follows:

1. First, the data point selection method is adjusted using training data, in other words, the fit method selects the optimal algorithm that suits best the time

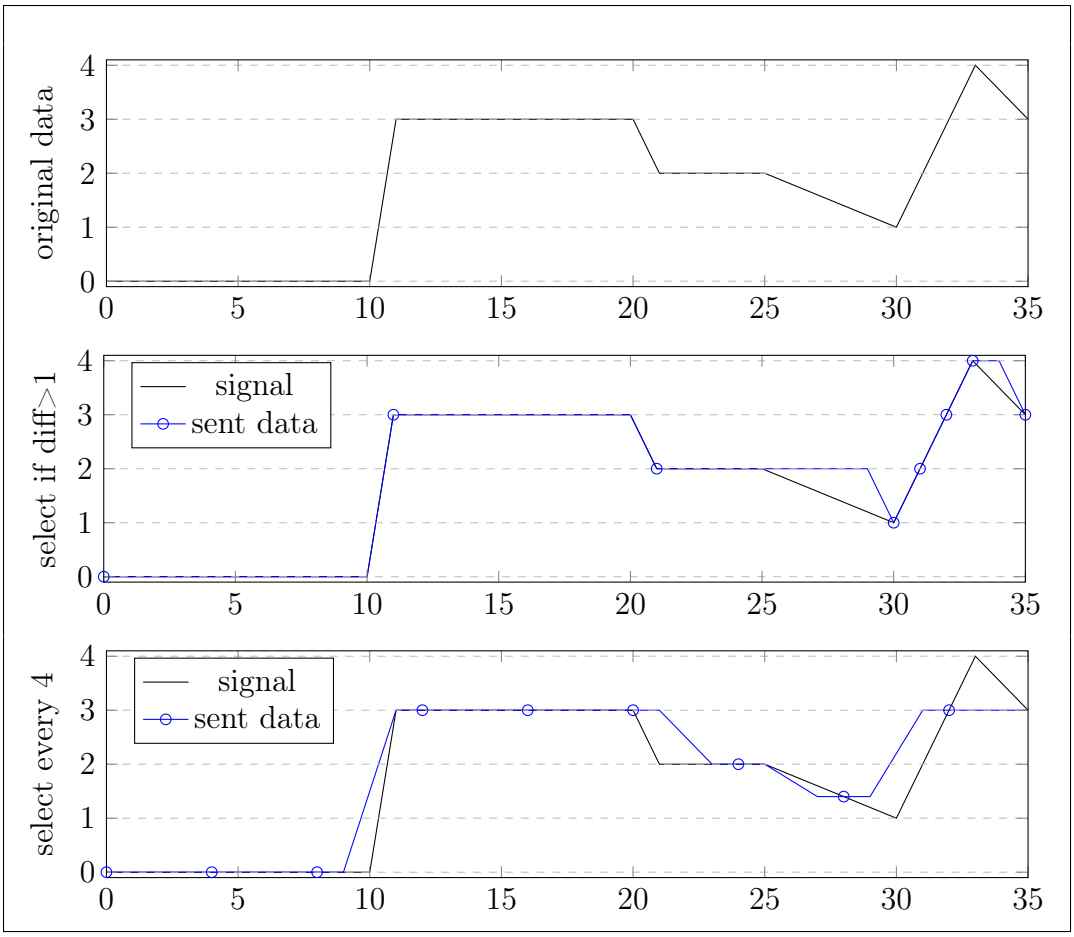


Figure 4.2: Two different compression examples are shown. In the first, a new data observation is saved when the difference with the previous observation is > 1. On the second case, only observations with time indices multiple of four are saved.

series provided in the training.

2. Then, a compressed version of the signal is obtained by applying the proper data selection method to the test data.

3. Finally, the adequateness of the data selection model is validated using the error between the original and the recovered signal from decoding the compressed signal.

In order to be able to compare different data point selection methodologies, the different compressed versions of the time series should have a similar quantity of selected points after each compression strategy is applied to the original data. For this reason, the methods considered in the smart selection algorithm are the ones that guarantee that the compressed version has a well-defined number of points in the compressed version and that can work in batch mode.

Depending on the purpose of the application, properties of interest of the signal could be totally different. The error functions can be defined in order to maintain

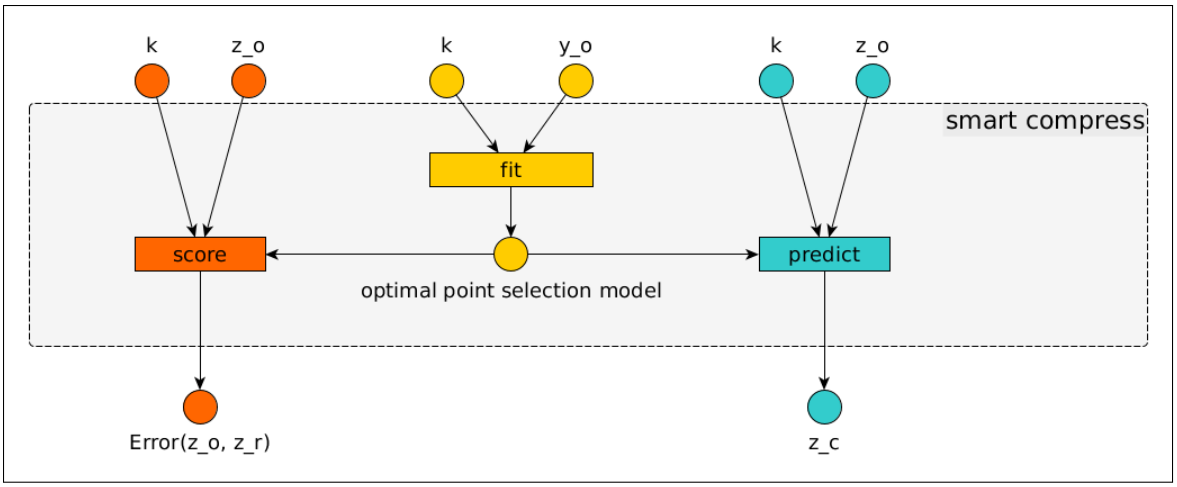


Figure 4.3: Smart compress methodology schema describing input parameters and the outputs of the methods provided.

these properties of the signal. Different signal characteristics are listed next for three different purposes:

- In visualization applications, properties such as shape of the signal, visual outliers, linear trend of data and number of peaks are important to maintain.
- In control applications, the appearance of new events (peaks), change in signal tendency or frequency are important.
- For analytical processes, outliers and statistical properties such as kurtosis, maximum, mean, median, minimum, quantiles, skewness, standard deviation and variation coefficient are essential.

Even if the total number of points in the compressed time series is the same, the distribution of the optimal method can change. In a smaller window sizes, the quantity of points to select from is reduced, and as consequence the algorithm can select them quicker. Furthermore, a smaller window allows adjusting the algorithm to the actual time series characteristics. However, a bigger window size could help distributing the points in time in a smarter way. It is important, therefore, that experiments are made with data from different origins and characteristics in order to ensure that the selection of the algorithm does not only depend on the error function used.

There are two possible strategies to define window size or batch length:

1. Based on a temporal window to schedule the data points selection periodically
2. Based on memory limitations, the compression methodology is planned only when a reduction is needed.

Suppose there is a window of the time series $y_o(t_o)$ where $t_o = [t_{o1}, t_{o2}, ..., t_{om}]$ and being m the number of points in the selected window. Let d be a data points selection

method from the available methods set $D = \{ltob, lttb, m4a, mmb, oen\}$. Then, the compressed version of the time series, $y_c(t_c)$, is defined by the selected data points from y_o corresponding to time indices $t_c = [t_{c1}, t_{c2}, ..., t_{cn}]$. The value $n \leq k$ where k is the maximum allowed number of points in $y_c(t_c)$.

From $y_c(t_c)$ the removed data points values are recovered by the use of a linear interpolation method between available points of $y_c(t_c)$. The notation used to refer to the reconstructed version of the time series is y_r and it is defined for time index values that were contained in the original time series signal t_o.

Finally, an optimization problem is defined to select the most adequate compression method for the signal. This optimization is represented by:

$$\arg\min_{d \in D} E_d(y_o, y_r) \tag{4.1}$$

For evaluation and validation experiments, the considered datasets are the ones available at the UCR Time Series Classification Archive [16]. The Archive contains 128 classification time series datasets of different types including sensor data, simulated data, motion data from several devices and health data such as electrocardiograph (ECG), electrooculography (EOG) and hemodynamic data. Depending on the dataset, either all the time series contained in it have same length, or the length varies between different time series.

For each time series from each dataset, the mean error value between y_r and y_o is used. The 'opt' column presents the mean values in the case of using the optimal method (minimal error) in each the time series. The datasets are sorted in ascending order starting with the dataset with the minimal mean value of the mean of errors of all the time series contained in the dataset. In this particular case, the mean absolute error [12] (MeAE) is used and k value is selected in order to reduce at leas 50% of the data points available in the time series. Due to printable table dimension limitations, datasets have been grouped in 8 different groups (16 datasets per group) in the same order and the sum of mean errors per method are shown in the Table 4.1. The article [25] appended in Part II shows complete results per datasets and with the applications of two different error functions.

Table 4.1: Grouped sum of mean errors of MeAE values obtained from each time series for datasets from URC Time Series Classification Archive.

datasets group	**opt**	**ltob**	**lttb**	**m4a**	**mmb**	**oen**
group 1	**0.377**	0.493	0.386	0.695	0.635	0.657
group 2	**0.853**	1.22	0.863	1.585	1.334	1.29
group 3	**1.508**	2.026	1.526	2.79	2.243	2.203
group 4	**2.74**	3.821	2.892	4.15	4.192	4.052
group 5	**4.493**	6.016	4.674	6.506	6.956	7.134
group 6	**7.055**	9.632	7.306	10.278	10.996	10.795
group 7	**10.384**	14.267	10.698	17.227	17.429	17.093
group 8	**59.529**	89.281	66.442	75.056	75.641	87.967
total sum	**86.939**	**126.756**	**94.787**	**118.287**	**119.426**	**131.191**

In general, the *lttb* method when using the MeAE is the most adequate method when different datasets are grouped. Moreover, when the capabilities for the representation of a complex signal are limited due to the maximum number of selected points allowed in the compression, selecting the optimal compression strategy adapted to the specific properties of the window is crucial. This is shown in Table 4.1 as the difference between choosing the optimal method in each window each time (column *opt* of the table) or using the same method for all the dataset (rest of the columns). The total sum of the grouped MeAE using the optimal point selection method in each time series is 86.939 that has nearly eight point difference compared to globally optimal methodology *lttb* value 94.787. Furthermore, this difference becomes much bigger when if another algorithm from the set D apart from *lttb* is selected, with value ranges from 31 to 44 points.

4.4 Conclusions and future work

When the selected points are not capable of representing the original signal with desired precision, selecting the optimal points in each compressed window is essential. Each downsampling method considered has its own characteristics and adaptability depending on the time series properties. In case of having a variable that has a stationary or similar statistical properties in the complete time domain, it is possible to select an unique optimal downsampling method that suits adequately the compression of the signal. However, this is not the usual case in industrial real sensor data as production processes vary with time and therefore variable properties also change with time.

The experiments carried out show that a methodology to adaptively select or / and monitor the point selection strategy is needed. Not all the time windows of a certain time series can be compressed equally, and this is even more important when it comes to real sensor data where context variables vary with time. A change in the production process, the dependency on other internal or external variables, or even noise can affect the point selection method. Controlling the error values is critical in order to detect a point selection model drift and, in consequence, retrain the point selection strategy to maintain point selection quality.

Considering the need to adaptively compress time series from stochastic processes into streams with constant or limited length in order to meet memory capacity limitation. The results have put forward and demonstrated in a practical implementation the idea of combining different data point selection methodologies using their potential in the current signal window, while providing a definition of errors for the determination of the optimal data points selection methodology. In each moment, and for different characteristics of interest relative depending on the envisaged application.

Future work to be considered includes combining algorithms to select points using a maximum allowed error value, in the signal representation in windows where maximum memory limitation per window is satisfied, together with methodologies that use maximum number of segments when memory limits are exceeded. With these combinations, it is possible to work with a trade off between the maximum error

allowed in the signal representation and the memory limitations due to the system properties. Furthermore, it should be possible to select points for multiple time series at once, for the cases in which they need to be synchronized for their use in an application such as visualisation [21]. Finally, controlling the window size and quantity of data points to be selected depending on the characteristics of the time series can prove beneficial in a number of application scenarios [36].

CHAPTER 5

Joining

For machine learning applications, an alignment between features is needed. Each feature should be resampled to obtain a common desired temporal reference system previous to any feature extraction / selection algorithm application. Depending on the feature and the application system, the optimal joining method could be different.

The sampling frequency of a signal is determined by the nature of the variable itself and the capabilities of the infrastructure and storage of the system where it is collected. Furthermore, if the time series data is collected by IIoT devices, the variables tend to have irregular sampling rate. When it comes to analysing the collected data from multiple devices, data should be combined and analysed together.

In the pre-processing steps of time series analytics by machine learning, one of the main steps often consists in joining all the features that will be used in model generation joined in an equally-sampled table. The specific case of working with time series has the advantage of the use of a temporal reference system, a timeline that enables merging the observations. However, often each feature has its own sampling, and all of them should be resampled to construct a synchronized single multi-variate time series.

5.1 Background

Very often, all the time series which are needed for a certain application are not preintegrated in a unique table; data is extracted from multiple sources, transformed and combined during query run-time. The effort needed for data cleaning during extraction and integration increases query response times, but it is mandatory in order to achieve useful query results. A task that often proceeds record matching is that of schema matching: the task of aligning attributes from different schemes.

Each attribute has its own sampling time, which can be uniform or varying in time. To integrate all the time series in a single table, it is necessary to resample each

of the variables in order to have a common time index in all the variables needed in the system. Resampling time series can be problematic, specially when dealing with streaming data. First of all, the selection of an appropriate resampling frequency is always something to be determined by subject matter experts. Once the frequency is set, for each new timestamp observation in the resampled set, there is no previously defined strategy to select which value to assign to it, the last recorded value, the next one or the closest observation. Furthermore, if several values were recorded between two new timestamps, the values in between could be aggregated or could be lost in the way.

Suppose that the system captures and stores streaming sensor-based data and in order to maintain a reduced number of registration entries, each sensor registers a new observation in the database only when there is a significant change in data values. The decision of the significance of the difference between data points is based on the scale of each feature. The aim of this data recording strategy is reducing data volume. Consequently, if a feature becomes unstable the writing frequency increases drastically.

5.2 Case study

In the context of Structured Query Language (SQL) database engines [15], a time series is a sequence of data values measured at successive, though not necessarily regular, points in time. Each entry in a time series is called an observation. Each observation comprises a timetick that indicates when the observation was made, and the data recorded for that observation. The set of timeticks defines a temporal base or temporal reference system for the series.

A temporal join is a join operation that operates on time series data. It produces a single array time series based on the original input data and the new temporal reference system. The most common joining strategies are described next.

Left Join The left join method takes only samples from y that are synchronized with t_D, in other words, only data that has originally the desired time is used.

Nearest Join A Nearest Join takes into account the nearest known available data from y. Depending on the distribution of y, future knowledge of future data can be added to the past in a non-causal manner.

Forward join In a forward join, samples of t_D that are not available in t_O are selected using subsequent matches from y.

Backward join In a backward join, samples of t_D that are not available in t_O are selected using the nearest prior match.

Outer join In an outer join, samples of t_D and t_O are combined and sorted. Missing values can be filled before selecting only the observations from t_D for the resulting table.

This section introduces a range of common SQL joining methodologies. Specifically, the following methods will be introduced: left join, nearest join, forward join and backward join. Notably, the outer join is not contemplated, as the obtained results are the same as those from other selected methods (backward join or forward join) depending on the selection of a function for filling in Non-Available (NA) values (propagating the last valid observation (*ffill*) or using next valid observation (*bfill*)).

An example on the possible effect of the selection of the joining method for a simple time series is shown in Figure 5.1. Depending on the selected joining strategy, future values are used for filling undetermined observations (as shown in the forward case) or multiple missing values can be obtained in the joined time series version (shown in the left case). This example demonstrates the importance of selecting adaptively the joining method and the necessity of monitoring the pre-processing steps to ensure, or at least being aware of, the propagation of errors that can affect the system.

Given the above existing methods, the remanding of this section considers the problem of locally selecting a optimal method by the optimization of a quantitative measure of the quality of the obtained joined series.

5.3 Innovation

The resampling step is done separately by feature, and, depending on the application objectives, different characteristics of data joining methods should be taken into account. Examples of objective measures of pre-processing quality are based on measures of distortion and on the number of unused observations in the process. Further considerations might be related to the causal nature of the resulting system, or the amount of delay or anticipation applied to the different original time series to synchronize them to the desired time sampling. The main idea of the application is described in Figure 5.2.

First of all, the user indicates the specific feature selected for the resapmling and the desired time index distribution for the joining. Apart from that, the system or application provides the properties or limitations that should be taken into account in the resampling. For example, these properties could limit the use of future time index values in the joining process. Later, with these inputs, the "smart join" model recommends the selected optimal joining method, and gives information about the quality of the join provided in this case. With both results, the user applies the joining method, obtaining the desired time distribution for the feature, and is capable of monitoring the error introduced in the pre-processing step.

The procedure pursued for defining the smart join model is described next. This smart join method is generated based on an optimization process. The aim of this optimization problem is to select the method that minimizes the errors of the sampling process for each feature. The general concept of the methodology of the smart join is shown in Figure 5.3 and explained as follows:

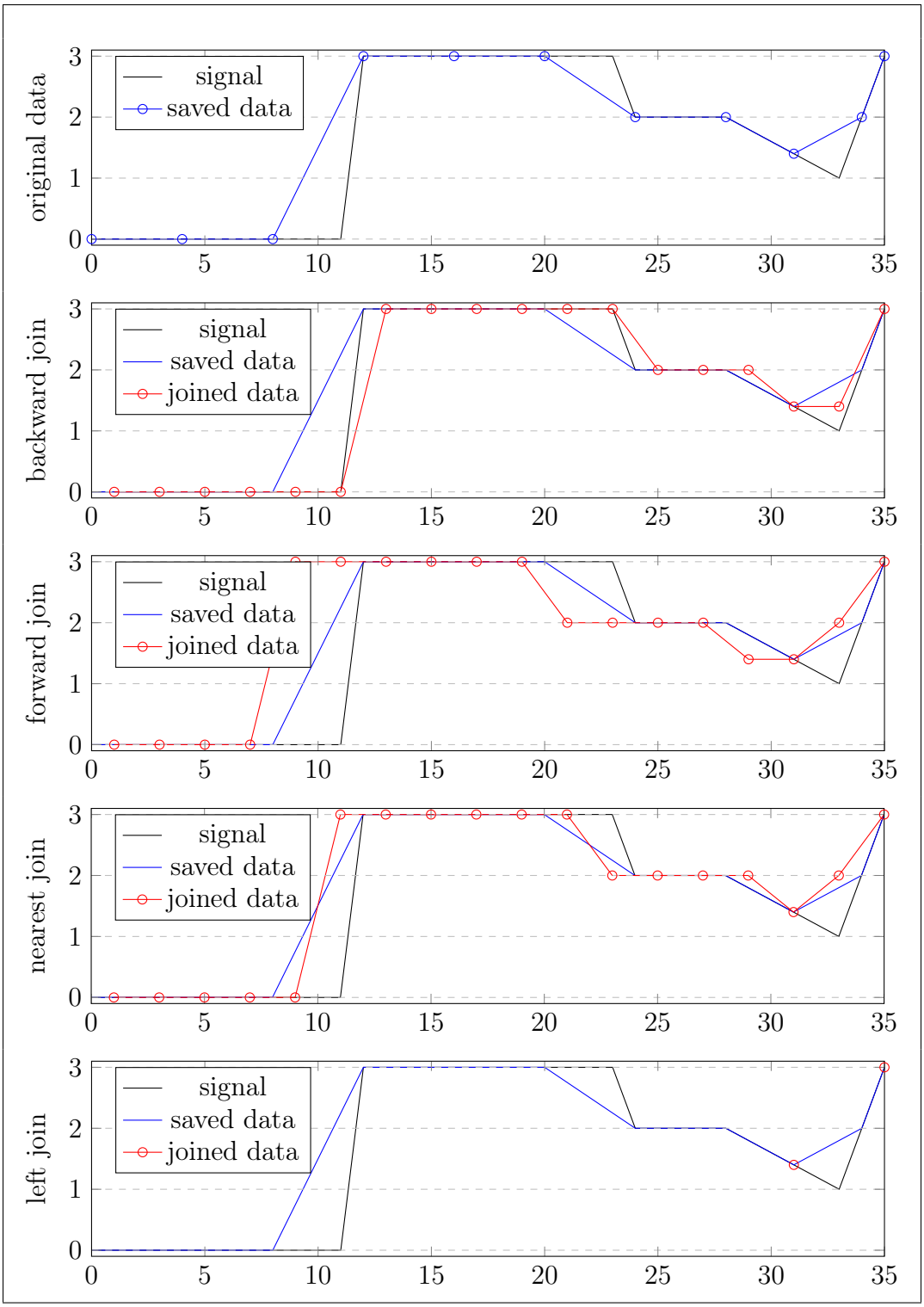


Figure 5.1: The most common joining strategies are applied to a piecewise constant function. In each case the obtained observation values are indicated, as well as the interpolated curve in order to compare the distortion with the original signal.

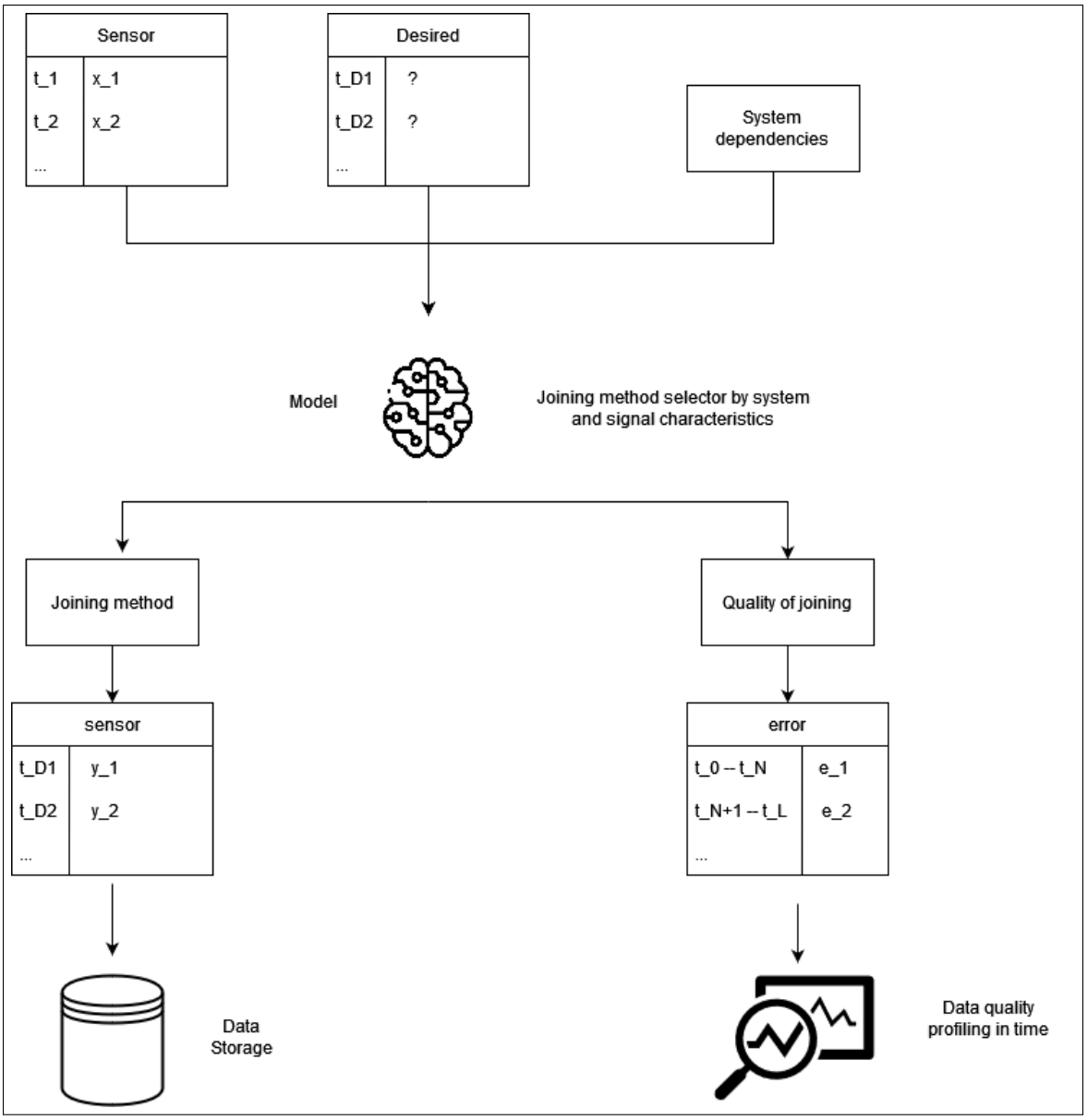


Figure 5.2: Smart joining application schema description. The original observations, the desired time sampling and the system dependencies are received as the inputs of the smart joining model. Then, the model process them and provides as output the signal with the desired sampling and the joining strategy used for it.

1. First, the joining model is fitted using training data. This needs to be done for each feature separately.
2. Then, resampled data is predicted by applying the selected join method to the test data.
3. Finally, the model is validated using resampling error.

Suppose we have a time series slice y of the selected feature that needs to be

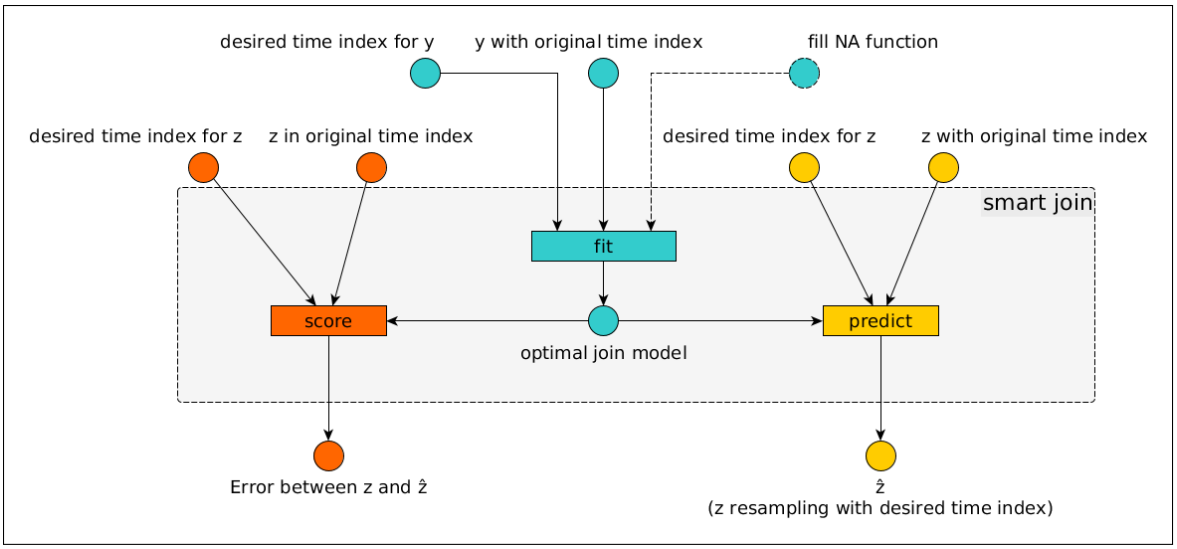


Figure 5.3: Smart join methodology schema describing input parameters and the outputs of the methods provided.

resampled to be joined with a desired time index. First of all, the *fit* method is used in order to obtain the "optimal" join method. The inputs needed for the join are the original time series slice (y with the original time index) and the desired time index. Another optional parameter can be a fill NA function as it can affect selecting the "optimal" method. Then, another slice of the same feature (z) is used for the testing by the use of the method *score*. Finally, the optimal joined strategy determined is used for resampling other time slices of the features with the *predict* method.

The fitting process to find an optimal joining model can be mathematically represented as follows:

Suppose we have the time series $y(t_O)$ where $t_O = [t_{O1}, t_{O2}, \ldots, t_{Om}]$ is the initial temporal reference system. Let j be a join method from the available methods set J ($left$, $backward$, $forward$, $nearest$). We need to obtain a new time series $\hat{y}(t_D; j, y)$ with the desired temporal reference system $t_D = [t_{D1}, t_{D2}, \ldots, t_{Dn}]$. The smart join algorithm aims to find the optimal join method $j \in J = \{left, backward, forward, nearest\}$ that minimizes an error function $E(y, \hat{y})$. The parameters for applying the smart join method are the function meant to fill unavailable measurement values $f \in F = \{None, bfill, ffill, nearest\}$. In case of not being specified, default values will be used (in which case $f = None$). The possible values of the imputation function f are $None$ (not filling NA values), $bfill$ (using subsequent value that is nearest) and $ffill$ (using prior value that is nearest). The optimization problem is defined as:

$$\arg\min_{j \in J} E(y, \hat{y}) \tag{5.1}$$

The current section introduces an illustrative example of the application of the proposed method to a dataset from a simple piecewise function.

Suppose that the piecewise function is sampled irregularly in order to save memory applying two criteria:

- The system checks every minute if the value of the data point has changed enough according to a pre-established criterion (in this particular case, a difference with the prior data point higher than 0.5) to save that data point.
- Every minute the system also checks the difference in time with the last saved data point and, if this difference is greater than or equal to four minutes, it saves the last available data point.

As an illustrative example, the methodology described above is applied to an piecewise function. The original piecewise function and the saved data using these criteria are shown in Figure 5.1. Suppose that the desired time reference system corresponds to $t_d = \{1, 3, 5, \ldots, 33\}$. Results after the application of different joining methods are shown in Figure 5.1. For example, a magnitude indicating the distortion committed due to the need of a joined data with synchronized temporal reference system can be used as error function. Specifically, $\mathit{Diff}(y, \hat{y})$ calculates the difference between the two time series (original and resampled).

$$\mathit{Diff}(y, \hat{y}) = \frac{\text{mean}(\text{abs}(y_{inter} - \hat{y}_{inter}))}{\max(y) - \min(y)}, \tag{5.2}$$

where y_{inter} and $\hat{y}_{inter}$ are obtained by means of linear interpolation of time series y and $\hat{y}$ respectively for time values in $t_O \bigcup t_D$. The value of this error is shown in the Table 5.1.

Table 5.1: Error values for the different join methods in the application example.

Method	**Diff$(y, \hat{y})$**
left	1.0
nearest	0.045
forward	0.092
backward	0.084

In the article [24] (appended in Part II) experimental results with synthetic simulated data and a real industrial application are shown. In addition, an error function is proposed especially designed taking into account time series properties.

5.4 Conclusions and future work

This section introduced the definition of an optimization problem for data pre-processing, and in particular for data joining process that imply a need for data resampling. In the experiments, only SQL methodologies are considered with basic NA filling strategies and the data joining is dependent of the desired time index and the window considered for evaluating the joining. Even considering the previously mentioned details, the proposed approach shows the capability to select the best fitting joining strategy and the necessity of using monitoring tools to control propagated errors due to pre-processing steps.

The approach presented in this section has several new paths to follow as future work. On the one hand, the approach could be improved, adding automatic selection of the time window size, or applying imputation strategies for missing observations. On the other hand, the benefits of the proposed Smart Join method should be quantified on a diverse range of real world applications.

CHAPTER 6

Imputation

Missing observations in time series data can be produced by multiple causes such as network communication problems, measurement errors or data acquisition problems. In general, the cause of a missing observation is unknown and its appearance is unpredictable.
The data quality and availability are essential for applications where decisions are taken based on data [55, 45, 17]. Depending on the relative amount of missing data and the cause(s) generating the missingness, the imputed data can be biased, severely affecting performance of algorithms, machine learning models, etc [28].

6.1 Background

Zhang [86] presents a categorization of imputation methods based on the number of imputations generated for each missing observation:

- **Single imputation.** This method provides a unique estimation for each missing value [69].
- **Multiple imputation.** This strategy makes multiple estimation values for each missing observation providing uncertainty measures in each case in order to combine them and reach a final imputation value [61].
- **Fractional imputation.** In this case, several imputed values are generated together with conditional probabilities given a known observation [82].
- **Iterative imputation.** These techniques are capable of estimating missing values when data points are incomplete by the use of multiple iterations obtaining refinement in the estimation values [43].

The rest of the section is centered in the description of different single imputation techniques as the research presented in this chapter is focused on their use.

The simplest imputation method is replacing missing observations with a constant value. This value could indicate that the data is missing, or be replaced by a statistical property such as mean, median or nearest, previous or following data points.

Interpolation methods [70] consist of constructing new observations from the information provided from a given amount of data points. Linear interpolation, and its generalization polynomial interpolation, are commonly used due to their simplicity. In the case of needing smooth functions, spline interpolation can be used [48]. Akima splines [3] are capable to adjust smooth functions near outliers without creating oscillations due to its non-linearity interpolation function. Kriging [53] is a geostatistical interpolation method that is based on a regression method of Gaussian processes that can be used for point estimation.

In the cases when sufficient historical data is available, time series forecasting models can be generated. Once the model is trained, it is possible to forecast the missing observations and use those predictions to fill the gaps. For this purpose, classical statistical models can be used [23, 13, 66]. Additionally, machine learning models [46, 83, 10] have been applied with success in time series forecasting.

Another possibility is using pattern recognition methods [80, 72] to identify similar patterns (or neighbours) in the historical data using observations previous to missing values to fill the gaps. In addition, clustering techniques [2, 50] are capable of grouping similar signals that can later be used for imputing missing data.

6.2 Case study

The case study considered is centered in univariate time series data with missing values with unknown origin. As univariate time series are considered, there is no relation to other time series that can provide information about the actual missing observations. There are three possible types of missing data [60, 42]:

- Missing completely at random (MCAR) describes the situation when missing data occurs entirely at random, all data points in the time series have the same probability of becoming missing data. In other words, the missing observation cannot depend on the value of the data point itself, neither on the value of another variable.

- If the data is missing at random (MAR), the missing probability for an observation is independent of the value of the observation itself, but it depends on the value of other variables. In the time series univariate case, the values of the time can be used as the other variable.

- Missing not at random (MNAR) denotes the case when the probability o an observation being missing depends on the value of the observation itself.

In Figure 6.1 an example of each type of amputation strategy is shown.

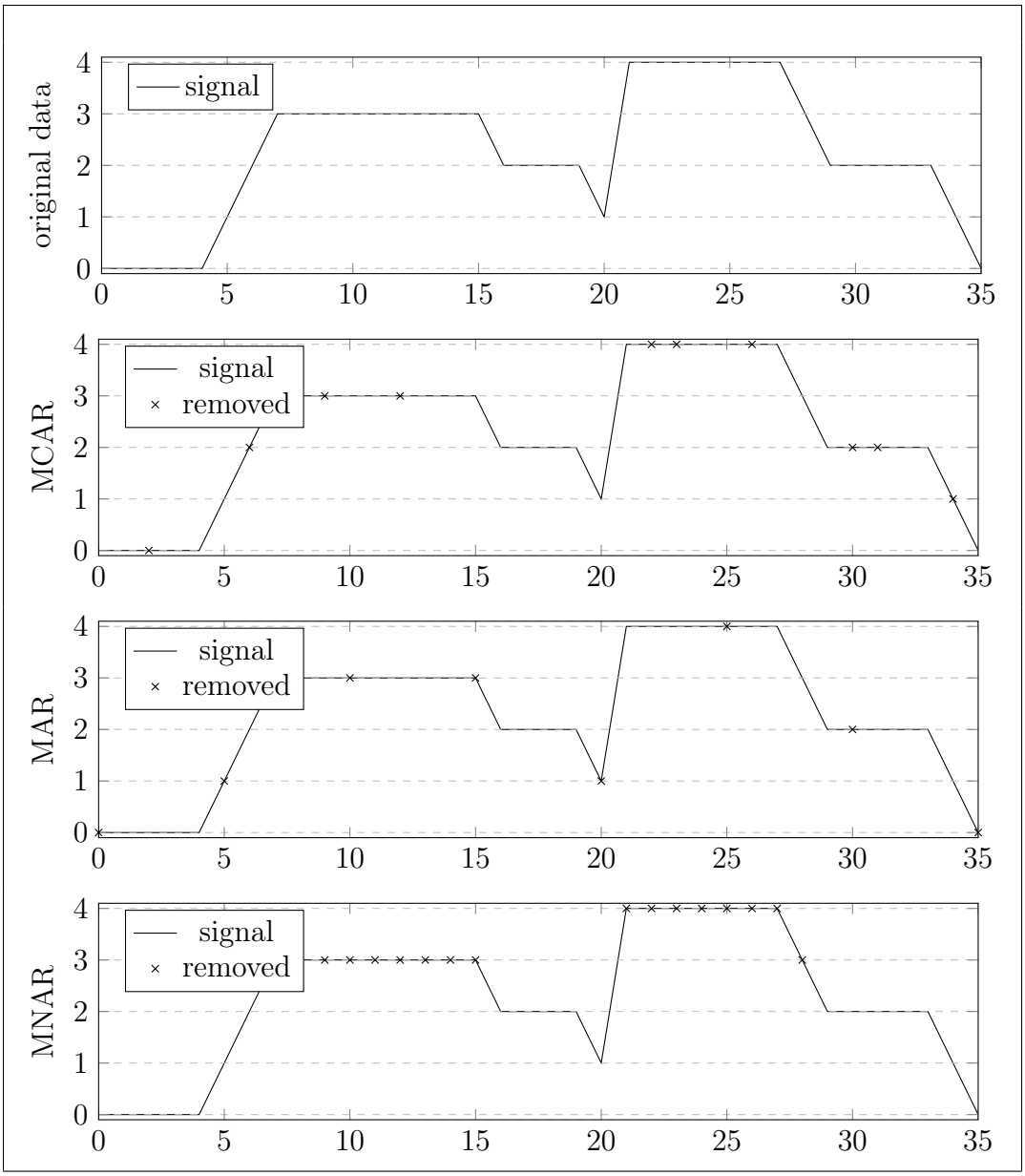


Figure 6.1: Three types of amputation strategies are applied to a piecewise constant example. In the MCAR case, 10 points are removed randomly. For MAR amputation strategy, the points with time index multiple of five are removed. Finally, for MNAR amputation method, the observations with values ≥ 3 are removed.

In real cases, the cause generating missing values is unknown, and can be a combination of multiple types described previously. Due to the lack of knowledge of the cause that has generated them, there is no defined "optimal" imputation strategy. In this thesis, imputation process is treated as a pre-processing step that should be controlled and optimized during data cleaning process and previous to any machine learning or similar type of application.

Multiple imputation strategies have been proposed in the latest years [52, 81,

7, 79]. However, as most of these methods are specifically designed for data with certain given characteristics of specific application fields, undesired results can be obtained when applied to real data or previously unconsidered situations [18, 64]. Therefore, as data in warehouses combines multiple input sources, each one with its own characteristics, and since the imputation methods should give support to multiple applications with different properties and needs, a combination of multiple imputation strategies is proposed.

The general idea is to work in a window-by-window basis, selecting in each case the imputation strategy that suits the situation best according to available signal information, missing data characteristics and known context (such as specific time series data properties or availability of historical data that contain similar patterns). The considered imputation methods between the ones to select from is dependent on the system properties such as context information and admissible time latency to impute.

6.3 Innovation

The work presented in this dissertation proposes a method for adaptively identifying an optimal data imputation method for a specific time series on a window-by-window basis. Thus, this contribution focuses on quantifying the effect of the application of different imputation algorithms to a time series depending on the characterization of the observed data and gap distribution characteristics. The approach consists of training a machine learning model that is able to predict, from signal and gap characterization, which imputation method to use in each case.

The model is trained simulating missing values in available historical data. The missing data is generated by amputing historical data, considering and combining the three types of missing values described above, and with different missing percentages. In this way, it is possible to optimise which imputation strategies work best in each case. It is necessary to amputate the time series, as this is the only way to guarantee with certainty the error made by the imputation algorithm.

First, a gap is defined as continuous rank of missing observations. For each gap their position in time, length, and known previous and following observations are used as descriptors. Then, for describing the signal values, statistical descriptors and signal processing features such as coefficient of variation, shape factor or mobility are proposed. Finally, imputation methods used in the optimisation and the error value is determined by the system limitations and application necessities. This methodology is described in Figure 6.2

The methodology used for generating the model for selecting the imputation method is shown in Figure 6.3 and detailed next:

1. First, historical data is divided in training and test sets. Each time series window is amputed by a strategy defined previously and with a fixed missing percentage.

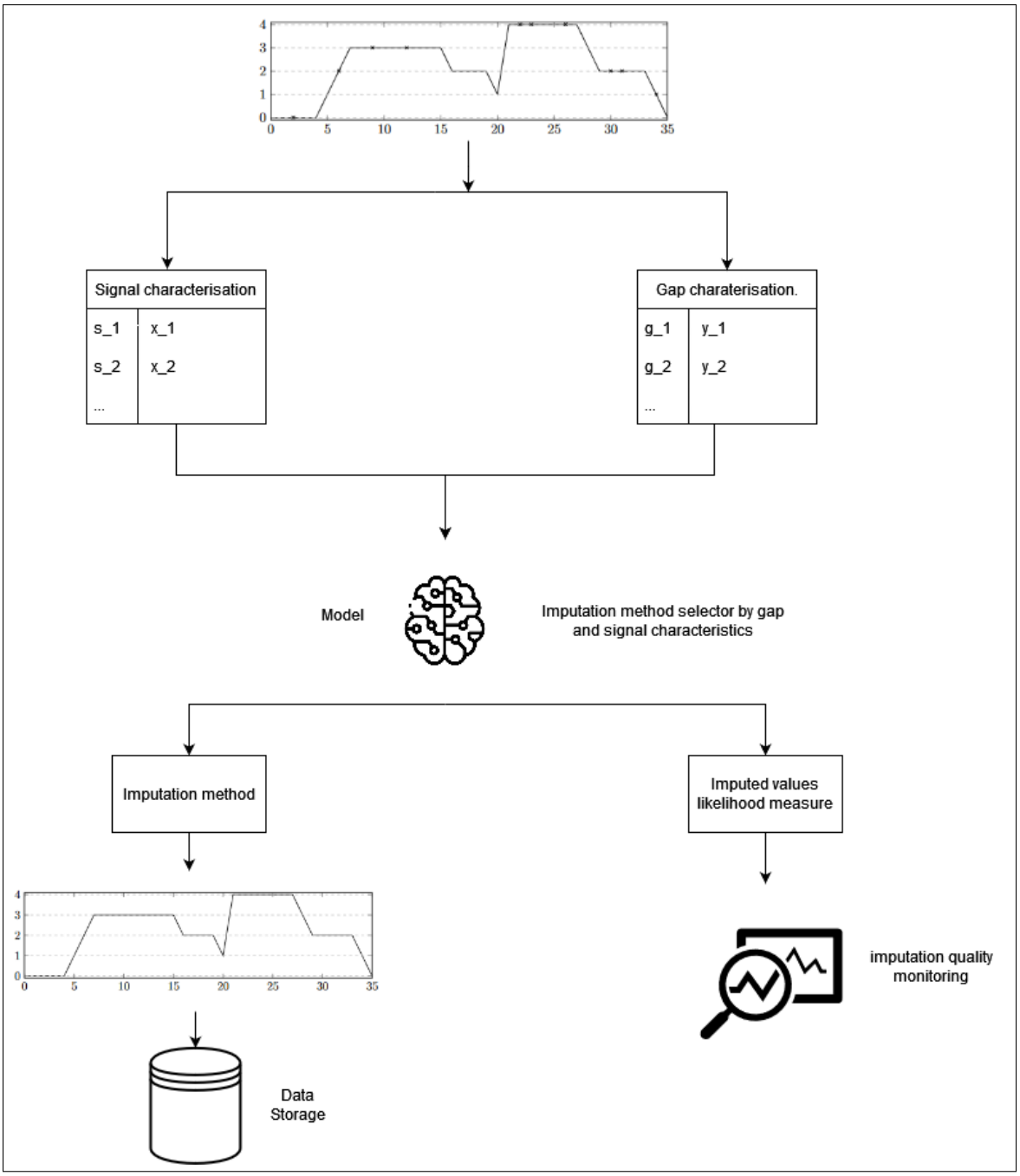


Figure 6.2: Imputation strategy application schema. The signal with the missing values is received and the signal and the gaps are characterised. The imputation method processes both characterisation results and the signal with the imputed observations are received in the output.

2. Then, each amputed time series is characterised and imputed by all available imputation method separately.

3. The error generated by each imputation strategy is calculated comparing im-

puted result with the original time series. The imputed method with minimal error value is determined as optimal for reconstructing the missing observations in this specific case.

4. The dataset for training the classification model is generated by the features obtained in the characterisation of each time series, together with the optimal imputation strategy as target value. Once the model is trained, the validation is pursued by the test set constructed similarly.

5. Finally, this classifier can be used when a new time series with missing observations arrives to the system in order to decide which imputation method from the available ones fits better.

6. Using statistical distribution descriptors of the historical data, classification model's uncertainty and its properties, the obtained results' evaluation is proposed. These results could be used for monitoring imputation quality, classifier's capability and detect when a model drift occurs or retraining is needed.

As an illustrative example, Figure 6.4 shows two curves with multiple amputation strategies applied to them. Depending on the distribution of the missing observations and signal characteristics, the confidence interval provided by Krige [38] in the missing data zones varies notably.

With this in mind, the proposed methodology aims to learn the optimal imputation strategies starting from the gap distribution and signal characterisation. In larger gaps the uncertainty is greater as distance between known observations is greater and therefore the importance of the known values used for imputation is reduced. Not only the distance between the known values and missing observations is decisive for imputation, the certainty of the results depends on signal properties such as linearity, seasonality or complexity. In Figure 6.5 multiple imputation methods are applied to three different amputed versions of two examples where the previously commented properties are visible.

In the article [26] (appended in Part II) an unique imputation methodology selection model is presented that provides an adaptative imputation strategy for the 128 classification time series datasets of the UCR Time Series Classification Archive [16].

6.4 Conclusions and future work

A general data imputation method is proposed that is not dependent of the data neither of the missing observation causes. This characteristics ensure that the strategy pursued is applicable in different domains, and that the imputation can be applied when the origin of the missingness is unknown. For that purpose, gap distributions and time series characterizations are provided in order to learn the the best suitable imputation methods in each case. The proposed methodology identifies an optimal imputation strategy for univariate time series in a window-by-window basis. This allows applying the proposed strategy in parallel for each time series and in user defined batch sizes.

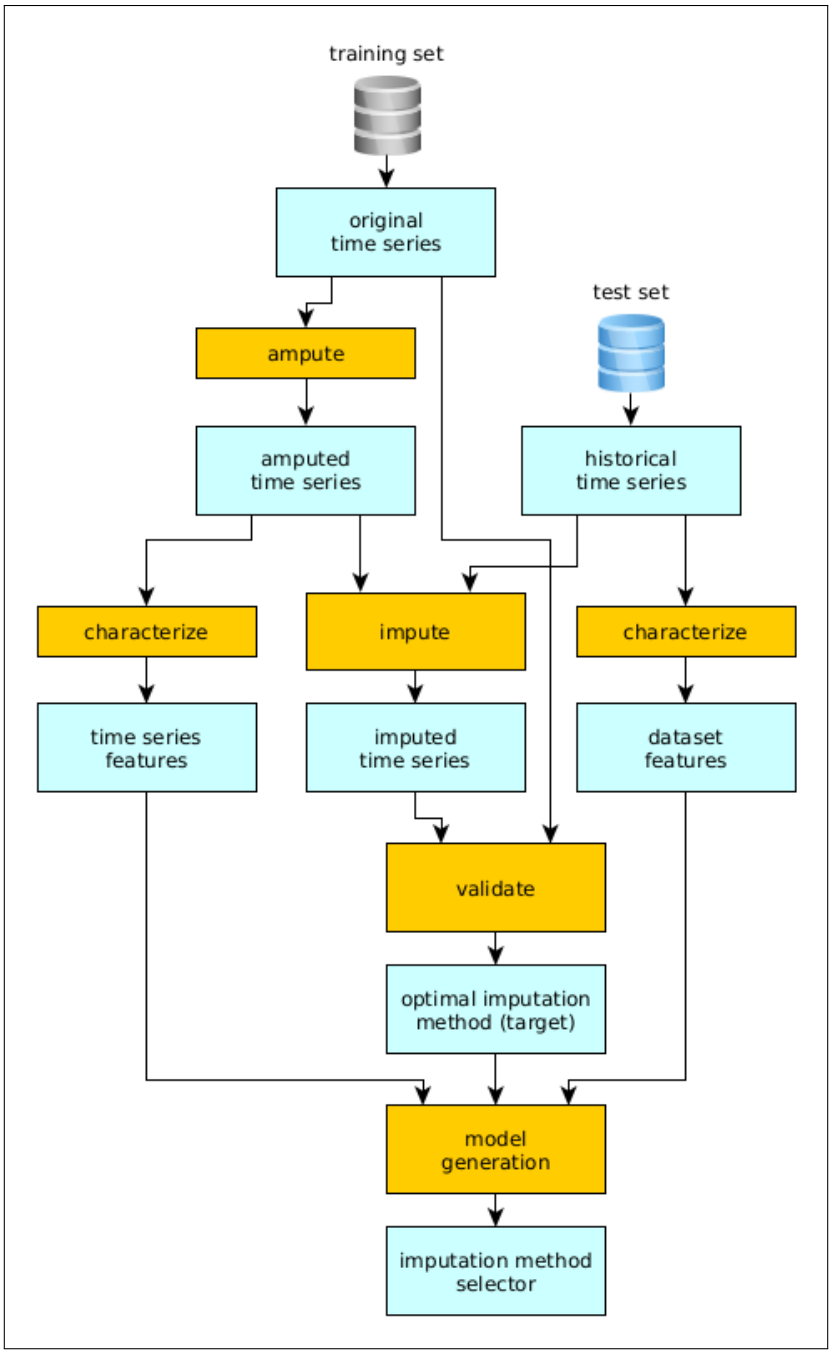


Figure 6.3: Imputation model generation description. First, historical data is amputed and characterised. Then, multiple imputation methods are applied and the resulting errors are calculated in each case. Finally, a machine learning model is trained using as input the characterisation features and as target the optimal imputation strategy obtained in the previous step.

Future work to be considered includes adding a risk assessment method at the end of the imputation process. In this step, the quality of the imputation method recommended can be evaluated before saving the data. For the suggested evaluation

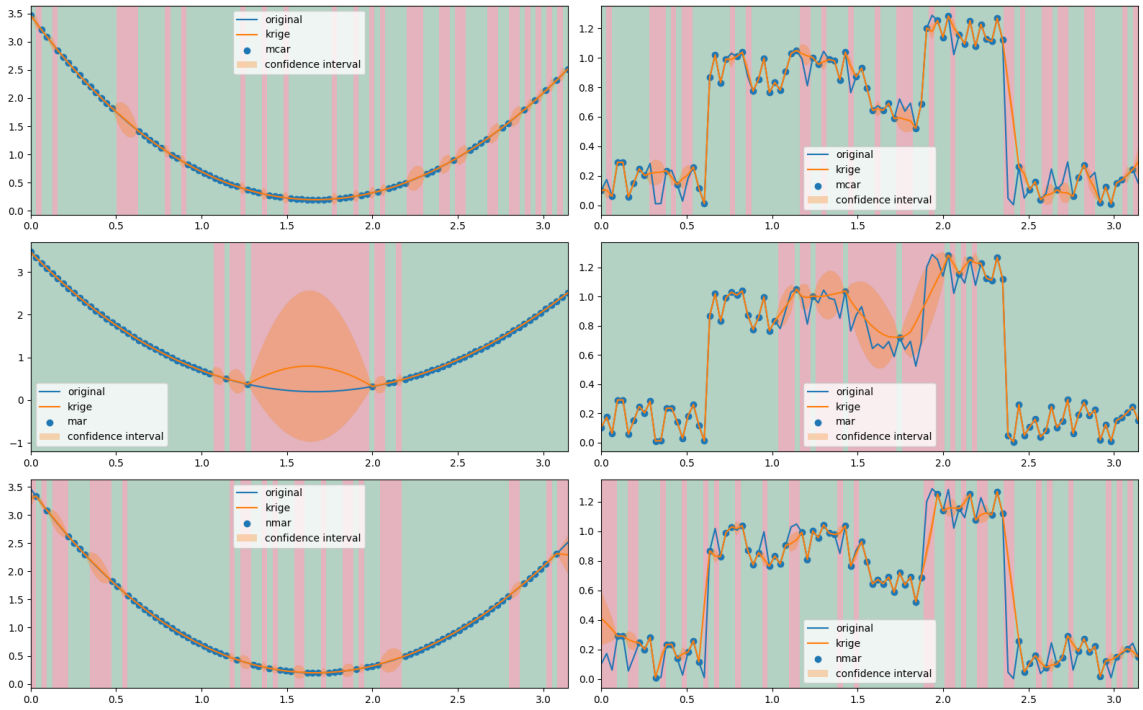


Figure 6.4: Two examples with different amputation strategies applied to them. In each of the examples, the original signal, the missing observations zone (red shade) and the confidence interval (calculated using Krige methodology [53]) for missing values is marked.

dataset, general historical characteristics can be used and monitored in time to detect drifts in data. Finally, the same data characteristics can be used to consider only a subset of imputation methods from the available ones in the training phase.

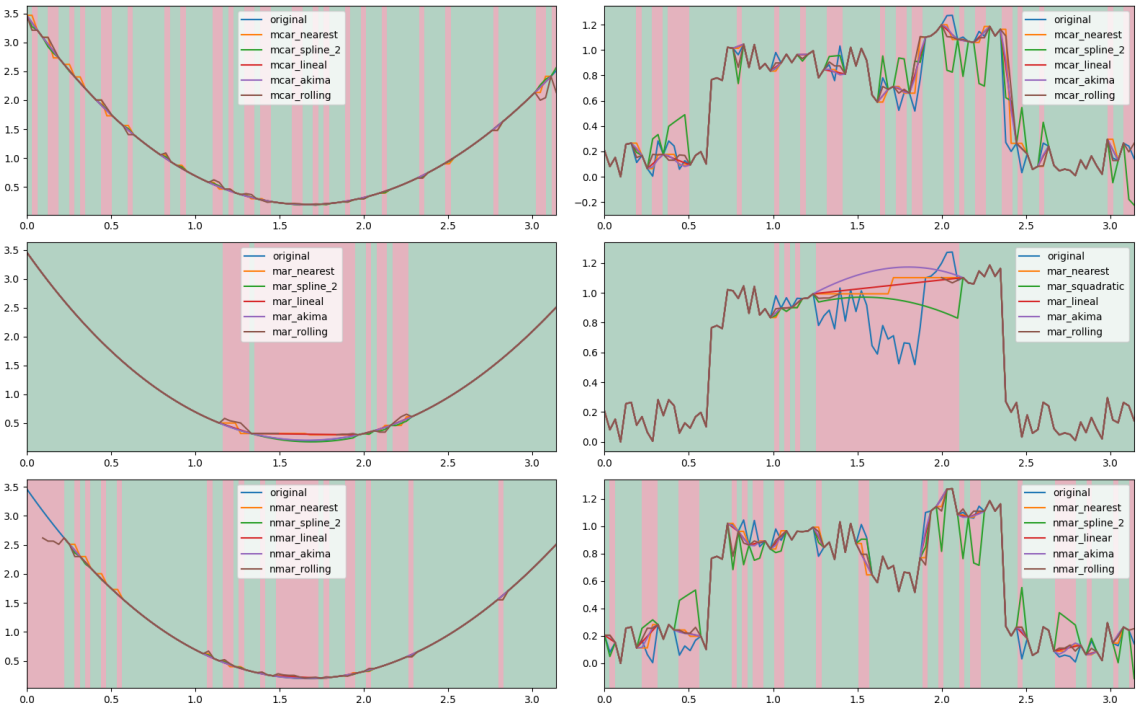


Figure 6.5: Multiple imputation methods are applied to two different signals with three different amputation strategies. In each of the examples, the original signal and the missing observations zones (red shade) are marked. These examples show the signal properties and missing observations distributions are decisive in the imputation success.

CHAPTER 7

Surrogate models

Computer simulations often used to model physical problems, play crucial roles in engineering, but can be so computationally expensive due that the computation required for a predictions makes them not suitable where results are expected in near real-time. For this purpose, machine learning surrogate models can be built on expensive simulation results, in order to estimate the predictions of mathematical models with far less computational load.

7.1 Background

Complex systems often involve exploring broad design spaces to obtain a general overview of the simulating scenario. Typically, the design space is screened to identify and remove design variables which are less important. Bearing this in mind, as surrogate models evaluations are capable of replacing individual simulations, they are widely used in initial phases of design to explore the solution space. These surrogate models can be used as guidance for optimising the simulation parameters to reduce the number of experiments and design settings.

The selection of the type of surrogate models to use for mimicking the simulation system depends on several factors. Elements to consider in selecting an appropriate surrogate model include the following:

- Size: the number of input and output parameters of the simulation
- Accuracy: the level of the approximation needed to consider the surrogate model adequate
- Computational time: how much time is required for training the surrogate model and, once trained for providing the prediction result(s)

- Quantity of samples: the number of simulation experiments needed to train and test the surrogate model
- Complexity of the surrogate model: includes surrogate model characteristics such as level of interpretability and number of parameters to estimate

All these factors are dependent on the simulating scenario and the specific application use case specifications such as response time limits, level of accuracy needed in the prediction and computation process available. The accuracy of a surrogate model is determined by the experimental design used to select data points, the size of the design space, or range of explored values of design variables. There is a trade-off between the accuracy of a surrogate model and the resources needed to build it. If surrogate models are built with a reduced number of data points, they are generally less accurate than models built with a larger number of data points. Using dimensionality reduction methods, sensitivity analysis or partitioning the problem in training multiple simpler surrogates models can help reducing the size of design experiments needed. An alternative is replacing computationally expensive simulations with approximate simulations, obtaining many more data points with less computational cost. However, a surrogate model built with approximate information may produce biased results. A combination of previous strategies is to run a large number of approximate simulations and a smaller number of detailed simulations and then combining the two sets of results to produce a final surrogate model.

7.2 Case study

Over the last three decades there have been big advances in lithium ion (Li-ion) batteries. Their interest resides in their high power and energy density characteristics and, because of that, nowadays they are popular in small devices, such as cell phones or laptops. However, durability, performance and security are critical characteristics of this technology and constitute a barrier to accessing other markets. There is interest in using this lithium ion batteries for other applications that may require bigger sizes, capacities or life, and therefore it is necessary to get to know how different variables can affect them. Furthermore, ensuring their correct performance under different conditions is essential for designing and predicting different problems as ageing phenomena. Having the capacity of simulating batteries performance under different conditions can help in experimental design, battery design optimisation and understanding their limitations.

The Figure 7.1 describes the inner functionality of a Li-ion battery. The electrochemical reaction, where lithium ions (Li+) and the electrons separate, occurs at the electrode-electrolyte interface. Li+ are shuttled from one electrode to the other through the separator; the direction in which the ions flow depends on whether the battery is charging or discharging. When a Li-ion leaves one electrode, the electron that was paired with it travels through an external circuit producing work and meets up with the lithium ion on the other electrode. Charge and discharge processes are detailed next.

- Charge process: In this process, electrical current is supplied to the battery, in such a way that lithium ions flow from the cathode to the anode through the electrolyte. Then, a reduction reaction in the host of anode happens, which causes guests (Li+) insertion between the layers of negative electrode. During this process, the cathode oxidises. At the same time, the electrons flow from the cathode to the anode through the external circuit, since the electrolyte does not allow passing them through it. This movement causes a rise in the differential potential between the electrodes that can be seen in the rise of voltage of the cell.

- Discharge process: During this process the battery supplies electrical current and the electrons flow from the external circuit, passing through the cathode and finally leaving through the anode. This process forces Li-ions to leave the anode, causing its oxidation, as well as lithium ions are inserted in the cathode, what causes its reduction.

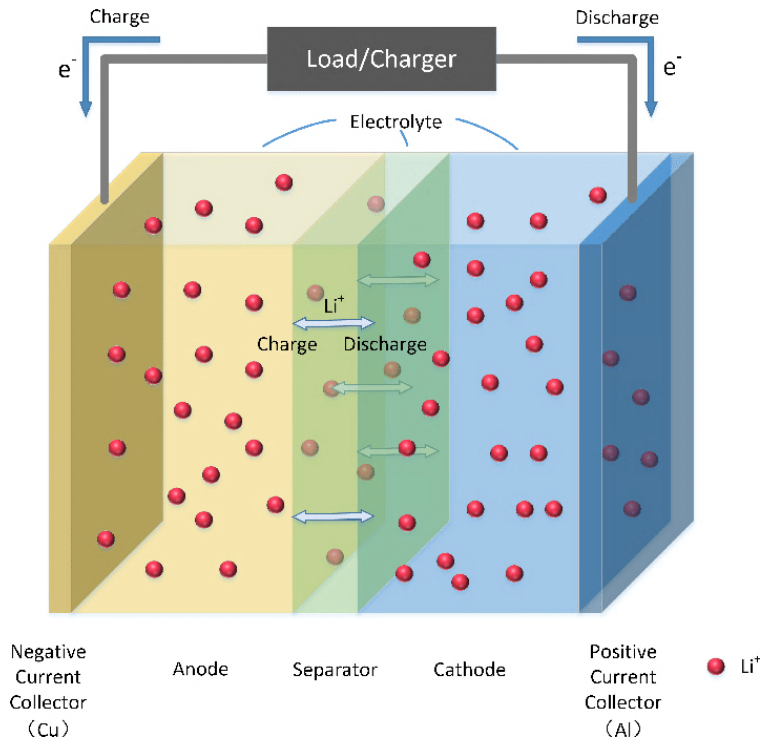


Figure 7.1: Lithium-ion battery schema [84].

Research interest resides in designing cells, needing optimising their material, cost and performance for different applications and scales. Therefore, the model should include electrochemical effects that are characteristic of material and electrolyte used in each case. Furthermore, it is necessary to solve the problem in an efficient way, because optimising requires large amount of simulations. Different simulations are needed to compare different characteristics of each cell, and to ensure that each of them performs correctly in different operations modes for a particular application.

Li-ion cell models are categorized into two main groups. The first type is based on an empirical approach that relyies on capacitor-resistor networks such as equivalent circuit models. Due to simplicity and low computational cost, empirical models have been used widely in online implementations. However, physical insight of the cell is neglected, resulting in prediction errors when used over a broad operating region. The second type includes physics-based models like pseudo two dimensional model where the distribution inside the cell along cell sandwich is obtained by solving material and charge balances. Thus, physics-based models can be applied in different operation modes, but the complexity and computational cost restricts the use of these models for on-board applications [33].

When it comes to modelling, most of the work focuses on validating and demonstrating 1D models for constant current charge or discharge. In low rates, Li-ion diffusion resistance within active particles can be treated as lumped masses and therefore, for the solid phase concentration, simplistic sub-grid scale models can be applied. However, Smith and Wang [68] pointed out that this assumption is inadequate for some applications such as hybrid electric vehicles. Consequently, physics-based models become essential for some applications that do not only work in low rates. In addition, physics-based models give the chance of using the model and its physical parameters to optimise the design of a cell for a specific use.

Furthermore, it is interesting to have an electrochemical based model in order to detect ageing effects. Dealing with the analysis and identification of deterioration of cells could lead to complex work if all influential parameters should take into account, such as operation mode, state of charge, state of health and so on. In a real use of a battery, the conditions in which it operates are part of large variety. That is why the experimentation can become unfeasible. In a model of this kind, there are difficulties that can be foreseen when it comes to physical-chemical parameters identification. In order to obtain the parameters, several electrochemical, morphological and phase analysis techniques can be applied.

Even so, in the literature for similar cells, it is easy to find different experimental equations and physical-chemical parameters [4, 62] due to parameter identification difficulties. Zhang et al. [85] analysed Li-ion physical-chemical parameters sensitivity to explain this issue in detail. More difficulties are found to get to know a parameter dependence to temperature in order to obtain a better thermal model. In these cases, dependencies are usually taken from literature, due to the difficulties involved to obtain them experimentally for a specific cell, as can be seen in [74].

7.3 Innovation

Due to the complexity of the simulation of the case study, surrogate models where proposed in order to reduce design space. The approximate predictions obtained by surrogates are good enough to guide domain experts in simulation parameters identification.

A two step solutions is proposed:

1. Determining if simulation with certain input parameters is capable of converg-

ing. A classification ensamble model is proposed as surrogate due to the complexity available in simulation data.

2. For the cases when the simulation model is expected to converge, obtaining Ragone energy and power estimates using two regression models (one per output to be estimated).

These two steps are represented in the diagram shown in Figure 7.2.

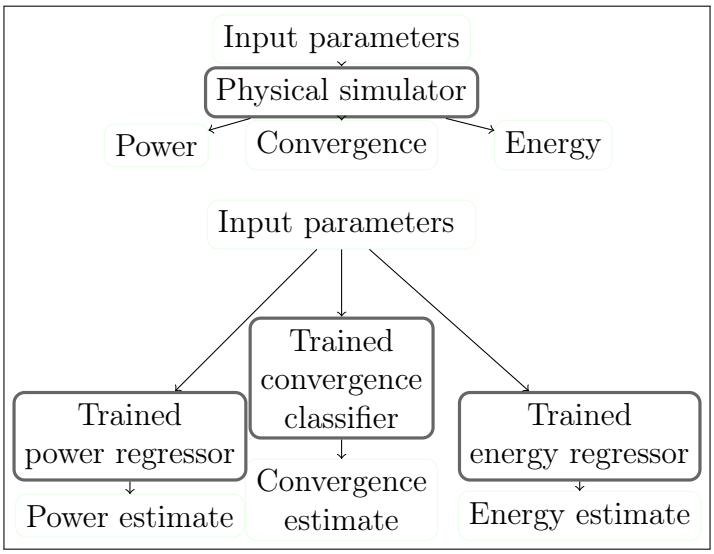


Figure 7.2: The general structure of the model. A simulator produces values for energy and power density integrals, together with a convergence flag indicating the successful execution of a run, up to a fixed end time. A classifier and two regressors learned from the simulated data can be exploited to generate estimates. The classifier estimates whether a simulation will produce physically unrealizable conditions, and therefore stop without completing a run. The two regressors estimate integrated energy and power for converging runs.

We consider that, in a Pseudo-2D electrochemical model, the completion status of a simulation can either be full (corresponding to propagation up to the intended complete temporal extension) or partial (corresponding to early termination). This status depends on several phenomenas, such as the consumption of chemical species or variations in the separator/electrode interface resistance due to parasitic ion deposition and layering reactions. In order to provide with flexibility capacity to the surrogate, the use of ensemble methods [63] was selected for this purpose. The article [56] appended in Part II details the ensemble method strategy and the classification models used for this particular approach.

7.4 Conclusions and future work

The present contribution has introduced and evaluated composite surrogate models for the prediction of the performance of Lithium-Ion Batteries which combine classi-

fiers based on deep ensembles and structured regressors for the prediction of energy and power densities. The quantitative results obtained indicate that ensemble models can indeed outperform state of the art models based on a single deep network. Furthermore, we have quantitatively validated the applicability of structured regressors to the estimation of energy and power.

Future work to be considered includes analyzing feature and parameter sensitivities [44] and the use of explainability and interpretability methodologies in order to understand the effect if the input parameters in the results. Furthermore, model and data uncertainty management [37, 30] should also be analyzed for the quantification of the prediction error before applying the developed methodology in a real-world scenario.

CHAPTER 8

Conclusions and future work

This chapter describes the general conclusions obtained in the framework of this research and ends with proposed future work.

8.1 Conclusions

In the context of industry 4.0, with the objective of providing insight into company operations, the data collected from multiple sensors together with external data and numerical models need to be processed in a controlled manner. However, multiple sources may contain erroneous observations, missing values and not uniform sampling, leading to low quality data. Thus, this uncontrolled collection process can produce poor decisions making that affects directly business results.

The aim of this thesis is guiding data managers in the definition of data pre-processing steps in order to guarantee the data quality of the system without loosing the traceability of the introduced errors. For that purpose, different tools are proposed that are capable of selecting optimal pre-processing strategies and providing insight into the errors made in the meantime. This work was inspired by the machine learning models generation. These models are trained using available historical data, minimising the errors of the fitting process.

Specifically, time series compression, feature joining, observation imputation and surrogate model generation processes are studied. In each of them, the adequate combination and selection between multiple strategies is pursued. This approach is defined according to data characteristics and user defined system properties and limitations.

With this objective in mind, this research has aimed to answer the three research questions posed in Chapter 1.

First of all, with the aim of automating each of the pre-process steps, the generation of a model for strategy selection has been proposed. In each of the cases

considered, the model acts as an oracle that, depending on the characteristics of the data received and the system limitations provided by the user, is able to decide which strategy fits best (i.e., the one generating the minimal error). For the generation of this model, previously user selected historical data is used. In each of the time series from the historical data, all the pre-processing strategies are tried separately and their response and committed error are saved. With these results and a characterization of the input signal, a machine learning model is trained for predicting which is the optimal strategy when a new signal arrives to the system.

Then, for controlling model's prediction quality, the pre-processing error is monitored in streaming mode. These models need to be revised with time to ensure that their strategy selection continuous to be valid. During the lifecycle of the machine learning model, the relationship between the input variables and the target variable can change. When this occurs, the model needs to be retrained with newer historical data to consider the new conditions. Using this strategy, the model is capable to adapt to the new situations.

Finally, the integration of the proposed methodology in the multiple pre-processing steps, allows the system manager to control the data quality of the data warehouse and to ensure that the applications that use those data as input will continue working as expected. This dissertation has studied specifically data compression, join, imputation and surrogate model construction steps. In each pre-processing steps, the experiments demonstrated how the proposed methodology is capable to adapt to different scenarios, showing its potential.

8.2 Future work

The approaches presented in this dissertation have several new paths to follow as future works. On the one hand, the proposed pre-process strategy selection models are specific for numerical time series data. Even if multiple datasets have been used in the experiments, with the idea that the provided results are not attached to specific dataset properties, unexplored situations may occur when unexpected data is received such as erroneous observations or unexpected data typology. On the other hand, each pre-processing steps considers a limited number of strategies to select from. Furthermore, the experiments use fixed input parameters, such as window size or univariate time series.

Apart from that, each pre-processing step has been considered separately, when in reality the performance of data cleaning can be evaluated in a combination on multiple processes. For example, in the optimisation of the joining method selection, observation imputation has not been considered as a possible method for filling missing time series values or reconstructing compressed versions of the time series. The combination of the pre-processing steps can enable a major precision in data cleaning, but at the same time keeping them separate allows detecting a problematic cleaning process in a simpler way.

Finally, in industrial environments, there are other processes related with data that still are not automated. One of those approaches is machine model generation,

where the model selection depends on the data experts and domain experts. In the lastest years, there is demand for machine learning systems that can be used by non-experts. This systems are supposed to automatically choose a good algorithm and feature pre-processing steps for the dataset. Moreover, this also allows replacing hand-engineered algorithms with novel approaches learned in a data-driven way.

Meta learning is the name given to the strategy of learning about the performance of machine learning algorithms across different datasets. For this purpose, the characteristics and performance of a large number of datasets are collected in order to determine which model could fit best in a new dataset. This technique can be used to recommend algorithms and search spaces parameters configuration from previous knowledge in similar tasks. These meta-models recommend algorithms or model configurations, but these models or configuration should be trained and parameters tuned for the specific task of the new dataset. In other words, the meta-learning results can be used as a guide or model parameter search space but it is not directly trained for the new dataset.

The features to construct the meta-models should be capable of estimating task similarity in order to provide promising configurations for the new task. There are two ways to characterise the dataset: 1) by features that describe statistical and information based properties of the dataset; 2) by the use of the performance of very simple learners (landmarking approach). In the first approach, to characterise datasets including simple, information-theoretic and statistical features are used, such as the number of instances, features, and classes of the datasets, as well as data skewness, and the entropy of the targets. For the second approach, simple models such as linear regressors or decision trees are used in order to estimate the relations between inputs and outputs. Once the relations are detected, these data properties are used for selecting adequate models for problem representations.

The results of meta-learning strategies can provide directly the recommended models and configurations for the new dataset, or provide a prediction of the error committed by certain algorithm.

Bibliography

[1] Foto N Afrati and Phokion G Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In *Proceedings of the 12th International Conference on Database Theory*, pages 31–41, 2009.

[2] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

[3] Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. *Journal of the ACM (JACM)*, 17(4):589–602, 1970.

[4] Paul Albertus, Jeremy Couts, Venkat Srinivasan, and John Newman. Ii. a combined model for determining capacity usage and battery size for hybrid and plug-in hybrid electric vehicles. *Journal of Power Sources*, 183(2):771–782, 2008.

[5] S. Aljanabi and A. Chalechale. Improving iot services using a hybrid fog-cloud offloading. *IEEE Access*, 9:13775–13788, 2021.

[6] Joseph Azar, Abdallah Makhoul, Mahmoud Barhamgi, and Raphaël Couturier. An energy efficient iot data compression approach for edge machine learning. *Future Generation Computer Systems*, 96:168–175, 2019.

[7] Dimitris Bertsimas, Colin Pawlowski, and Ying Daisy Zhuo. From predictive methods to missing data imputation: an optimization approach. *The Journal of Machine Learning Research*, 18(1):7133–7171, 2017.

[8] Davis Blalock, Samuel Madden, and John Guttag. Sprintz: Time series compression for the internet of things. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2:1–23, 2018.

[9] Ronald Newbold Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[10] Li-Juan Cao and Francis Eng Hock Tay. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on neural networks*, 14(6):1506–1518, 2003.

[11] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 2015.

[12] Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)? *GMDD*, 7(1):1525–1534, 2014.

[13] Chris Chatfield and Mohammad Yar. Holt-winters forecasting: some practical issues. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 37(2):129–140, 1988.

[14] Jan Chomicki and Jerzy Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Information and Computation*, 197(1-2):90–121, 2005.

[15] Edward Frank Codd. *The relational model for database management.* Addison-Wesley Publishing Company, 1990.

[16] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Annh Ratanamahatana, and Eamonn Keogh. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[17] Zengyu Ding, Gang Mei, Salvatore Cuomo, Yixuan Li, and Nengxiong Xu. Comparison of estimating missing values in iot time series data using different interpolation algorithms. *International Journal of Parallel Programming*, 48(3):534–548, 2020.

[18] Yiran Dong and Chao-Ying Joanne Peng. Principled missing data methods for researchers. *SpringerPlus*, 2(1):1–17, 2013.

[19] Frank Eichinger, Pavel Efros, Stamatis Karnouskos, and Klemens Böhm. A time-series compression technique and its application to the smart grid. *The VLDB Journal*, 24(2):193–218, 2015.

[20] Ronald Fagin, Benny Kimelfeld, and Phokion G Kolaitis. Dichotomies in the complexity of preferred repairs. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 3–15, 2015.

[21] Javier Franco, Ander Garcia, and Amaia Gil. Multivariate adaptive downsampling algorithm for industry 4.0 data visualization. In Hugo Sanjurjo González,

Iker Pastor López, Pablo García Bringas, Héctor Quintián, and Emilio Corchado, editors, *16th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2021)*, pages 588–597, Cham, 2022. Springer International Publishing.

[22] Venkatesh Ganti and Anish Das Sarma. Data cleaning: A practical perspective. *Synthesis Lectures on Data Management*, 5(3):1–85, 2013.

[23] Everette S Gardner Jr. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1):1–28, 1985.

[24] Amaia Gil, Marco Quartulli, Igor G. Olaizola, and Basilio Sierra. Learning optimal time series combination and pre-processing by smart joins. *Applied Sciences*, 10(18), 2020.

[25] Amaia Gil, Marco Quartulli, Igor G. Olaizola, and Basilio Sierra. Towards smart data selection from time series using statistical methods. *IEEE Access*, 9:44390–44401, 2021.

[26] Amaia Gil, Marco Quartulli, Igor G. Olaizola, and Basilio Sierra. Assist: Automatic smart selection of the suitable imputation technique. *Advanced Engineering Informatics*, 2022.

[27] Alasdair Gilchrist. *Industry 4.0: the industrial internet of things*. Springer, 2016.

[28] Marc H Gorelick. Bias arising from missing data in predictive models. *Journal of Clinical Epidemiology*, 59(10):1115–1123, 2006.

[29] Amara Graps. An introduction to wavelets. *IEEE computational science and engineering*, 2(2):50–61, 1995.

[30] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[31] Ankur Jain and Edward Y Chang. Adaptive sampling for sensor networks. In *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*, pages 10–16, 2004.

[32] Jacob Høxbroe Jeppesen, Rune Hylsberg Jacobsen, Fadil Inceoglu, and Thomas Skjødeberg Toftegaard. A cloud detection algorithm for satellite imagery based on deep learning. *Remote Sensing of Environment*, 229:247–259, 2019.

[33] Xing Jin, Ashish Vora, Vaidehi Hoshing, Tridib Saha, Gregory Shaver, Oleg Wasynczuk, and Subbarao Varigonda. Applicability of available li-ion battery degradation models for system and control algorithm design. *Control Engineering Practice*, 71:1–9, 2018.

[34] Uwe Jugel, Zbigniew Jerzak, Gregor Hackenbroich, and Volker Markl. M4: A visualization-oriented time series data aggregation. In *Proceedings of the VLDB Endowment, 7(10)*, pages 797–808, 2014.

[35] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Proceedings 2001 IEEE international conference on data mining*, pages 289–296. IEEE, 2001.

[36] Takuto Kimura, Tatsuaki Kimura, Arifumi Matsumoto, and Kazuhisa Yamagishi. Balancing quality of experience and traffic volume in adaptive bitrate streaming. *IEEE Access*, 9:15530–15547, 2021.

[37] Michael Kläs and Anna Maria Vollmer. Uncertainty in machine learning applications: A practice-driven classification of uncertainty. In *International Conference on Computer Safety, Reliability, and Security*, pages 431–438. Springer, 2018.

[38] Danie Gerhardus Krige. Two-dimensional, weighted moving average trend surfaces for ore valuation. *Journal of the South African Institute of Mining and Metallurgy*, 66:13–38, 1966.

[39] Alexander LeClair, Aakash Bansal, and Collin McMillan. Ensemble models for neural source code summarization of subroutines. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 286–297. IEEE, 2021.

[40] Jeng-Wei Lin, Shih-Wei Liao, and Fang-Yie Leu. Sensor data compression using bounded error piecewise linear approximation with resolution reduction. *Energies*, 12(13):2523, 2019.

[41] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11, 2003.

[42] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[43] Jingchen Liu, Andrew Gelman, Jennifer Hill, Yu-Sung Su, and Jonathan Kropko. On the stationary distribution of iterative imputations. *Biometrika*, 101(1):155–173, 2014.

[44] Kailong Liu, Xiaosong Hu, Huiyu Zhou, Lei Tong, Dhammika Widanalage, and James Marco. Feature analyses and modelling of lithium-ion batteries manufacturing based on random forest classification. *IEEE/ASME Transactions on Mechatronics*, 2021.

[45] Yuehua Liu, Tharam Dillon, Wenjin Yu, Wenny Rahayu, and Fahed Mostafa. Missing value imputation for industrial IoT sensor data with large gaps. *IEEE Internet of Things Journal*, 7(8):6855–6867, 2020.

[46] Ruizhe Ma, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, and Rafal A Angryk. Solar flare prediction using multivariate time series decision trees. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2569–2578. IEEE, 2017.

[47] Diego Martín, Damaris Fuentes-Lorenzo, Borja Bordel, and Ramón Alcarria. Towards outlier sensor detection in ambient intelligent platforms—a low-complexity statistical approach. *Sensors*, 20(15):4217, 2020.

[48] Sky McKinley and Megan Levine. Cubic spline interpolation. *College of the Redwoods*, 45(1):1049–1060, 1998.

[49] Andrew V Metcalfe and Paul SP Cowpertwait. *Introductory time series with R*. Springer, 2009.

[50] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[51] Frank Nielsen. *Parallel Linear Algebra*, pages 121–145. Springer International Publishing, 2016.

[52] Stefan Oehmcke, Oliver Zielinski, and Oliver Kramer. knn ensembles with penalized dtw for multivariate time series imputation. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 2774–2781. IEEE, 2016.

[53] Margaret A Oliver and Richard Webster. Kriging: a method of interpolation for geographical information systems. *International Journal of Geographical Information System*, 4(3):313–332, 1990.

[54] Tuomas Pelkonen, Scott Franklin, Justin Teller, Paul Cavallaro, Qi Huang, Justin Meza, and Kaushik Veeraraghavan. Gorilla: A fast, scalable, in-memory time series database. *Proceedings of the VLDB Endowment*, 8(12):1816–1827, 2015.

[55] Tao Peng, Sana Sellami, and Omar Boucelma. Iot data imputation with incremental multiple linear regression. *Open Journal of Internet Of Things (OJIOT)*, 5(1):69–79, 2019.

[56] Marco Quartulli, Amaia Gil, Ane Miren Florez-Tapia, Pablo Cereijo, Elixabete Ayerbe, and Igor G. Olaizola. Ensemble surrogate models for fast lib performance predictions. *Energies*, 14(14), 2021.

[57] Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[58] Vijayshankar Raman and Joseph M Hellerstein. Potter's wheel: An interactive data cleaning system. In *VLDB*, volume 1, pages 381–390, 2001.

[59] Qiubing Ren, Mingchao Li, and Shuai Han. Tectonic discrimination of olivine in basalt using data mining techniques based on major elements: a comparative study from multiple perspectives. *Big Earth Data*, 3(1):8–25, 2019.

[60] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[61] Donald B Rubin. Multiple imputation after 18+ years. *Journal of the American statistical Association*, 91(434):473–489, 1996.

[62] M. Safari and C. Delacourt. Modeling of a commercial graphite/lifepo4 cell. *The electrochemical Society*, 158, 2011.

[63] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[64] Rianne Margaretha Schouten, Peter Lugtig, and Gerko Vink. Generating missing values for simulation purposes: a multivariate amputation procedure. *Journal of Statistical Computation and Simulation*, 88(15):2909–2930, 2018.

[65] Klaus Schwab. *The fourth industrial revolution*. Currency, 2017.

[66] Ritei Shibata. Selection of the order of an autoregressive model by akaike's information criterion. *Biometrika*, 63(1):117–126, 1976.

[67] Mark Skilton and Felix Hovsepian. *The 4th industrial revolution*. Springer, 2018.

[68] Kandler Smith and Chao-Yang Wang. Solid-state diffusion limitations on pulse operation of a lithium ion cell for hybrid electric vehicles. *Journal of Power Sources*, 161(1):628 – 639, 2006.

[69] RS Somasundaram and R Nedunchezhian. Evaluation of three simple imputation methods for enhancing preprocessing of data with missing values. *International Journal of Computer Applications*, 21(10):14–19, 2011.

[70] Johan Frederik Steffensen. *Interpolation*. Courier Corporation, 2006.

[71] Sveinn Steinarsson. *Downsampling Time Series for Visual Representation*. PhD thesis, Mechanical Engineering and Computer Science: University of Iceland, 2013.

[72] Karsten Sternickel. Automatic pattern recognition in ECG time series. *Computer Methods and Programs in Biomedicine*, 68(2):109–115, 2002.

[73] Alp Ustundag and Emre Cevikcan. *Industry 4.0: managing the digital transformation*. Springer, 2017.

[74] Lars Ole Valøen and Jan N Reimers. Transport properties of lipf6-based li-ion battery electrolytes. *Journal of The Electrochemical Society*, 152(5):A882, 2005.

[75] Rasmus Vestergaard, Daniel E. Lucani, and Qi Zhang. A randomly accessible lossless compression scheme for time-series data. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 2145–2154, 2020.

[76] Maksims Volkovs, Fei Chiang, Jaroslaw Szlichta, and Renée J Miller. Continuous data cleaning. In *2014 IEEE 30th international conference on data engineering*, pages 244–255. IEEE, 2014.

[77] Xi Wang and Chen Wang. Time series data cleaning: A survey. *IEEE Access*, 8:1866–1881, 2019.

[78] Matthew J Watson, Antonios Liakopoulos, Dragana Brzakovic, and Christos Georgakis. A practical assessment of process data compression techniques. *Industrial & Engineering Chemistry Research*, 37(1):267–274, 1998.

[79] Rebecca E Wilson, Idris A Eckley, Matthew A Nunes, and Timothy Park. A wavelet-based approach for imputation in nonstationary multivariate time series. *Statistics and Computing*, 31(2):1–18, 2021.

[80] Zhengzheng Xing, Jian Pei, and S Yu Philip. Early prediction on time series: A nearest neighbor approach. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[81] Xiao Xu, Xiaoshuang Liu, Yanni Kang, Xian Xu, Junmei Wang, Yuyao Sun, Quanhe Chen, Xiaoyu Jia, Xinyue Ma, Xiaoyan Meng, et al. A multi-directional approach for missing value estimation in multivariate time series clinical data. *Journal of Healthcare Informatics Research*, 4:365–382, 2020.

[82] Shu Yang and Jae Kwang Kim. Fractional imputation in survey sampling: A comparative review. *Statistical Science*, 31(3):415–432, 2016.

[83] Tae-Woong Yoo and Il-Seok Oh. Time series forecasting of agricultural products' sales volumes based on seasonal long short-term memory. *Applied Sciences*, 10(22), 2020.

[84] Jianan Zhang, Lei Zhang, Fengchun Sun, and Zhenpo Wang. An overview on thermal safety issues of lithium-ion batteries for electric vehicle application. *IEEE Access*, 6:23848–23863, 2018.

[85] Liquiang Zhang, Chao Lyu, Gareth Hinds, Lixin Wang, Weilin Luo, Jun Zheng, and Kehua Ma. Parameter sensitivity analysis of cylindrical lifepo4 battery performace using multi-physics modeling. *The Electrochemical Society*, 161, 2014.

[86] Shichao Zhang. Shell-neighbor method and its application in missing data imputation. *Applied Intelligence*, 35(1):123–133, 2011.

Part II

Appended Papers

CHAPTER 9

Summary of the appended papers

This Part first presents a summary of the four appended papers. It gives the tile, the purpose, the proposed methodology and the findings of each paper. Then, the listed papers are appended.

9.1 Paper 1

Title: Towards Smart Data Selection From Time Series Using Statistical Methods.

Journal: IEEE Access, IEEE

Year: 2021

Purpose: Effectively select a subset of significant data points in order to reduce data volumes without sacrificing the quality of the results of the subsequent analysis.

Methodology: This paper proposes a method for adaptively identifying optimal data point selection algorithms for sensor time series on a window-by-window basis. Thus, this contribution focuses on quantifying the effect of the application of data selection algorithms to time series windows.

Findings: The proposed method has been implemented and applied to a wide variety of real-world time series datasets from a public open database, demonstrating their value for the characterization and the compression of the data.

9.2 Paper 2

Title: Learning Optimal Time Series Combination and Pre-Processing by Smart Joins.

Journal: Applied Sciences, MPDI

Year: 2020

Purpose: This paper proposes the use of optimization in the pre-processing step, specifically studying a time series joining methodology, and introduces an error function to measure the adequateness of the joining.

Methodology: The proposed smart join methods is inspired in machine learning models generation. First, the joining model is fitted using training data; then, resampled data is predicted by applying the selected join method to the test data; finally, the model is validated using resampling error.

Findings: Experiments show how the method allows monitoring pre-processing errors for different time slices, indicating when a retraining of the pre-processing may be needed. Thus, this contribution helps quantifying the implications of data preprocessing on the result of data analysis and machine learning methods. The methodology is applied to two cases studies: synthetic simulation data with controlled distortions, and a real scenario of an industrial process.

9.3 Paper 3

Title: ASSIST: Automatic Smart Selection of the Suitable Imputation Technique.

Journal: Springer

Year: 2022 (submitted)

Purpose: Dealing with data that is missing due to unknown causes is a common problem in industrial applications. A possibility to mitigate this issue is to effectively impute missing observations that adjust to the actual characteristics of the signal received.

Methodology: This paper proposes a new method called ASSIST for adaptively identifying optimal imputation strategies for time series data on a window-by-window basis. For this purpose, time series and gap distributions are characterized first, and a model is trained from those features in order to predict the best fit imputation strategy from the available ones.

Findings: By the use of a simple machine learning model trained from signal features and gap distribution characteristics, the model is capapble of selecting an appropriate imputation method. This approach has been validated in the entire UCR time series public data archive.

9.4 Paper 4

Title: Ensemble Surrogate Models for Fast LIB Performance Predictions.

Journal: Energies, MPDI

Year: 2021

Purpose: Battery Cell design and control have been widely explored through modelling and simulation. Empirical models obtained for example by Machine Learning (ML) methods represent a simpler and computationally more efficient complement to electrochemical models and have been widely used for Battery Management System control purposes.

Methodology: This article proposes ML-based ensemble models to be used for the estimation of the performance of a LIB cell across a wide range of input material characteristics and parameters and evaluates 1. Deep Learning ensembles for simulation convergence classification and 2. structured regressors for battery energy and power predictions.

Findings: The results represent an improvement on state-of-the-art LIB surrogate models and indicate that deep ensembles represent a promising direction for battery modeling and design.

CHAPTER 10

Appended papers

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 26, 2021, accepted March 9, 2021, date of publication March 17, 2021, date of current version March 25, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3066686

Towards Smart Data Selection From Time Series Using Statistical Methods

AMAIA GIL[1,2], MARCO QUARTULLI[1], IGOR G. OLAIZOLA[1], AND BASILIO SIERRA[2]

[1]Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), 20009 Donostia-San Sebastián, Spain
[2]Department of Computer Sciences and Artificial Intelligence, University of the Basque Country (UPV/EHU), 20018 Donostia-San Sebastián, Spain

Corresponding author: Amaia Gil (agil@vicomtech.org)

This work was supported in part by the 3KIA Project through ELKARTEK, Basque Government.

ABSTRACT Transmitting and storing large volumes of dynamic / time series data collected by modern sensors can represent a significant technological challenge. A possibility to mitigate this challenge is to effectively select a subset of significant data points in order to reduce data volumes without sacrificing the quality of the results of the subsequent analysis. This paper proposes a method for adaptively identifying optimal data point selection algorithms for sensor time series on a window-by-window basis. Thus, this contribution focuses on quantifying the effect of the application of data selection algorithms to time series windows. The proposed approach is first used on multiple synthetically generated time series obtained by concatenating multiple sources one after the other, and then validated in the entire UCR time series public data archive.

INDEX TERMS Data selection, machine learning, optimization, time series.

I. INTRODUCTION

Fine grained, high temporal resolution sensor dynamic data is often useful for short-term forecasting and visualization [1]. However, communication latency, bandwidth constraints, high energy consumption and storage requirements for such data can be problematic [2]. Reducing the amount of data to be transmitted can help control latency time and save in energy consumption and storage [3].

A key challenge in the setup of point selection methodologies is reducing the size of the transmitted data without sacrificing its quality. A natural solution is to compress the data at the sensing devices, monitoring in real time the error introduced by this process. When adaptive point selection strategies are used [4], the objective is to select a subset of data points with a well-defined number of items to be transmitted periodically. Then, the effect of this compression methodology on subsequent data analysis and exploitation processes can be studied, for instance considering the difference between the recovered and the compressed versions of the data for a given original time series.

Blalock *et al.* [5] describe desirable properties of the compression algorithms:

1) Minimal buffering: on devices with small memory capacities, only small time windows can be used before data is compressed. Furthermore, large buffering can add unacceptable latency.
2) High decompression speed: decompression of data in order to recover the time series for other parts of the service such as visualization and machine learning applications needs to be quick.
3) Losslessness: noise and oversampling of data vary with time and depend on application. The compression is seen as a preprocessing step that is application specific. Using lossless compression algorithms ensure that the data could not depend on previously defined preprocessing strategies.

On the one hand, most work on compressing time series has focused on lossy techniques. Classical approaches for data compression include Fourier transforms [6], wavelets transforms [7], symbolic representation [8] and piecewise regression [1], [9].

The Fourier transform is a tool widely used for spectral analysis, signal filtering and compression. This transformation is adequate for analyzing the components of a stationary signal, as the sinusoidal components are propagated in all the time domain. For non-stationary signals the Fourier transform analysis is not appropriate because it is not able to

The associate editor coordinating the review of this manuscript and approving it for publication was Gianluigi Ciocca.

maintain any localized information of a signal. Wavelets are oscillations that decay quickly allowing an adequate analysis of non-stationary signals [10]. A common application of these transformations is signal compression. A threshold is defined and components with smaller value than the threshold defined are removed. Thus, after reconstruction by inversion formulas the signal maintains its original shape [11].

Symbolic representation approaches are designed to preserve enough information about the time series to support indexing or specific data mining algorithms, rather than to compress the time series per se. In order to change to symbolic compression, dimensionality reduction is usually applied by a window aggregation function such as piecewise aggregate approximation. Later, the variable values are normalized. Defining the aggregate functions implies not being able to reconstruct the signal easily, which amounts to losing the original measured data points. Finally, a symbol is selected depending on the range of values that it maps to.

Piecewise regression techniques divide a time series into fixed-length or variable length intervals and describe them using regression functions. As for regression functions, all types of functions can be used in principle. However, low-order polynomial functions, particularly constant and linear functions, can be estimated efficiently and are used frequently [12].

Yang *et al.* [13] propose using clustering techniques to group time series by similarity. A workload distribution strategy can be taken using this cluster division saving time in processing the compression. Then, each of the time series is compressed using autoregressive models.

Classical compression techniques reduce the volume of data by using transformations, regression models or aggregations functions. The result of the transformations, the parameters of the regression models or the symbolic representation of the aggregated values are stored to represent the signal. None of the data points measured are transmitted and the signal representation is dependent on the efficacy and adequateness of the compression methodology used.

On the other hand, lossless compression techniques use binary encoding for the representation. Pelkonen *et al.* [14] propose a compression algorithm that maintains the full representation of time series. It compresses separately the timestamp and the value of the data point. The timestamp part employs an efficient delta-of-delta encoding, while the measurement part uses a XOR'd floating point approach. The strategy of Blalock *et al.* [5] employs the predictability of a data point to obtain an effective encoding of the difference between the predicted values and the original one. This is done in order to take advantage of the correlation between continuous data samples.

Vestergaard *et al.* [15] propose a two step compression technique that allows advantages in terms of random access. First, in the preprocessing step the system determines the adequate values of the input parameters of the compression technique, such as number of samples in a chunk, using part of the data for the training. Then, the time series is divided in chunks and compressed separately, implying no need to decompress complete the time series for a random access.

Lossless compresssion techniques help reducing the volume and storage of data to be transmitted and satisfy all the properties above mentioned. However, the compressed version of the signal cannot be used for visualization, control or analytic applications directly, as all the data points are saved with the same quality, only the encoding time series data has been optimized to save storage.

An alternative to compressing techniques is using point selection algorithms to reduce the data volume. These techniques aim to select the most significant or representative points. One option apart from selecting points from a regular sampled signal would be using adaptive sampling methods [16]. These approaches study the level of variance between the collected data over a certain time frame and dynamically adjust the sampling frequency of the device. Adaptive sampling approaches work well in applications where the collected time series are stationary. In the case of quickly varying data, these approaches perform poorly.

It is desirable to be able to compress time series from stochastic processes into streams with constant or limited length in order to meet memory capacity limitations. To the best of our knowledge, there has been no reported work on time series compression with rate adaptability and the ability to flexibly preserve different characteristics of interest of a given time series. In this sense, the contributions put forward by the present paper are:

1) The idea of combining different data points selection methodologies using their potential in the current signal window.
2) A definition of errors for the determination of the optimal data points selection methodology in each moment and for different characteristics of interest relative depending on the envisaged application
3) An algorithm that implements the above methodology.

The proposed approach searches, inside a defined set of point selection algorithms, the optimal solution for the actual time frame of the time series. The adequateness of the selected algorithm can be evaluated and monitored in time to guarantee the quality of the compression technique.

Even if this compression strategy is lossy, monitoring the compression allows controlling window sizes and deciding when to retrain the point selection model in order to adjust it to the current characteristics of the signal. With this approach, the above mentioned desired properties of compression systems can be controlled by the user. This process is shown in Figure 1.

This methodology has been validated with several synthetically generated time series and with all the datasets available in the UCR Time Series Classification Archive [17]. The proposed approach is capable to adapt to the dynamics of the time series effectively using different error functions.

The rest of the paper is structured as follows. Section II introduces the available point selection methods. Then,

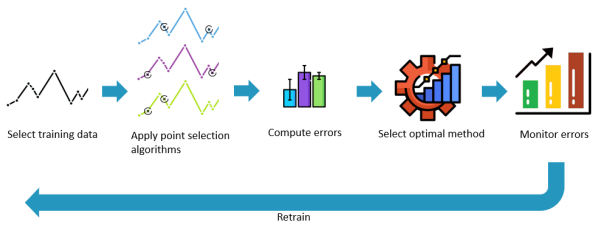


FIGURE 1. Adaptive optimal point selection method process. First, the training data is selected and multiple point selection methods are applied. Then, the optimal algorithm is selected and introduced in the system as subsampling method. Finally, the errors are monitored in the process and if a model drift is detected, the point selection model is retrained.

the methodology and the proposed approach are explained in section III-A. Next, an example is shown in order to demonstrate its usefulness in section III-B. Section IV provides a description of the experiments setup, whereas section V shows the results of those experiments. Finally, conclusions and future work are presented in section VI.

II. THEORETICAL BACKGROUND

Consider a time series signal which is sampled with a constant frequency. Due to system limitations, for example with respect to memory, not all captured points can be stored, and therefore a data point selection methodology needs to be applied.

One possible classification of the data points selection algorithms is to consider the way in which those are applied [18]:

- Algorithms that work in batch mode: the data is processed in group or batches. The algorithm is used only when the batch or group is complete. This can require fewer network resources than online systems.
- Algorithms that work in online / streaming mode: when a new point arrives to the system the data points selection methodology is applied directly. A previously saved snapshot representing the (e.g. statistical) properties of previous points and the most recently received points need to be available in order to decide if the actual received point is saved.

Other possibility was proposed by Keogh *et al.* [4], a classification for data point selection algorithms based on the point selection strategy they adopt:

- Selection of the best representation of the time series with a maximum error at each point (local error) less than a certain value (*max_error*).
- Selection of the best representation of the time series with a maximum combined error by all the segments (global error) less than a given value (*max_total_error*).
- Selection of the best representation of the time series using $k-1$ segments (or equivalently k points).

In sections II-A, II-B and II-C different point selection algorithms found in the state of the art are explained using the above mentioned classification system.

A. DATA POINTS SELECTION USING A MAXIMUM VALUE FOR LOCAL ERRORS

A review of classical data point selection methodologies based on a maximal error value in a point of the time series is described in Watson *et al.* [19]. All the strategies work in online mode and depend on a maximum error threshold (*max_error*) that should be indicated by the user. This input value is defined using background knowledge such as sensor limitations or a variable noise scale. The appropriate selection of a threshold value guarantees that no important information is lost in the data point selection procedure.

The boxcar algorithm makes a selection of a point when the current value differs from the last saved value by an amount greater than or equal to the determined maximum error threshold bound for that variable. The last processed value before the one which exceeds the limit should be saved.

The backward slope methodology projects the error bound into the future on the basis of the line formed by the previous two data points. The first data point is selected if the second value lies outside the error bound. Once a new data point is recorded, the new line and the error bound are projected into the future, repeating the same strategy.

The swinging door strategy is similar to the one considered by the backward slope algorithm, except that the error bound is based on the slope of the line between the first and the current data point. When the current data point has exceeded the error bound defined, the data point at the previous time index is selected. Then the error bound and line are recalculated and the algorithm is repeated.

Keogh *et al.* [4] propose two point selection strategies that work in an iterative mode. Therefore, a fixed buffer or window size is needed in these cases, i.e, these strategies work in batch mode.

The top-down strategy starts considering all the segments between adjacent points. Then, the algorithm continuous merging contiguous segments by removing the intermediate point, i.e., the common extreme of the contiguous segments, that adds a minimal error value from all the possibilities. This is done in a iterative way and until the *max_error* value is not exceeded.

The bottom-up strategy starts instead with a unique segment defined by the two extreme points of the time series (first and last in time index). The algorithm adds points to the selected set in an iterative. Each time the point that have the greatest error in the actual representation of segments is added in the set and the segments are recalculated. This is done until the error committed is less than the *max_error* value.

These two strategies are adapted to the specific cases detailed in sections II-B and II-C by specifying different stopping criteria.

B. DATA POINTS SELECTION USING A MAXIMUM TOTAL ERROR VALUE

In the case of data points selection using a maximum total error value, a total error is calculated each time using an

aggregation function from total errors. That value is used and points are added to the selected set while the total error value is higher that the defined limit *max_total_error*.

C. DATA POINTS SELECTION USING MAXIMUM NUMBER OF SEGMENTS

The algorithms detailed in this section work in batch mode. In the case of data points selection using maximum number of segments, a fixed window in time series data is used as a batch and from there a maximum number of points k is selected to be part of the compressed signal. The value k should be defined by the user taking into account the limitations of the system, such as memory limits. The following paragraphs detail different algorithms with this objective.

The different versions of the largest triangle algorithm [20] are based on the use of the effective area of the data points: the significance of a point is indicated by the area of the triangle formed with its two adjacent points. Depending on how the adjacent points are selected or how the buckets are constructed, three different algorithms are generated.

- Largest-triangle-one-bucket (*ltob*): first, the effective areas for each point is calculated using prior and posterior data points in the time series. Then, k buckets are generated splitting the time series with approximately equal number of points in each of them. From each bucket the data point with the largest effective area is selected. In order to guarantee that the first and last point of the time series are selected, extreme buckets only contain those points.
- Largest-triangle-three-buckets (*lttb*): in this case, the effective area of a point does not depend on the position of its two adjacent points as in the previous case, it takes into account all data points from previous and posterior buckets. For that, first buckets are generated in the same way as the previous version of the algorithm (each bucket with nearly equal quantity of data points, except for extreme buckets that only contain the first an the last data points). Then, the effective area of each point is calculated using the mean value of data points from the posterior bucket and the data point selected from the previous bucket. Finally, the point with largest effective area is selected in each bucket.
- Largest-triangle-dynamic: this version of the algorithm does not rely on equal size buckets, but the buckets are generated in a iterative mode, starting from default buckets (equally sized). In order to determine which bucket needs to be larger or smaller, a linear regression model is fitted with the data points in each the bucket, the last data point of the previous bucket and the first point of the posterior bucket. Then, the fitted linear model validity is measured by the sum of squared error (SSE). Later, the bucket with the maximum SSE is divided in two and the bucket containing the minimum error is merged with one of its adjacent buckets (the one with minimum error option), guaranteeing that the number of buckets continuous to be equal to the limitation of the maximum number of selected points k. After each iteration, as new buckets are generated, linear regression models need to be recalculated. After a certain number of iterations, when the buckets sizes have become stable (or with similar SSE values), largest-triangle-three-buckets algorithm is used to calculate the effective area of each data point, finally selecting the most meaningful data point from each bucket.

The mode-median-bucket (*mmb*) [20] algorithm uses the mode and the median values of data points in each bucket in order to select a point from it. The data points are split into buckets that contain approximately the same number of data points. Then, each bucket is studied separately. If there is a unique mode in the bucket, the leftmost corresponding data point is selected. Otherwise, the data point equal to the median value from the bucket is selected. An exception happens with the minimum and maximum values of the time series, these peak points are selected directly from the buckets that contain them, in order to guarantee the preservation of extreme data points.

The M4 (*m4a*) strategy was defined by Jugel *et al.* [21]. First, n buckets are generated containing approximately equal number of points. In this case, as 4 points could be selected from each bucket, the number of buckets is equal to $n = truncate(k/4)$. Then, from each bucket the minimum and maximum values from both axis (time index and data values) are selected (hence the name M4). In some cases, the minimum or/and maximum point(s) in both axes can be represented by a unique data point, i.e., when the maximum or minimum data values occur in the minimum or maximum time index of the bucket, one point could be selected by two rules, selecting finally less quantity of points than expected initially. Bae *et al.* [22] expand the *m4a* point selection strategy for visualization services also using gradient values between adjacent columns of pixels to reduce more points.

Major extrema extraction technique proposed by Fink and Gandhi [23] consists on ranking the extrema values of the time series and selecting the most meaningful ones. These extremes would be the finally selected points for the compressed version of the time series. They considered four types of extrema: strict, left, right and flat. By strict they refer to local minimum and maximum points of the time series. Left and right are the extremes in time of a flat chunk and flat is a inner point of the flat chunk. Then, the importance of each type of extrema is calculated by the use of a distance and a positive parameter that determines the compression rate.

The simplest way to select data points from a time series with a constant sampling frequency is to pick them using a lower frequency value than the original one, i.e, selecting a point each n points (*oen*). This algorithm takes into account the maximum number of selected points (k) in order to select the new frequency w ($w = truncate\left(\frac{length(y_o)}{k}\right)$).

Top-down and bottom-up strategies can be modified using this time the length of the selected points set equal to the number of desired selected points k as stopping rule.

III. PROPOSED APPROACH: SMART COMPRESS

In this paper, a smart data selection method based on a optimization process is proposed. The aim of this optimization problem is to select the method that minimizes the errors of the point selection process for each feature.

First, a detailed explanation of the methodology is presented in section III-A and an example of application is shown afterwards in section III-B.

A. METHODOLOGY

The general concept of the proposed Smart Compress methodology can be described as follows:

1) First, the data point selection model is fitted using training data; in other words, the fit method selects the optimal algorithm that suits best the time series provided in the training.
2) Then, a compressed version of the signal is obtained by applying the fitted data selection method to the test data.
3) Finally, the adequateness of the data selection model is validated using the error between the original and the recovered signal from decoding the compressed signal.

Suppose there is a time window of the selected time series y_o that needs to be compressed to be transmitted by a limited channel. First of all, the fitting method is used to identify the optimal data points selection method for compression. The inputs needed for the fitting method are the data points in the selected time window y_o and a threshold indicating the maximum number of points that a compressed version of the signal could have (k). Then, another window of the same time series (z_o) can be used for testing the adequateness of the data points selection algorithm by the use of the method score. Finally, the optimal data points selection method is used for compressing other windows of the same time series with the predict method. This process is depicted in Figure 2.

In order to be able to compare different data point selection methodologies, the compressed version of the time series (y_c) should have a similar quantity of selected points after the compression strategy is applied to the original data (y_o). For this reason, the methods considered in the smart selection algorithm are the ones described in section II-C.

The algorithm selected as the baseline is *oen* as it is the simplest strategy that can be applied and the quality of the signal can be random in some cases. Furthermore, top-down, bottom-up, major extrema extraction and largest-triangle-dynamic algorithms are not considered in the experiments as the methods become very slow depending on the length of points in the window / buffer considered and the number of points to be selected.

The fitting process to find an optimal data points selection model could be mathematically represented as follows:

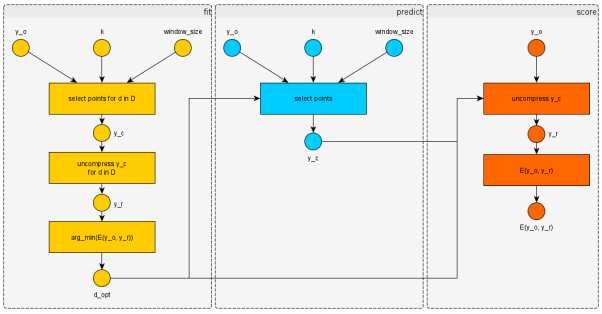


FIGURE 2. High-level view of the Smart Compress concept. The three available methods (fit, predict and score) are shown to indicate outputs and inputs in each case. First, a training time series is used, together with the parameter k, to identify an optimal points selection method (yellow part of the diagram). Once the method is identified, this optimal point selection is used by the score method to validate the result (shown in orange) and by the predict method to obtain the compressed version of a time series (shown in blue).

Suppose there is a window of the time series $y_o(t_o)$ where $t_o = [t_{o1}, t_{o2}, \ldots, t_{om}]$ and being m the number of points in the selected window. Let d be a data points selection method from the available methods set $D = \{ltob, lttb, m4a, mmb, oen\}$. Then, the compressed version of the time series, $y_c(t_c)$, is defined by the selected data points from y_o corresponding to time indexes $t_c = [t_{c1}, t_{c2}, \ldots, t_{cn}]$. The value $n \leq k$ where k is the maximum allowed quantity of $y_c(t_c)$.

From $y_c(t_c)$ the removed data points values are recovered by the use of linear interpolation method between available points of $y_c(t_c)$. The notation used to refer to the reconstructed version of the time series is y_r and it is defined for time index values that where contained in the original time series signal t_o.

Finally, an optimization problem is defined to select the most adequate method for the signal. This optimization is represented by:

$$\arg\min_{d \in D} E(y_o, y_r) \tag{1}$$

The detail of the fit method just explained is described graphically in Figure 3.

Depending on the purpose of the application, the most interesting properties of the signal could be totally different. The error functions can be defined in order to maintain these properties of the signal. Different signal characteristics are listed next for three different purposes:

- In visualization applications, properties such shape of the signal, visual outliers, linear trend of data and number of peaks are important to maintain. The general visual distortion generated from the compression can be measured by absolute sum of changes, mean absolute change, mean change, mean second derivative central and complexity-invariant distance.
- In control applications, the appearance of new events (peaks), change in signal tendency or frequency are

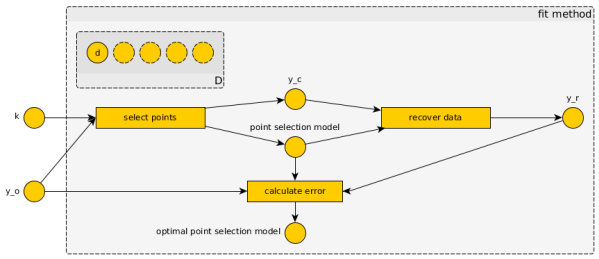


FIGURE 3. Fit methodology detail. The k parameter and the time series used for the training (y_o) are the inputs of the fit method. All the points selection methods available in D are considered separately. From each method used, a compressed version y_c and a recovered version y_r of the time series are obtained. This last time series y_r is compared with y_o, and the quality of the compression algorithm is measured by the error function. Finally, the identified optimal algorithm is saved as an inner object of the Smart Compress system for its later use by the score and predict methods.

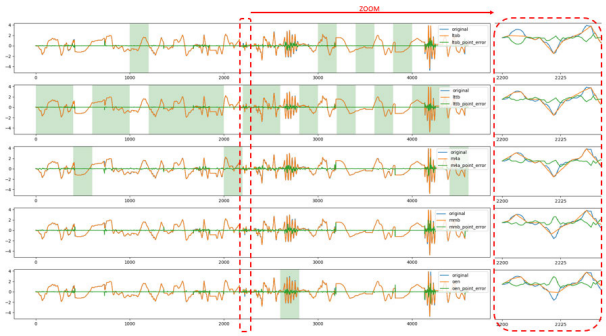


FIGURE 4. y_o (in blue), y_r (in orange), $y_r - y_o$ (in green) time series from a synthetically generated time series. The optimal method selected by *MaAE* in each window is highlighted. At right, a detailed view of a representative time segment is added.

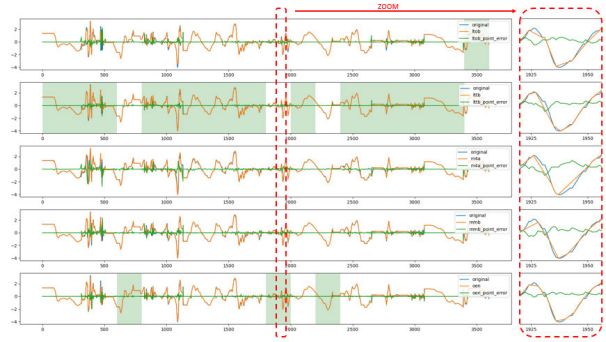


FIGURE 5. y_o (in blue), y_r (in orange), $y_r - y_o$ (in green) time series from a synthetically generated time series. The optimal method selected by *MeAE* in each window is highlighted. At right, a detailed view of a representative time segment is added.

important. In statistical control process applications, values ratio beyond r times standard deviation, longest strike above/below mean and count elements above / below certain value need to be considered.

- For analytical proposes, outliers and statistical properties such kurtosis, maximum, mean, median, minimum, quantiles, skewness, standard deviation and variation coefficient are essential.

Four different error functions have been used in the experiments. These error functions are selected in order to measure the distortion generated due to the use of the point selection algorithm. These error functions are detailed next:

- Percentage RMS difference [24]:

$$PRD = \left(\frac{\sum (y_o(i) - y_r(i))^2}{\sum (y_o(i))^2} \right)^{1/2} \tag{2}$$

- Normalized root mean square deviation [25]:

$$NRMSD = \frac{\left(\frac{\sum (y_o(i) - y_r(i))^2}{length(y_o)} \right)^{1/2}}{max(y_o) - min(y_o)} \tag{3}$$

- Mean absolute error [26]:

$$MeAE = \frac{mean(abs(y_o(i), y_r(i)))}{max(y_o) - min(y_r)} \tag{4}$$

- Maximum absolute error [27]:

$$MaAE = \frac{max(abs(y_o(i), y_r(i)))}{max(y_o) - min(y_r)} \tag{5}$$

B. PRELIMINARY APPLICATION EXAMPLE

The considered datasets are the ones available at the UCR Time Series Classification Archive [17]. The Archive contains 128 classification time series datasets of different types including sensor data, simulated data, motion data from several devices and health data such as electrocardiograph (ECG), electrooculography (EOG) and hemodynamic data. Depending on the dataset, either all the time series contained in it have same length, or the length varies between different time series.

Several synthetic time series are generated combining time series from two different datasets (AllGestureWiimoteX and UWaveGestureLibraryX) with significantly different statistical properties. The optimal method changes depending on the error considered and the characteristics of the synthetic time series in the time window that is being processed. Figure 4 considers *MaAE*, Figure 5 *MeAE*, Figure 6 *NRMS* and Figure 7 *PRD*. The figures show the original time series values (blue), the recovered time series values (orange) and the error in each point (green). The analysis window size is fixed to 200 points and from each window the maximum number of points that can be selected (k) is fixed to 50. In each window, the optimal method is marked with a green background.

In all cases, the method that is selected as optimal in most of the windows is the *lttb* algorithm. This effect is more notable where a mean or cumulative value of point to point errors is used. By contrast, if the importance to extreme values is given, for example using maximum value, when a function such as *MaAE* is used, the optimal method depends more on the local characteristics of the window studied. Thus, Figure 4 shows that it is not a clear winner when it comes

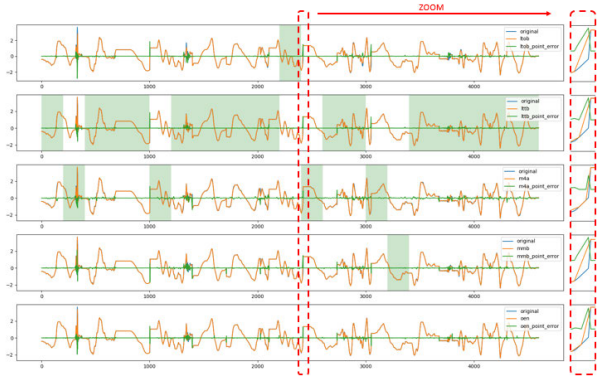


FIGURE 6. y_o (in blue), y_r (in orange), $y_r - y_o$ (in green) time series from a synthetically generated time series. The optimal method selected by *NRMS* in each window is highlighted. A detailed view of the some representative time segment is added. At right, a detailed view of a representative time segment is added.

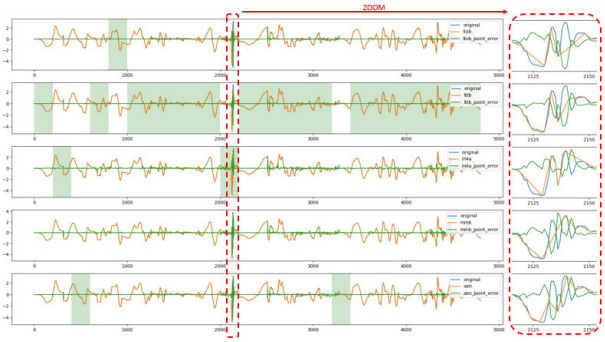
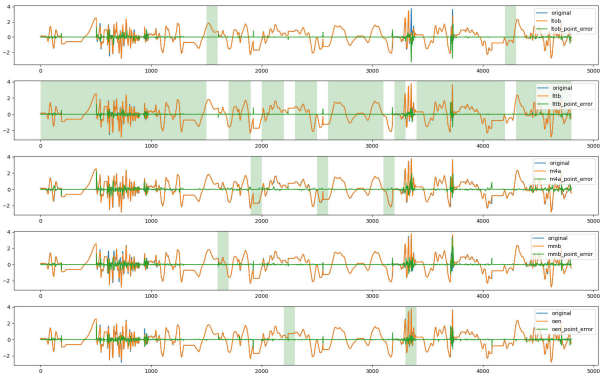
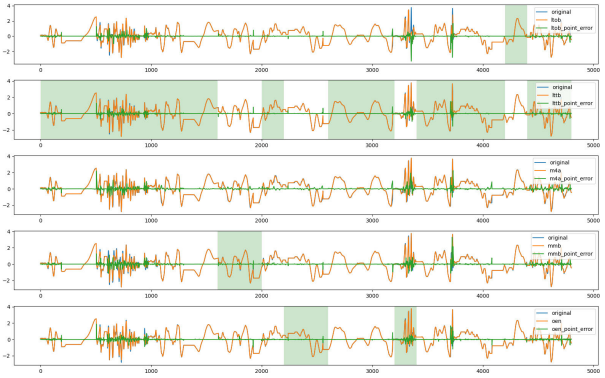


FIGURE 7. y_o (in blue), y_r (in orange), $y_r - y_o$ (in green) time series from a synthetically generated time series. The optimal method selected by *PRD* in each window is highlighted. At right, a detailed view of a representative time segment is added.



(a) window size = 100



(b) window size = 200

FIGURE 8. Effect of window size parameter in the optimal method selection.

to point selection strategies, as all depends on the time series characteristics and on the error used to measure it.

In Figure 8 the effect of the window size selection is shown. Even if the total number of points in the complete time series is the same, the optimal method distribution changes. In a smaller window size, the quantity of points to select from is reduced so the algorithm can select them quicker. Furthermore, a smaller window allows adjusting the algorithm to the actual time series characteristics. However, a bigger window size could help distributing the points in time in an smarter way.

It is important, therefore, that experiments are made with data from different origins and characteristics in order to ensure that the selection of the algorithm does not only depend on the error function used. For that purpose, an extensive experimentation is needed and this is shown in sections IV and V.

IV. EXPERIMENTAL SETUP

The experiments presented in this Section use the UCR Time Series data archive introduced in Section III-B is used. In all experiments, each dataset from the Archive is studied separately. Furthermore, from each dataset, each time series contained in the train or test set is used as an independent time series for point selection algorithm applications.

There are two possible strategies to define window size or batch length:

1) Based on a temporal window to schedule the data points selection periodically
2) Based on memory limitations that raise the data points selection algorithm when a reduction is needed.

In the particular case of having equidistant points, both previous cases arrive into the same definition of window size or batch length.

Time series in UCR Time Series Classification Archive do not have a time index. Therefore, an uniformly sampled index is used as time index of the series. As an uniformed time sampling is used, both strategies detailed above are equivalent. For each dataset, each time series is taken independently and the objective is to reduce at least 50% of the data points available in each of the time series. If the length of time series is variant among the dataset, the maximum length of the time series is taken to define the value of k in the downsampling

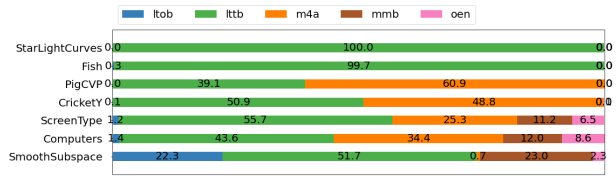


FIGURE 9. Zoom of the Figures 13 and 14 that are commented in the discussion for an easier comparison.

methodologies. Hence, k value corresponds to:

$$k = truncate\left(\frac{max(length(y_o))}{2}\right) \tag{6}$$

This k value was selected in order to ensure that in the inner buckets generated by the algorithms *lttb* and *ltob* contain at least two points. For each time series available in the dataset the fit method is applied, i.e, all available point selection algorithms are tried for compressing the signal and the optimal solution is saved.

V. EXPERIMENTAL RESULTS

Tables and Figures with the results of all datasets from UCR Time Series Classification Archive can be found in A.

For each time series from each dataset, the mean error value between y_r and y_o is used. Tables 2 and 3 report the mean and standard deviation values of all the errors among datasets for each method. The *opt* column presents the mean and standard deviation values in the case of using the optimal method (minimal error) in each of the time series. The datasets are sorted in ascending order starting with the dataset with the minimal mean value of the mean of errors of all the time series contained in the dataset. Similar information is shown visually in Figure 12. Due to printable table dimension limitations, datasets have been grouped in 8 different groups (16 datasets per group) in the same order that appear in Tables 2 and 3 and the sum of mean errors per method are shown in the Table 1.

TABLE 1. Grouped sum of mean errors of *MeAE* values obtained from each time series for datasets from URC Time Series Classification Archive.

datasets group	opt	ltob	lttb	m4a	mmb	oen
group 1	**0.377**	0.493	0.386	0.695	0.635	0.657
group 2	**0.853**	1.22	0.863	1.585	1.334	1.29
group 3	**1.508**	2.026	1.526	2.79	2.243	2.203
group 4	**2.74**	3.821	2.892	4.15	4.192	4.052
group 5	**4.493**	6.016	4.674	6.506	6.956	7.134
group 6	**7.055**	9.632	7.306	10.278	10.996	10.795
group 7	**10.384**	14.267	10.698	17.227	17.429	17.093
group 8	**59.529**	89.281	66.442	75.056	75.641	87.967
total sum	**86.939**	**126.756**	**94.787**	**118.287**	**119.426**	**131.191**

In general, the *lttb* method when using the *MeAE* is the most adequate method when different datasets are grouped. Moreover, selecting the optimal method in each time window when the difference between y_r and y_o time series is greater becomes more important. In other words, when the selecting points are not enough to preserve all the data adequately,

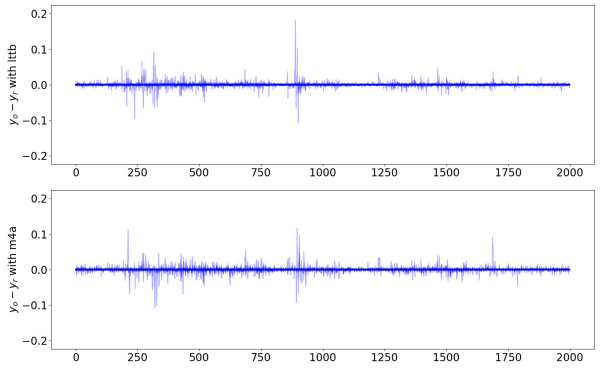


FIGURE 10. Difference between y_o and y_r when different point selection algorithms from D are applied to the PigCVP dataset.

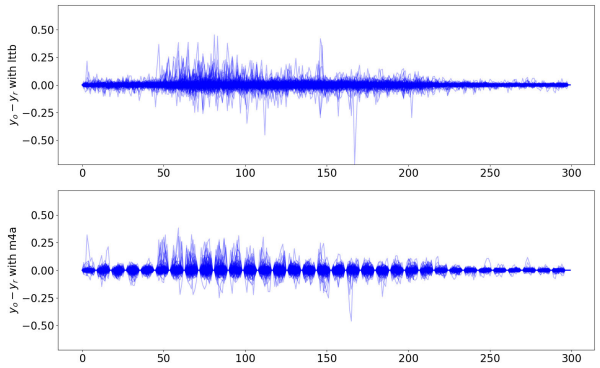


FIGURE 11. Difference between y_o and y_r when different point selection algorithms from D are applied to the CricketY dataset.

selecting the optimal points each time could have a greater impact. This is shown in Table 1 as the difference between choosing the optimal method in each window each time (column *opt* of the table) or using the same method for all the dataset (rest of the columns). The total sum of the grouped *MeAE* using the optimal point selection method in each time series is 86.939 that has nearly eight point difference compared to globally optimal methodology *lttb* value 94.787. Furthermore, this difference becomes much bigger when if another algorithm from the set D apart from *lttb* is selected, with value ranges from 31 to 44 points.

Each downsampling method has its own characteristics and adaptability depending on the time series properties. In case of having a variable that has a static or similar properties during all the time range, it is possible to select an unique optimal downsampling method that suits adequately the needs. However, this is not the usual case in real sensor data as process properties may vary with time and there is an stochastic part of the variable that cannot be controlled. In Figures 13 and 14 for each dataset, the percentage of the number of times each method has become the optimal one for downsampling a time

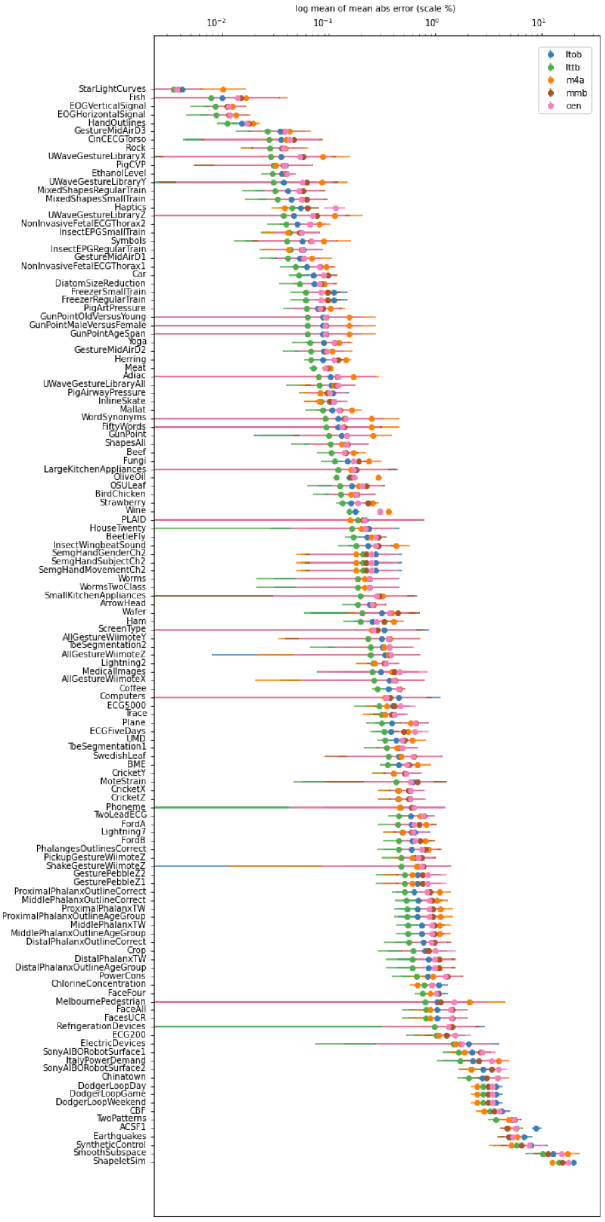


FIGURE 12. Mean value per dataset of *MeAE* values obtained for each time series of the dataset.

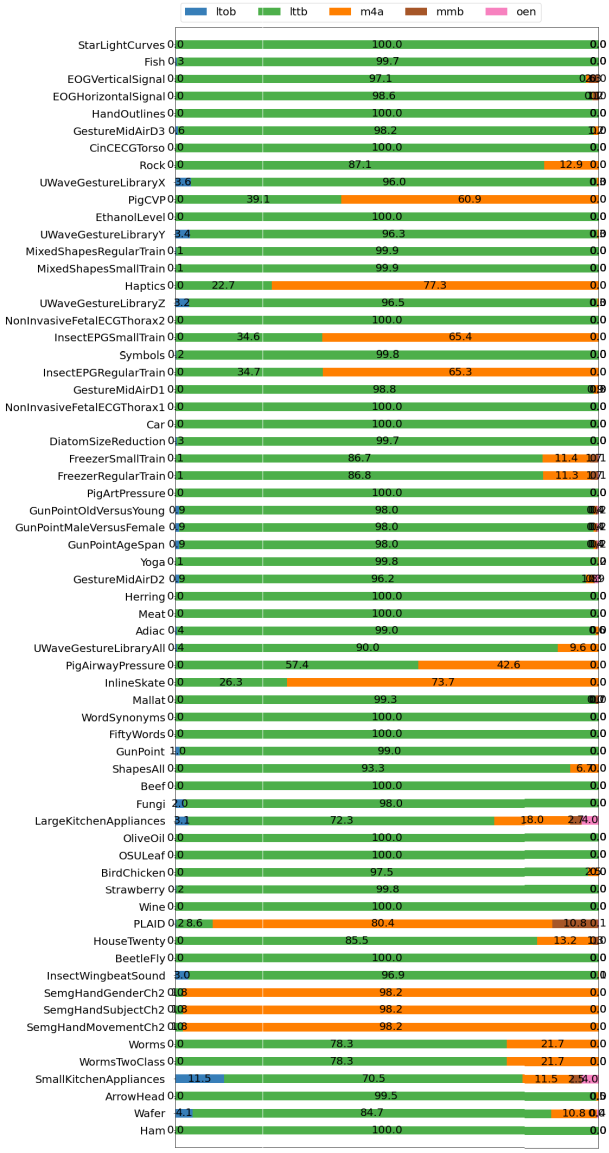


FIGURE 13. Distribution by dataset of the optimal data points selection method as determined by the use of *MeAE* per time series separately for datasets in Subset 1 of the UCR archive.

series (an instance of the dataset, equivalent to a specific time window) is shown.

In some cases, a unique method is responsible of being the most adequate in more than 90% of the cases, as it happens in StarlightCurves and Fish datasets. However, the selection for an adequate downsampling method is not obvious for others, such as for the PigCVP and CricketY datasets where the optimal solution is divided between *lltb* and *m4a* methods or even more distributed as it happens for datasets ScreenType, Computers and SmoothSubspace. In Figure 9 a zoom of Figures 13 and 14 is shown for datasets mentioned in both cases.

Figures 10 and 11 show point to point errors between y_o and y_r time series of the PigCVP and CricketY datasets where the optimal point selection strategy is not obvious. In both Figures, certain peaks of errors that appear using one of the methods are considerably reduced by the use of the other method.

This experiment shows that a methodology to adaptively select or / and monitor the point selection strategy is needed. Not all the time windows of a certain time series can be treated equally and even less when it comes to a real sensor data where all the context variables are not totally controlled.

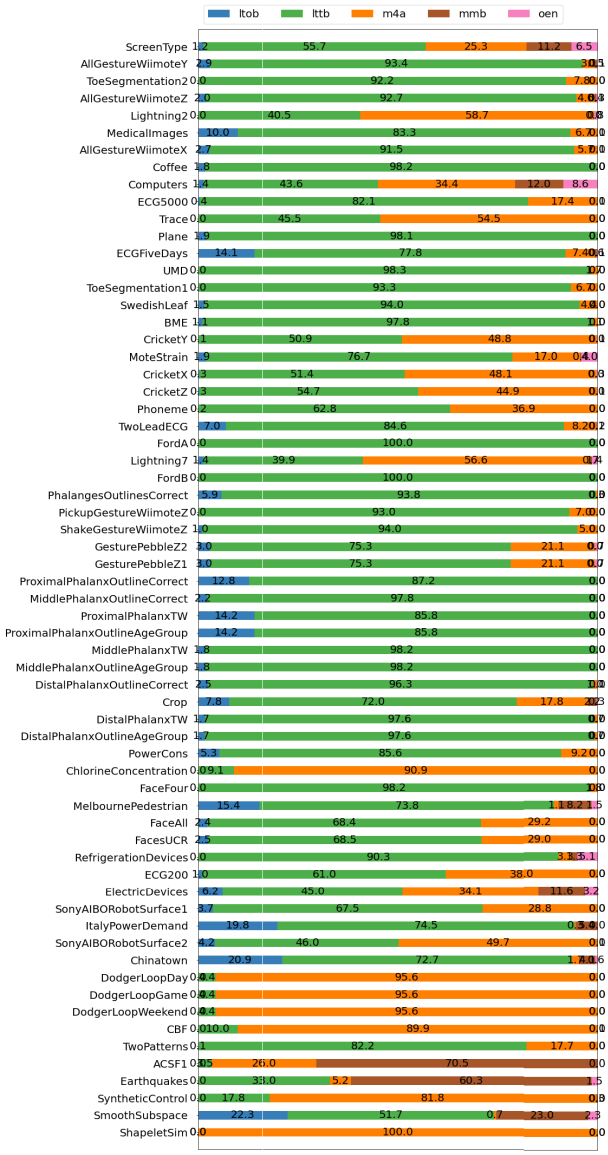


FIGURE 14. Distribution by dataset of the optimal data points selection method as determined by the use of *MeAE* per time series separately for datasets in Subset 2 of the UCR archive.

A change in the process, the dependency on other internal or external variables or even noise can affect the point selection method. Controlling the error values is critical in order to detect a point selection model drift and, in consequence, retrain the point selection strategy to maintain point selection quality.

VI. CONCLUSION AND FUTURE WORK

Considering the need to adaptively compress time series from stochastic processes into streams with constant or limited length in order to meet memory capacity limitations, this paper has put forward and demonstrated in a practical implementation the idea of combining different data points selection methodologies using their potential in the current signal window, while providing a definition of errors for the determination of the optimal data points selection methodology in each moment, and for different characteristics of interest relative depending on the envisaged application.

The proposed methods have been implemented and applied to a wide variety of real-world time series datasets from a public open database, demonstrating their value for the characterization and the compression of the data.

Future work to be considered includes combining algorithms to select points using a maximum value error for local errors, the ones that appeared in section II-A, in windows where maximum memory limitation per window is satisfied with methodologies that use maximum number of segments. With these combinations, it is possible to work with a trade off between maximum error allowed and memory size control. Furthermore, it should be possible to select points for multiple time series at once, as all of them will be saved in the same dataset or need to be synchronized, for example to be able to plot them in multiple dimensions. Finally, controlling the window size and quantity of data points to be selected depending on the characteristics of the time series would prove beneficial in a number of application scenarios [28]. Jain *et al.* [16] introduce an adaptive resampling frequency depending on the time series characteristics. The idea is using the actual error values not only to retrain the point selection strategy, but also to select adequate input parameters.

TABLE 2. Per-dataset mean and standard deviation values of the *MeAE* values obtained for each time series and by each method separately for Subset 1 of the UCR archive. The optimal column shows the mean and standard deviation values of the *MeAE* values when the optimal point selection method is selected for each time series of the dataset.

dataset	opt	ltob	lttb	m4a	mmb	oen
StarLightCurves	0.004 ± 0.002	0.004 ± 0.002	0.004 ± 0.002	0.01 ± 0.006	0.004 ± 0.002	0.004 ± 0.002
Fish	0.008 ± 0.01	0.01 ± 0.01	0.008 ± 0.01	0.017 ± 0.024	0.015 ± 0.019	0.014 ± 0.019
EOGVerticalSignal	0.009 ± 0.004	0.011 ± 0.005	0.009 ± 0.004	0.012 ± 0.004	0.011 ± 0.004	0.011 ± 0.004
EOGHorizontalSignal	0.009 ± 0.004	0.012 ± 0.006	0.009 ± 0.004	0.013 ± 0.005	0.012 ± 0.004	0.012 ± 0.004
HandOutlines	0.011 ± 0.002	0.015 ± 0.003	0.011 ± 0.002	0.019 ± 0.003	0.017 ± 0.003	0.017 ± 0.003
GestureMidAirD3	0.026 ± 0.013	0.035 ± 0.018	0.027 ± 0.014	0.043 ± 0.024	0.038 ± 0.019	0.039 ± 0.02
CinCECGTorso	0.027 ± 0.023	0.035 ± 0.031	0.027 ± 0.023	0.039 ± 0.027	0.046 ± 0.04	0.042 ± 0.036
Rock	0.027 ± 0.013	0.036 ± 0.018	0.028 ± 0.013	0.039 ± 0.024	0.037 ± 0.019	0.037 ± 0.018
UWaveGestureLibraryX	0.029 ± 0.03	0.036 ± 0.038	0.029 ± 0.03	0.087 ± 0.068	0.058 ± 0.057	0.053 ± 0.051
PigCVP	0.029 ± 0.019	0.039 ± 0.024	0.03 ± 0.019	0.031 ± 0.026	0.038 ± 0.032	0.038 ± 0.03
EthanolLevel	0.029 ± 0.006	0.036 ± 0.008	0.029 ± 0.006	0.039 ± 0.007	0.04 ± 0.008	0.04 ± 0.008
UWaveGestureLibraryY	0.03 ± 0.032	0.038 ± 0.041	0.03 ± 0.032	0.085 ± 0.062	0.061 ± 0.059	0.056 ± 0.053
MixedShapesRegularTrain	0.031 ± 0.016	0.041 ± 0.021	0.031 ± 0.016	0.053 ± 0.023	0.058 ± 0.032	0.052 ± 0.028
MixedShapesSmallTrain	0.033 ± 0.017	0.044 ± 0.023	0.033 ± 0.017	0.056 ± 0.025	0.061 ± 0.035	0.056 ± 0.03
Haptics	0.037 ± 0.009	0.054 ± 0.013	0.045 ± 0.011	0.038 ± 0.01	0.063 ± 0.016	0.114 ± 0.025
UWaveGestureLibraryZ	0.037 ± 0.039	0.046 ± 0.049	0.038 ± 0.039	0.113 ± 0.092	0.076 ± 0.077	0.071 ± 0.068
NonInvasiveFetalECGThorax2	0.04 ± 0.014	0.051 ± 0.017	0.04 ± 0.014	0.082 ± 0.019	0.067 ± 0.026	0.067 ± 0.025
InsectEPGSmallTrain	0.04 ± 0.017	0.055 ± 0.026	0.042 ± 0.02	0.041 ± 0.017	0.053 ± 0.023	0.055 ± 0.025
Symbols	0.04 ± 0.028	0.056 ± 0.039	0.04 ± 0.028	0.089 ± 0.069	0.068 ± 0.049	0.068 ± 0.047
InsectEPGRegularTrain	0.041 ± 0.019	0.057 ± 0.028	0.044 ± 0.022	0.042 ± 0.018	0.055 ± 0.024	0.056 ± 0.027
GestureMidAirD1	0.041 ± 0.019	0.053 ± 0.026	0.041 ± 0.019	0.069 ± 0.035	0.057 ± 0.026	0.058 ± 0.026
NonInvasiveFetalECGThorax1	0.048 ± 0.015	0.062 ± 0.019	0.048 ± 0.015	0.094 ± 0.019	0.082 ± 0.027	0.081 ± 0.023
Car	0.052 ± 0.011	0.071 ± 0.015	0.052 ± 0.011	0.089 ± 0.017	0.098 ± 0.019	0.089 ± 0.017
DiatomSizeReduction	0.053 ± 0.019	0.072 ± 0.027	0.053 ± 0.019	0.091 ± 0.026	0.084 ± 0.03	0.081 ± 0.028
FreezerSmallTrain	0.058 ± 0.016	0.111 ± 0.036	0.06 ± 0.018	0.095 ± 0.032	0.098 ± 0.027	0.084 ± 0.02
FreezerRegularTrain	0.058 ± 0.016	0.111 ± 0.036	0.06 ± 0.018	0.095 ± 0.032	0.098 ± 0.027	0.084 ± 0.02
PigArtPressure	0.061 ± 0.025	0.078 ± 0.031	0.061 ± 0.025	0.103 ± 0.035	0.087 ± 0.042	0.083 ± 0.039
GunPointOldVersusYoung	0.062 ± 0.069	0.088 ± 0.111	0.062 ± 0.07	0.154 ± 0.114	0.093 ± 0.102	0.092 ± 0.104
GunPointMaleVersusFemale	0.062 ± 0.069	0.088 ± 0.111	0.062 ± 0.07	0.154 ± 0.114	0.093 ± 0.102	0.092 ± 0.104
GunPointAgeSpan	0.062 ± 0.069	0.088 ± 0.111	0.062 ± 0.07	0.154 ± 0.114	0.093 ± 0.102	0.092 ± 0.104
Yoga	0.066 ± 0.022	0.089 ± 0.031	0.066 ± 0.022	0.124 ± 0.039	0.114 ± 0.038	0.112 ± 0.037
GestureMidAirD2	0.067 ± 0.031	0.09 ± 0.043	0.068 ± 0.032	0.107 ± 0.056	0.094 ± 0.043	0.095 ± 0.044
Herring	0.068 ± 0.01	0.088 ± 0.015	0.068 ± 0.01	0.144 ± 0.018	0.121 ± 0.017	0.111 ± 0.017
Meat	0.071 ± 0.007	0.097 ± 0.011	0.071 ± 0.007	0.104 ± 0.005	0.095 ± 0.008	0.092 ± 0.007
Adiac	0.079 ± 0.115	0.104 ± 0.16	0.08 ± 0.116	0.168 ± 0.121	0.119 ± 0.152	0.121 ± 0.149
UWaveGestureLibraryAll	0.08 ± 0.043	0.105 ± 0.053	0.081 ± 0.042	0.111 ± 0.06	0.117 ± 0.057	0.121 ± 0.054
PigAirwayPressure	0.081 ± 0.029	0.108 ± 0.043	0.083 ± 0.033	0.083 ± 0.028	0.096 ± 0.034	0.102 ± 0.039
InlineSkate	0.081 ± 0.024	0.11 ± 0.036	0.084 ± 0.027	0.081 ± 0.024	0.102 ± 0.029	0.112 ± 0.033
Mallat	0.088 ± 0.028	0.108 ± 0.033	0.088 ± 0.028	0.162 ± 0.038	0.127 ± 0.039	0.123 ± 0.039
WordSynonyms	0.093 ± 0.098	0.122 ± 0.135	0.093 ± 0.098	0.252 ± 0.202	0.139 ± 0.178	0.143 ± 0.165
FiftyWords	0.094 ± 0.099	0.123 ± 0.134	0.094 ± 0.099	0.252 ± 0.202	0.138 ± 0.173	0.134 ± 0.162
GunPoint	0.099 ± 0.076	0.13 ± 0.111	0.099 ± 0.076	0.257 ± 0.123	0.145 ± 0.1	0.147 ± 0.094
ShapesAll	0.104 ± 0.06	0.144 ± 0.087	0.104 ± 0.06	0.131 ± 0.063	0.148 ± 0.082	0.146 ± 0.082
Beef	0.105 ± 0.029	0.142 ± 0.04	0.105 ± 0.029	0.171 ± 0.047	0.144 ± 0.041	0.138 ± 0.039
Fungi	0.112 ± 0.03	0.15 ± 0.044	0.112 ± 0.03	0.236 ± 0.069	0.188 ± 0.055	0.169 ± 0.039
LargeKitchenAppliances	0.113 ± 0.138	0.179 ± 0.258	0.123 ± 0.164	0.161 ± 0.149	0.18 ± 0.24	0.17 ± 0.183
OliveOil	0.116 ± 0.001	0.154 ± 0.004	0.116 ± 0.001	0.287 ± 0.003	0.16 ± 0.002	0.173 ± 0.001
OSULeaf	0.126 ± 0.065	0.164 ± 0.086	0.126 ± 0.065	0.19 ± 0.071	0.224 ± 0.107	0.202 ± 0.097
BirdChicken	0.127 ± 0.057	0.176 ± 0.087	0.128 ± 0.057	0.16 ± 0.059	0.184 ± 0.083	0.18 ± 0.082
Strawberry	0.134 ± 0.018	0.159 ± 0.022	0.134 ± 0.018	0.255 ± 0.034	0.231 ± 0.042	0.187 ± 0.022
Wine	0.155 ± 0.006	0.177 ± 0.011	0.155 ± 0.006	0.36 ± 0.013	0.301 ± 0.011	0.3 ± 0.013
PLAID	0.155 ± 0.386	0.22 ± 0.552	0.186 ± 0.46	0.158 ± 0.391	0.209 ± 0.522	0.214 ± 0.535
HouseTwenty	0.158 ± 0.151	0.238 ± 0.211	0.162 ± 0.161	0.199 ± 0.15	0.218 ± 0.156	0.221 ± 0.178
BeetleFly	0.171 ± 0.031	0.226 ± 0.044	0.171 ± 0.031	0.25 ± 0.043	0.29 ± 0.05	0.273 ± 0.043
InsectWingbeatSound	0.179 ± 0.06	0.23 ± 0.081	0.179 ± 0.06	0.423 ± 0.138	0.29 ± 0.112	0.271 ± 0.102
SemgHandGenderCh2	0.18 ± 0.132	0.273 ± 0.201	0.207 ± 0.153	0.18 ± 0.132	0.222 ± 0.164	0.249 ± 0.184
SemgHandSubjectCh2	0.18 ± 0.132	0.273 ± 0.201	0.207 ± 0.153	0.18 ± 0.132	0.222 ± 0.164	0.249 ± 0.184
SemgHandMovementCh2	0.18 ± 0.132	0.273 ± 0.201	0.207 ± 0.153	0.18 ± 0.132	0.222 ± 0.164	0.249 ± 0.184
Worms	0.181 ± 0.15	0.24 ± 0.21	0.185 ± 0.165	0.215 ± 0.153	0.239 ± 0.181	0.241 ± 0.192
WormsTwoClass	0.181 ± 0.15	0.24 ± 0.21	0.185 ± 0.165	0.215 ± 0.153	0.239 ± 0.181	0.241 ± 0.192
SmallKitchenAppliances	0.183 ± 0.197	0.293 ± 0.367	0.198 ± 0.254	0.318 ± 0.263	0.295 ± 0.333	0.28 ± 0.25
ArrowHead	0.186 ± 0.055	0.241 ± 0.073	0.186 ± 0.055	0.262 ± 0.078	0.261 ± 0.082	0.253 ± 0.075
Wafer	0.194 ± 0.14	0.309 ± 0.243	0.204 ± 0.147	0.391 ± 0.243	0.443 ± 0.255	0.365 ± 0.192
Ham	0.197 ± 0.063	0.252 ± 0.086	0.197 ± 0.063	0.406 ± 0.087	0.327 ± 0.111	0.278 ± 0.093

APPENDIX
COMPLETE RESULTS OF UCR ARCHIVE

See Tables 2 and 3.

TABLE 3. Per-dataset mean and standard deviation values of the *MeAE* values obtained for each time series and by each method separately for Subset 2 of the UCR archive. The optimal column shows the mean and standard deviation values of the *MeAE* values when the optimal point selection method is selected for each time series of the dataset.

dataset	opt	lttob	lttb	m4a	mmb	oen
ScreenType	0.21 ± 0.309	0.332 ± 0.52	0.26 ± 0.423	0.251 ± 0.337	0.287 ± 0.472	0.26 ± 0.387
AllGestureWiimoteY	0.229 ± 0.189	0.31 ± 0.261	0.231 ± 0.191	0.376 ± 0.343	0.363 ± 0.325	0.36 ± 0.309
ToeSegmentation2	0.24 ± 0.162	0.318 ± 0.221	0.245 ± 0.179	0.332 ± 0.157	0.365 ± 0.232	0.304 ± 0.252
AllGestureWiimoteZ	0.242 ± 0.211	0.334 ± 0.326	0.244 ± 0.213	0.363 ± 0.342	0.376 ± 0.321	0.379 ± 0.334
Lightning2	0.244 ± 0.085	0.344 ± 0.11	0.261 ± 0.085	0.268 ± 0.093	0.33 ± 0.105	0.344 ± 0.108
MedicalImages	0.248 ± 0.162	0.307 ± 0.21	0.256 ± 0.181	0.391 ± 0.171	0.41 ± 0.271	0.456 ± 0.377
AllGestureWiimoteX	0.262 ± 0.227	0.364 ± 0.326	0.265 ± 0.231	0.402 ± 0.382	0.416 ± 0.355	0.414 ± 0.359
Coffee	0.283 ± 0.034	0.362 ± 0.05	0.283 ± 0.035	0.452 ± 0.044	0.461 ± 0.052	0.452 ± 0.055
Computers	0.286 ± 0.373	0.45 ± 0.646	0.341 ± 0.49	0.338 ± 0.431	0.372 ± 0.538	0.349 ± 0.465
ECG5000	0.288 ± 0.111	0.392 ± 0.172	0.294 ± 0.125	0.345 ± 0.118	0.413 ± 0.155	0.471 ± 0.171
Trace	0.295 ± 0.08	0.403 ± 0.107	0.311 ± 0.079	0.336 ± 0.133	0.389 ± 0.135	0.413 ± 0.126
Plane	0.311 ± 0.091	0.386 ± 0.122	0.311 ± 0.091	0.582 ± 0.144	0.647 ± 0.19	0.637 ± 0.169
ECGFiveDays	0.317 ± 0.078	0.379 ± 0.118	0.329 ± 0.091	0.553 ± 0.184	0.507 ± 0.123	0.644 ± 0.204
UMD	0.334 ± 0.056	0.428 ± 0.088	0.334 ± 0.056	0.604 ± 0.198	0.512 ± 0.112	0.489 ± 0.203
ToeSegmentation1	0.348 ± 0.139	0.454 ± 0.183	0.35 ± 0.14	0.433 ± 0.146	0.487 ± 0.18	0.485 ± 0.188
SwedishLeaf	0.356 ± 0.232	0.453 ± 0.293	0.358 ± 0.236	0.478 ± 0.27	0.622 ± 0.533	0.615 ± 0.472
BME	0.356 ± 0.064	0.448 ± 0.09	0.356 ± 0.063	0.673 ± 0.226	0.566 ± 0.136	0.539 ± 0.11
CricketY	0.384 ± 0.141	0.52 ± 0.198	0.405 ± 0.157	0.406 ± 0.146	0.516 ± 0.196	0.536 ± 0.201
MoteStrain	0.388 ± 0.334	0.621 ± 0.566	0.428 ± 0.382	0.587 ± 0.498	0.676 ± 0.587	0.578 ± 0.369
CricketX	0.419 ± 0.142	0.567 ± 0.202	0.441 ± 0.155	0.449 ± 0.166	0.557 ± 0.199	0.583 ± 0.205
CricketZ	0.422 ± 0.145	0.572 ± 0.206	0.444 ± 0.159	0.454 ± 0.169	0.562 ± 0.202	0.586 ± 0.208
Phoneme	0.431 ± 0.418	0.592 ± 0.62	0.468 ± 0.486	0.462 ± 0.403	0.605 ± 0.517	0.63 ± 0.589
TwoLeadECG	0.441 ± 0.093	0.584 ± 0.167	0.449 ± 0.097	0.713 ± 0.206	0.781 ± 0.17	0.766 ± 0.169
FordA	0.445 ± 0.159	0.594 ± 0.204	0.445 ± 0.159	0.818 ± 0.147	0.707 ± 0.306	0.628 ± 0.257
Lightning7	0.446 ± 0.152	0.637 ± 0.235	0.488 ± 0.169	0.496 ± 0.181	0.595 ± 0.2	0.62 ± 0.202
FordB	0.447 ± 0.132	0.6 ± 0.166	0.447 ± 0.132	0.794 ± 0.115	0.718 ± 0.256	0.636 ± 0.215
PhalangesOutlinesCorrect	0.451 ± 0.175	0.597 ± 0.279	0.453 ± 0.174	0.865 ± 0.26	0.795 ± 0.299	0.741 ± 0.271
PickupGestureWiimoteZ	0.472 ± 0.165	0.642 ± 0.228	0.475 ± 0.169	0.607 ± 0.205	0.742 ± 0.257	0.686 ± 0.222
ShakeGestureWiimoteZ	0.475 ± 0.433	0.679 ± 0.718	0.478 ± 0.437	0.668 ± 0.658	0.765 ± 0.629	0.741 ± 0.657
GesturePebbleZ2	0.486 ± 0.213	0.673 ± 0.356	0.509 ± 0.239	0.602 ± 0.279	0.759 ± 0.344	0.847 ± 0.401
GesturePebbleZ1	0.486 ± 0.215	0.673 ± 0.356	0.509 ± 0.239	0.602 ± 0.279	0.759 ± 0.344	0.847 ± 0.401
ProximalPhalanxOutlineCorrect	0.505 ± 0.133	0.633 ± 0.210	0.51 ± 0.131	1.084 ± 0.301	0.895 ± 0.222	0.831 ± 0.177
MiddlePhalanxOutlineCorrect	0.526 ± 0.119	0.689 ± 0.205	0.527 ± 0.119	1.041 ± 0.238	0.926 ± 0.214	0.881 ± 0.217
ProximalPhalanxTW	0.531 ± 0.139	0.674 ± 0.234	0.537 ± 0.134	1.12 ± 0.315	0.945 ± 0.234	0.87 ± 0.176
ProximalPhalanxOutlineAgeGroup	0.531 ± 0.139	0.674 ± 0.234	0.537 ± 0.134	1.12 ± 0.315	0.945 ± 0.234	0.87 ± 0.176
MiddlePhalanxTW	0.556 ± 0.136	0.736 ± 0.239	0.556 ± 0.136	1.092 ± 0.269	0.977 ± 0.23	0.916 ± 0.241
MiddlePhalanxOutlineAgeGroup	0.556 ± 0.136	0.736 ± 0.239	0.556 ± 0.136	1.093 ± 0.27	0.976 ± 0.23	0.916 ± 0.241
DistalPhalanxOutlineCorrect	0.556 ± 0.235	0.769 ± 0.366	0.557 ± 0.236	0.954 ± 0.323	0.98 ± 0.397	0.899 ± 0.36
Crop	0.593 ± 0.317	0.796 ± 0.447	0.619 ± 0.337	0.856 ± 0.402	0.865 ± 0.396	0.998 ± 0.52
DistalPhalanxTW	0.603 ± 0.266	0.862 ± 0.438	0.604 ± 0.267	1.086 ± 0.393	1.075 ± 0.458	0.981 ± 0.403
DistalPhalanxOutlineAgeGroup	0.603 ± 0.266	0.862 ± 0.438	0.604 ± 0.267	1.086 ± 0.393	1.075 ± 0.458	0.981 ± 0.403
PowerCons	0.653 ± 0.267	0.85 ± 0.382	0.665 ± 0.28	0.946 ± 0.377	1.283 ± 0.5	1.244 ± 0.465
ChlorineConcentration	0.673 ± 0.118	1.064 ± 0.213	0.776 ± 0.141	0.677 ± 0.114	0.925 ± 0.164	0.933 ± 0.189
FaceFour	0.75 ± 0.125	1.072 ± 0.206	0.751 ± 0.126	0.892 ± 0.137	1.031 ± 0.171	1.016 ± 0.164
MelbournePedestrian	0.784 ± 0.795	1.031 ± 1.095	0.799 ± 0.806	2.096 ± 2.353	1.118 ± 1.117	1.483 ± 1.507
FaceAll	0.786 ± 0.309	1.025 ± 0.469	0.819 ± 0.347	0.895 ± 0.296	1.433 ± 0.53	1.38 ± 0.54
FaceUCR	0.79 ± 0.318	1.025 ± 0.469	0.82 ± 0.343	0.886 ± 0.319	1.433 ± 0.539	1.407 ± 0.531
RefrigerationDevices	0.892 ± 0.836	1.401 ± 1.449	0.97 ± 1.152	1.386 ± 0.969	1.44 ± 1.087	1.319 ± 1.006
ECG200	0.959 ± 0.416	1.263 ± 0.647	1.018 ± 0.494	1.066 ± 0.388	1.262 ± 0.48	1.525 ± 0.597
ElectricDevices	1.251 ± 1.152	2.027 ± 1.89	1.463 ± 1.39	1.552 ± 1.144	1.721 ± 1.458	1.751 ± 1.483
SonyAIBORobotSurface1	1.577 ± 0.451	2.182 ± 0.765	1.644 ± 0.498	1.863 ± 0.518	2.62 ± 0.694	2.719 ± 0.888
ItalyPowerDemand	1.671 ± 0.681	2.233 ± 1.078	1.71 ± 0.703	3.861 ± 1.039	2.546 ± 0.933	3.341 ± 1.111
SonyAIBORobotSurface2	1.979 ± 0.462	2.817 ± 0.99	2.16 ± 0.587	2.155 ± 0.554	3.384 ± 0.752	3.89 ± 0.693
Chinatown	1.997 ± 0.438	2.764 ± 0.98	2.044 ± 0.462	3.849 ± 1.0	3.03 ± 0.617	3.855 ± 0.649
DodgerLoopDay	2.473 ± 0.363	3.681 ± 0.477	2.816 ± 0.414	2.475 ± 0.363	3.101 ± 0.433	3.425 ± 0.449
DodgerLoopGame	2.473 ± 0.363	3.681 ± 0.477	2.816 ± 0.414	2.475 ± 0.363	3.101 ± 0.433	3.425 ± 0.449
DodgerLoopWeekend	2.473 ± 0.363	3.681 ± 0.477	2.816 ± 0.414	2.475 ± 0.363	3.101 ± 0.433	3.425 ± 0.449
CBF	2.802 ± 0.467	4.215 ± 0.729	3.238 ± 0.543	2.819 ± 0.474	3.541 ± 0.526	3.972 ± 0.621
TwoPatterns	3.612 ± 0.662	5.228 ± 1.064	3.715 ± 0.679	4.843 ± 1.289	5.5 ± 0.828	5.419 ± 0.774
ACSF1	4.452 ± 0.5	8.8 ± 1.022	5.758 ± 0.704	4.804 ± 0.427	4.65 ± 0.704	5.609 ± 0.582
Earthquakes	4.759 ± 1.075	6.754 ± 1.28	5.014 ± 1.084	5.886 ± 0.77	4.892 ± 1.175	5.333 ± 1.025
SyntheticControl	5.014 ± 1.957	7.801 ± 3.306	5.77 ± 2.374	5.087 ± 1.906	6.446 ± 2.42	7.577 ± 2.936
SmoothSubspace	9.628 ± 2.871	12.584 ± 4.351	10.064 ± 3.152	17.437 ± 4.913	11.338 ± 3.272	15.214 ± 3.689
ShapeletSim	12.409 ± 0.808	19.569 ± 1.165	14.394 ± 0.767	12.409 ± 0.808	15.407 ± 0.78	17.486 ± 0.962

REFERENCES

[1] F. Eichinger, P. Efros, S. Karnouskos, and K. Böhm, "A time-series compression technique and its application to the smart grid," *VLDB J.*, vol. 24, no. 2, pp. 193–218, Apr. 2015.

[2] J. Azar, A. Makhoul, M. Barhamgi, and R. Couturier, "An energy efficient IoT data compression approach for edge machine learning," *Future Gener. Comput. Syst.*, vol. 96, pp. 168–175, Jul. 2019.

[3] S. Aljanabi and A. Chalechale, "Improving IoT services using a hybrid fog-cloud offloading," *IEEE Access*, vol. 9, pp. 13775–13788, 2021, doi: 10.1109/ACCESS.2021.3052458.

[4] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2001, pp. 289–296.

[5] D. Blalock, S. Madden, and J. Guttag, "Sprintz: Time series compression for the Internet of Things," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 2, no. 3, pp. 1–23, Sep. 2018.

[6] R. N. Bracewell and R. N. Bracewell, *The Fourier Transform and Its Applications*, vol. 31999. New York, NY, USA: McGraw-Hill, 1986.

[7] A. Graps, "An introduction to wavelets," *IEEE Comput. Sci. Eng.*, vol. 2, no. 2, pp. 50–61, Jun. 1995.

[8] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proc. 8th ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery (DMKD)*, 2003, pp. 2–11.

[9] J. W. Lin, S. Liao, and F. W. Leu, "Sensor data compression using bounded error piecewise linear approximation with resolution reduction," *Energies*, vol. 12, no. 13, p. 2523, Jun. 2019.

[10] M. Sifuzzaman, M. R. Islam, and M. Ali, "Application of wavelet transform and its advantages compared to Fourier transform," *J. Phys. Sci.*, vol. 13, no. 1, pp. 121–137, 2009.

[11] S. A. A. Karim, M. H. Kamarudin, B. A. Karim, M. K. Hasan, and J. Sulaiman, "Wavelet transform and fast Fourier transform for signal compression: A comparative study," in *Proc. Int. Conf. Electron. Devices, Syst. Appl. (ICEDSA)*, Apr. 2011, pp. 280–285, doi: 10.1109/ICEDSA.2011.5959031.

[12] J. Ledolter, "Smoothing time series with local polynomial regression on time," *Commun. Statist.-Theory Methods*, vol. 37, no. 6, pp. 959–971, Feb. 2008.

[13] C. Yang, X. Zhang, C. Zhong, C. Liu, J. Pei, K. Ramamohanarao, and J. Chen, "A spatiotemporal compression based approach for efficient big data processing on cloud," *J. Comput. Syst. Sci.*, vol. 80, no. 8, pp. 1563–1583, Dec. 2014.

[14] T. Pelkonen, S. Franklin, J. Teller, P. Cavallaro, Q. Huang, J. Meza, and K. Veeraraghavan, "Gorilla: A fast, scalable, in-memory time series database," *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1816–1827, Aug. 2015.

[15] R. Vestergaard, D. E. Lucani, and Q. Zhang, "A randomly accessible lossless compression scheme for time-series data," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 2145–2154, doi: 10.1109/INFOCOM41043.2020.9155450.

[16] A. Jain and E. Y. Chang, "Adaptive sampling for sensor networks," in *Proc. 1st Int. Workshop Data Manage. Sensor Netw., Conjunct VLDB*, 2004, pp. 10–16.

[17] H. A. Dau, A. Bagnall, K. Kamgar, C.-C.-M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The UCR time series archive," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 6, pp. 1293–1305, Nov. 2019.

[18] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," *Bull. IEEE Comput. Soc. Tech. Committee Data Eng.*, vol. 36, no. 4, pp. 1–12, 2015.

[19] M. J. Watson, A. Liakopoulos, D. Brzakovic, and C. Georgakis, "A practical assessment of process data compression techniques," *Ind. Eng. Chem. Res.*, vol. 37, no. 1, pp. 267–274, Jan. 1998.

[20] S. Steinarsson, "Downsampling time series for visual representation," Ph.D. dissertation, Univ. Iceland, Reykjavík, Iceland, 2013.

[21] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl, "M4: A visualization-oriented time series data aggregation," *Proc. VLDB Endowment*, vol. 7, no. 10, pp. 797–808, Jun. 2014.

[22] P. Bae, K.-W. Lim, W.-S. Jung, and Y.-B. Ko, "Practical implementation of m4 for Web visualization service," *J. Commun. Netw.*, vol. 19, no. 4, pp. 384–391, Aug. 2017, doi: 10.1109/JCN.2017.000062.

[23] E. Fink and H. S. Gandhi, "Compression of time series by extracting major extrema," *J. Exp. Theor. Artif. Intell.*, vol. 23, no. 2, pp. 255–270, Jun. 2011, doi: 10.1080/0952813X.2010.505800.

[24] J. Liu, F. Chen, and D. Wang, "Data compression based on stacked RBM-AE model for wireless sensor networks," *Sensors*, vol. 18, no. 12, p. 4273, Dec. 2018.

[25] K. Kaindl and B. Steipe, "Metric properties of the root-mean-square deviation of vector sets," *Acta Crystallograph. A, Found. Crystallogr.*, vol. 53, no. 6, p. 809, 1997.

[26] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?" *Geosci. Model Develop. Discuss.*, vol. 7, no. 1, pp. 1525–1534, Feb. 2014.

[27] K. Jaskolka and A. Kaup, "Joint optimization of rate, distortion, and maximum absolute error for compression of medical volumes using HEVC intra," in *Proc. Picture Coding Symp. (PCS)*, Jun. 2018, pp. 126–130.

[28] T. Kimura, T. Kimura, A. Matsumoto, and K. Yamagishi, "Balancing quality of experience and traffic volume in adaptive bitrate streaming," *IEEE Access*, vol. 9, pp. 15530–15547, 2021, doi: 10.1109/ACCESS.2021.3052552.

AMAIA GIL received the degree in mathematics from the Faculty of Science and Technology, University of the Basque Country, Leioa, Spain, in 2014, and the master's degree in industrial mathematics from the Technical University of Madrid, Madrid, in 2016. She is currently pursuing the Ph.D. degree in computer science with the focus on the optimization of preprocessing steps for machine learning. She developed her End of Master Degree Project with Cidetec, about physical modeling of lithium pouch cells, in 2016. Since May 2016, she has been working with the Vicomtech Research Center, where she developed projects of the Smart Environment and Energy oriented to machine learning, visualization, and data science.

MARCO QUARTULLI received the Laurea degree in physics from the University of Bari, Italy, in 1997, and the Ph.D. degree in electrical engineering and computer sceince from the University of Siegen, Germany, in 2005. From 1997 to 2010, he worked on remote sensing ground segment engineering, image analysis, archives, and mining with Advanced Computer Systems, Italy. From 2000 to 2003, he was with the German Aerospace Center (DLR), Image Analysis Group, Remote Sensing Technology Institute, Germany, where he worked on metric resolution synthetic aperture radar image understanding in urban environments and content-based image retrieval. Since 2010, he has been with the Vicomtech, where he works in the Data Intelligence for Energy, Department of Industry and Environment.

IGOR G. OLAIZOLA received the degree in electronics engineering from the University of Navarra, Spain, in 2001, and the Ph.D. degree from the University of the Basque Country (UPV/EHU), Spain, in 2013. He is currently the Head of the Department of Data Intelligence for Industry-Energy, and Environment, Vicomtech, Spain. He has participated in more than 30 Research and Development Projects in the area of media technologies, signal/data analysis, and machine learning. Since 2013, he has been an Invited Lecturer with the University of Navarra. His current research interests include signal processing and methodologies to apply data exploitation techniques in industrial processes.

BASILIO SIERRA is currently a Full Professor with the Computer Sciences and Artificial Intelligence Department, University of the Basque Country. He is also the Co-Director of the Robotics and Autonomous Systems Group, Donostia-San Sebastian. He is also a Researcher in robotics and machine learning, where he is working on the use of different paradigms to improve behaviours. His research interests include machine learning, data analysis, computer vision, and robotics.

applied sciences

MDPI

Article

Learning Optimal Time Series Combination and Pre-Processing by Smart Joins

Amaia Gil [1,*], Marco Quartulli [1], Igor G. Olaizola [1] and Basilio Sierra [2]

[1] Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Mikeletegi 57, 20009 Donostia-San Sebastián, Spain; mquartulli@vicomtech.org (M.Q.); iolaizola@vicomtech.org (I.G.O.)

[2] Department of Computer Sciences and Artificial Intelligence, University of the Basque Country (UPV/EHU), 20018 Donostia-San Sebastián, Spain; b.sierra@ehu.es

* Correspondence: agil@vicomtech.org

Received: 28 July 2020; Accepted: 8 September 2020; Published: 11 September 2020

Abstract: In industrial applications of data science and machine learning, most of the steps of a typical pipeline focus on optimizing measures of model fitness to the available data. Data preprocessing, instead, is often ad-hoc, and not based on the optimization of quantitative measures. This paper proposes the use of optimization in the preprocessing step, specifically studying a time series joining methodology, and introduces an error function to measure the adequateness of the joining. Experiments show how the method allows monitoring preprocessing errors for different time slices, indicating when a retraining of the preprocessing may be needed. Thus, this contribution helps quantifying the implications of data preprocessing on the result of data analysis and machine learning methods. The methodology is applied to two case studies: synthetic simulation data with controlled distortions, and a real scenario of an industrial process.

Keywords: optimization; machine learning; preprocessing

1. Introduction and Description of the Problem

In machine learning there are several steps to follow in order to perform model construction. Many of them, such as feature selection, feature extraction and model training, are based on mathematical optimization. However, the initial preprocessing is often not explicitly and quantitatively optimized.

In preprocessing, one of the main steps consists of obtaining all the features that will be used in model generation. The features can come from different origins and joining all the data adequately can be hard. The specific case of working with time series has the advantage of the use of a temporal reference system, a timeline that enables merging the observations. Nevertheless, each feature has its own sampling, and all of them should be resampled to construct a single multi-variate time series in a synchronized way.

This resampling is done by feature, and, depending on the application objectives, different characteristics of data joining methods should be taken into account. Examples of objective measures of preprocessing quality might be based on measures of distortion and on the information lost in the process. Further considerations might be related to the causal nature of the resulting system, or to the amount of delay (or anticipation, in case of being shifted to a prior time instance) applied to the different original series to synchronize them.

Note that these properties can have different degrees of practical importance depending on the application domain. On the one hand, in the case of real-time prediction, data anticipation can imply the need of waiting for a new data entrance, resulting a big delay in the prediction; obviously a data prediction approach based on time series analysis could be used to avoid this problem, using a

correction method in case a significant difference between predicted and real values is detected. On the other hand, information loss and data distortion can have a significant impact on the predictive power of the model.

1.1. Background

In the context of SQL database engines [1], a time series is a sequence of data values measured at successive, though not necessarily regular, points in time (https://cloud.ibm.com/docs/sql-query?topic=sql-query-ts_intro—IBM Cloud SQL Query documentation). Each entry in a time series is called an observation. Each observation comprises a timetick that indicates when the observation was made, and the data recorded for that observation. The set of timeticks defines a temporal base or temporal reference system for the series.

A temporal join is a join operation that operates on time series data. It produces a single array time series based on the original input data and the new temporal reference system.

This section introduces a range of common SQL joining methodologies. Specifically, the following methods will be introduced: left join, nearest join, forward join and backward join. Notably, the outer join is not contemplated, as the obtained results are the same as those from other selected methods (backward join or forward join) depending on the selection of a function for filling in Non-Available (NA) values (ffill or bfill).

In order to simplify the explanation of these methods, a specific example will be used, together with terminology from the documentation of the widely adopted `pandas` (https://pandas.org—Python data analysis and manipulation tool) data analysis library. Suppose that sensor data $y(t_O)$ is acquired with the temporal reference system t_O as shown in Table 1a. For model learning, suppose the temporal reference system t_D shown in Table 1b is required.

Table 1. Problem definition. (**a**) Captured data. (**b**) Desired temporal reference system.

(**a**)	
t_O	y
10:00	y_1
10:02	y_2
10:16	y_3
10:27	y_4
(**b**)	
t_D	$\hat{y}$
10:00	
10:15	
10:30	

Finally, suppose the function *ffill* is selected for filling NA values, and that this function operates by forward-filling such NA values with the nearest prior known data value.

1.1.1. Left Join

The left join method takes only samples from y that are synchronized with t_D, in other words, only data that originally had the desired time is used. Table 2a shows the application of a left join to the example. After filling NA values, the results shown in Table 2b are obtained.

In this particular example, three samples from y are not taken into account in the joined dataset. In this sense, part of the information in the original data is lost.

Table 2. Result of left join. (**a**) After applying the join. (**b**) After filling Non-Available (NA) values.

(**a**)	
t_D	$\hat{y}$
10:00	y_1
10:15	NA
10:30	NA
(**b**)	
t_D	$\hat{y}$
10:00	y_1
10:15	y_1
10:30	y_1

1.1.2. Nearest Join

A nearest join takes into account the nearest known available data from y. Results from the join are shown in Table 3.

Table 3. Result nearest join.

t_D	$\hat{y}$
10:00	y_1
10:15	y_3
10:30	y_4

Depending on the distribution of y, future knowledge of future data can be added to the past in a non-causal manner. In the example, the joined series at 10:15 uses data from 10:16.

1.1.3. Forward Join

In a forward join, samples of t_D that are not available in t_O are selected using subsequent matches from y. Results from the join are shown in Table 4a, and after filling NA values in Table 4b.

Table 4. Result of forward join. (**a**) After applying the join. (**b**) After filling NA values.

(**a**)	
t_D	$\hat{y}$
10:00	y_1
10:15	y_3
10:30	NA
(**b**)	
t_D	$\hat{y}$
10:00	y_1
10:15	y_3
10:30	y_3

1.1.4. Backward Join

In a backward join, samples of t_D that are not available in t_O are selected using the nearest prior match. Results from the join are shown in Table 5.

Given the above existing methods, the remainder of this paper considers the problem of locally selecting an optimal method by the optimization of a quantitative measure of the quality of the obtained joined series.

Table 5. Result of backward join.

t_D	$\hat{y}$
10:00	y_1
10:15	y_2
10:30	y_4

1.2. Paper Contributions and Structure

We consider that the need to define an operational mechanism to align multiple time series with a different time base by optimizing of a cost function that can be defined by the user is not adequately addressed in the present literature. In this sense, the contributions put forward by this paper include:

- The idea that the preprocessing steps in a machine learning workflow can be subject to an optimization procedure that is similar to the one used with e.g., an empirical risk estimate in the actual model learning step.
- The idea that a join operation among tables representing time series with different time bases as operated by e.g., a SQL database engines can be learned based on previous data records.
- A specific algorithm and implementation for a method meant to align multiple time series with different time bases.

The rest of the paper is structured as follows. Section 2 introduces the state of the art approaches. The methodology and a proposed solution are explained in Section 3. Section 4 provides a description of the case studies, whereas Section 5 shows the results of those case studies. Finally, conclusions and future work are presented in Section 6.

2. State of the Art

A number of contributions have been put forward in the literature that deal with the need to align of time series. On the one hand, such a need could stem from the fact that the time series described related phenomena with "warped" temporal aspects (as in Dynamic Time Warping). On the other hand, such a need could depend on the fact that the time series suffer from the effects of different decimation processes (as in the literature related to Dynamic Processes).

In the first group, Folgado et al. [2] considered an extension of Dynamic Time Warping based on a distance which characterized the degree of time warping between two sequences meant for applications where the timing factor is essential, and proposed a Time Alignment Measurement, which delivered similarity information on the temporal domain.

Morel et al. [3] extended Dynamic Time Warping to sets of signals. A significant point with respect to the topic of the present paper is the definition of a tolerance that takes into account the admissible variability around the average signal.

One of the nearest related topics is trying to solve, at the same time, several goals, or to deal with several constraints in parallel. In this sense, there are some works which tackle scheduling problems; a review of this type of models in a practical problem related to flow shop scheduling is presented by Sun et al. [4]. The authors stated that that heuristic and meta-heuristic methods and hybrid procedures were proven much more useful than other methods in large and complex situations.

Tawhid and Savsani [5] proposed a novel multi-objective optimization algorithm named multi-objective sine–cosine algorithm (MO-SCA) which was based on the search technique of the sine–cosine algorithm. They ended obtaining different non-discriminatory levels to preserve the diversity among the set of solutions.

Task scheduling is another problem related to this paper requiring multi-objective optimization paradigms. Zuo et al. [6] presented a solution based on an Ant Colony approach to deal with Cloud Computing computational load and storage minimization. In the same direction, Zahedi et al. [7] presented an approach related to vehicle routing for goods distributions in emergency situations.

The data from a 2017 big earthquake in India was used, considering the demands heterogeneity and dynamics, distribution planning of goods and routing of vehicles simultaneously by means of a genetic algorithm.

Finally, regarding forecasting, Yang et al. [8] presented a system based on a dual decomposition strategy and multi-objective optimization for electricity price forecasting with the goal of balancing electricity generation and consumption. Data pre-processing was fundamental in the selected time window.

3. Proposed Solution: Smart Join

In this paper, a smart join method based on an optimization process is proposed. The aim of this optimization problem is to select the method that minimizes the errors of the resampling process for each feature.

First, a detailed explanation is presented in Section 3.1 and an example of application is shown afterwards in Section 3.2.

3.1. Description of The Methodology

The general concept of the methodology of the smart join is explained next:

1. First, the joining model is fitted using training data; in other words, the optimal joining solution of the process is obtained. This needs to be done for each feature separately.
2. Then, resampled data is predicted by applying the selected join method to the test data.
3. Finally, the model is validated using resampling error.

Suppose we have a time series slice y of the selected feature that needs to be resampled to be joined with a desired time index. First of all, the fit method is used in order to obtain the "optimal" join method. The inputs needed for the join are the original time series slice (y with the original time index) and the desired time index. Other optional parameter can be a fill NA function as it can affect selecting the "optimal" method. Then, another slice of the same feature (z) is used for the testing by the use of the method score. Finally, the optimal joined method is used for resampling other time slices of the features with the predict method. The structure of the different methods can be depicted as in Figure 1.

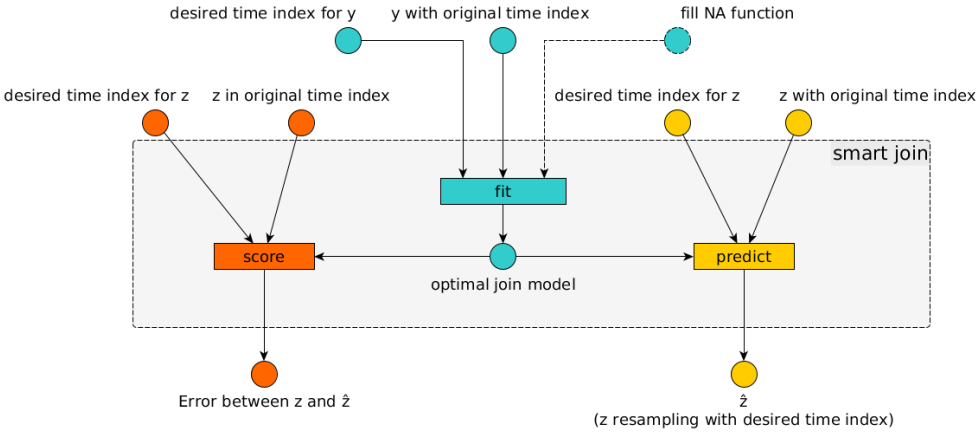


Figure 1. Smart join methodology implementation structure. Firstly fit method is used for the selection of the optimal join method and then, predict and score methods are used to resample other slices of the time series and in order to control the error produced by the join.

The fitting process to find an optimal joining model could be mathematically represented as follows:

Suppose we have the time series $y(t_O)$ where $t_O = [t_{O1}, t_{O2}, \ldots, t_{Om}]$ is the initial temporal reference system. Let j be a join method from the available methods set J (*left*, *backward*, *forward*, *nearest*). We need to obtain a new time series $\hat{y}(t_D; j, y)$ with the desired temporal reference system $t_D = [t_{D1}, t_{D2}, \ldots, t_{Dn}]$. The smart join algorithm aims to find the optimal join method $j \in J = \{left, backward, forward, nearest\}$ that minimizes an error function $E(y, \hat{y})$. The parameters for applying the smart join method are the function meant to fill unavailable measurement values $f \in F = \{None, bfill, ffill, nearest\}$. In case of not being specified, default values will be used (in which case $f = None$). The possible values of the imputation function f are *None* (not filling NA values), *bfill* (using subsequent value that is nearest) and *ffill* (using prior value that is nearest). The optimization problem is defined as:

$$\arg\min_{j \in J} E(y, \hat{y}) \tag{1}$$

With respect to the second contribution put forward by the present paper, the error function $E(y, \hat{y})$ proposed is defined by Equation (2).

$$\begin{aligned} E(y, \hat{y}) = & w_1 \cdot NaEl(\hat{y}) + w_2 \cdot MissEl(y, \hat{y}) + w_3 \cdot DelEl(y, \hat{y}) \\ & + w_4 \cdot DelT(y, \hat{y}) + w_5 \cdot AntEl(y, \hat{y}) + w_6 \cdot AntT(y, \hat{y}) \\ & + w_7 \cdot Diff(y, \hat{y}), \end{aligned} \tag{2}$$

where $w_i > 0$ with $i \in \{1, 2, \ldots, 7\}$ and $\sum_{i=1}^{7} w_i = 1$ are the weights for the total error calculation and, in case of not being specified, their default value is $w_i = 1/7 \; \forall i$.

In the following paragraphs, each function that takes part in the error $E(y, \hat{y})$ is presented. Suppose $k \in \{1, 2, \ldots, n\}$ and $l \in \{1, 2, \ldots, m\}$ indicate the index of elements in $\hat{y}$ and y respectively.

$NaEl(\hat{y})$ represents the percentage of NA elements of $\hat{y}$ after the application of f. NA values can be problematic in machine learning applications implying for example the need to remove data points with NA value on the model training process or the impossibility to predict an output value using the trained model.

$$NaEl(\hat{y}) = \frac{\sum_{k=1}^{n} s_k}{n}, \text{ where } s_k = \begin{cases} 1 & \text{if } \hat{y}_k \text{ is NA} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$MissEl(y, \hat{y})$ is the percentage of elements from y that are not used in $\hat{y}$. This value is related to the lost of information from the original time series due to the resampling needed.

$$MissEl(y, \hat{y}) = \frac{\sum_{l=1}^{m} x_l}{m}, \text{ where } x_l = \begin{cases} 1 & \text{if } y_l \notin \hat{y} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$DelEl(y, \hat{y})$ indicates the percentage of delayed elements. If most of the data points from y are delayed, the reality for the machine learning model is displaced. Depending on the application environment, taking decisions supported by the machine learning system that could not adequately represent the current situation can be problematic.

$$DelEl(y, \hat{y}) = \frac{\sum_{k=1}^{n} d_k}{n}, \text{ where } d_k = \begin{cases} 1 & \text{if } (\hat{y}_k = y_l) \text{ and } (t_{Dk} > t_{Ol}) \; \forall l \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$DelT(y, \hat{y})$ is the maximum difference in time between a delayed element used in $\hat{y}$ and its original time position normalized by the time window of y. Whereas the previous case considers the frequency of delayed elements, $DelT(y, \hat{y})$ takes into account the magnitude of the displacement.

$$DelT(y, \hat{y}) = \frac{\max(e_k)}{t_{Om} - t_{O1}}, \text{where } e_k = \begin{cases} t_{Dk} - t_{Ol} & \text{if } (\hat{y}_k = y_l) \text{ and } (t_{Dk} > t_{Ol}) \; \forall l \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$AntEl(y, \hat{y})$ and $AntT(y, \hat{y})$ are equivalent functions but in this case for anticipated elements.

$$AntEl(y, \hat{y}) = \frac{\sum_{k=1}^{n} a_k}{n}, \text{where } a_k = \begin{cases} 1 & \text{if } (\hat{y}_k = y_l) \text{ and } (t_{Dk} < t_{Ol}) \ \forall l \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

and

$$AntT(y, \hat{y}) = \frac{\max(b_k)}{t_{Om} - t_{O1}}, \text{where } b_k = \begin{cases} t_{Ol} - t_{Dk} & \text{if } (\hat{y}_k = y_l) \text{ and } (t_{Dk} < t_{Ol}) \ \forall l \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

On the one side, the use of anticipated data is equivalent to the use of future information for prediction and results can be misleading and the used approximation should be sound enough to deal with value forecasting. On the other side, using future data could imply a need to wait for the arrival of a new observation to be able to make a prediction, or a correction would be needed once the predicted value and the real one are compared.

Finally $Diff(y, \hat{y})$ calculates the difference between the two time series (original and resampled). This value could represent the magnitude of the distortion committed due to the need of a joined data with synchronized temporal reference system.

$$Diff(y, \hat{y}) = \frac{\text{mean}(\text{abs}(y_{inter} - \hat{y}_{inter}))}{\max(y) - \min(y)}, \tag{9}$$

where y_{inter} and $\hat{y}_{inter}$ are obtained by means of linear interpolation of time series y and $\hat{y}$ respectively for time values in $t_O \bigcup t_D$.

Each part of the sum of the error calculation Equations (3)–(9) is normalized to guarantee that the result is in range $[0, 1]$ so different errors are comparable between them.

The fitting method can be seen graphically in Figure 2.

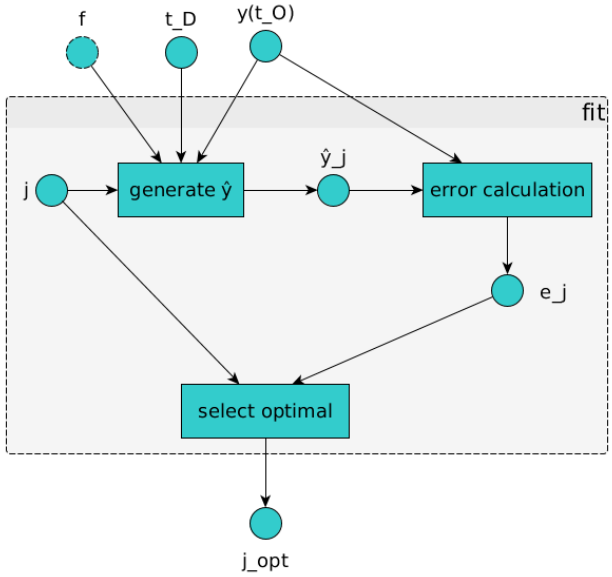


Figure 2. Fitting method diagram. First, $\hat{y}_j$ resampled time series are generated from each joining method ($j \in J$). Using the generated resampled time series, error is calculated in each case and the optimal solution is selected j_{opt}.

Validating the joined method in different time slices of the time series is crucial. If the slice of data used to train the joining model is adequately selected, the errors should be similar in different time

windows. Depending on the stability of the feature, retraining may be required as the optimal join method could not be the most adequate during all time period. Furthermore, selecting the desired temporal reference system (t_D) has equal importance as it should be the same for all the features, in order to be able to construct a database with all the features used by the model. Although the error calculation and the optimal joining methodology is chosen separately per feature, the desired temporal reference system is a common input of all the optimization problems and its selection affects to all the features.

3.2. Application Example

The current subsection introduces an illustrative example of the application of the proposed method to a dataset from a simple piecewise function. Suppose that the piecewise function is sampled irregularly in order to save memory applying two criteria:

- The system checks every minute if the value of the data point has changed enough according to a pre-established criterion (in this particular case, a difference with the prior data point higher than 0.5) to save that data point.
- Every minute the system also checks the difference in time with the last saved data point and if this difference is greater than or equal to four minutes it saves the last available data point.

The original piecewise function and the saved data using these criteria are shown in Figure 3.

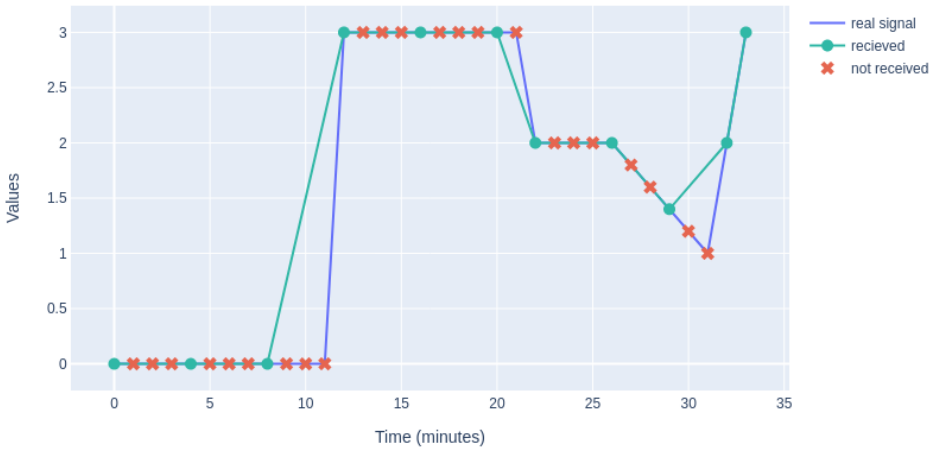


Figure 3. Application example problem.

Suppose that the desired time reference system corresponds to $t_d = \{1, 3, 5, \ldots, 33\}$. Results after the application of different joining methods are shown in Figure 4. Error values used in the optimization of the Smart Join methodology are shown in Table 6.

Because the input for the algorithm is the received data, when default weights in the error function are used ($w_i = 1/7\ \forall i$), the minimal error is obtained by the nearest join (see Figure 4b). However, if knowledge about the irregular sampling approach used by the system is introduced by penalizing the anticipation of data points (for example with $w_5 = w_6 = 2/9$ and $w_1 = w_2 = w_3 = w_4 = w_7 = 1/9$), the optimal join method is backward join. Figure 4d shows that the data points obtained by the backward join as a result of taking into account this extended description of the data sampling mechanism are the ones that are the closest to the real piecewise function.

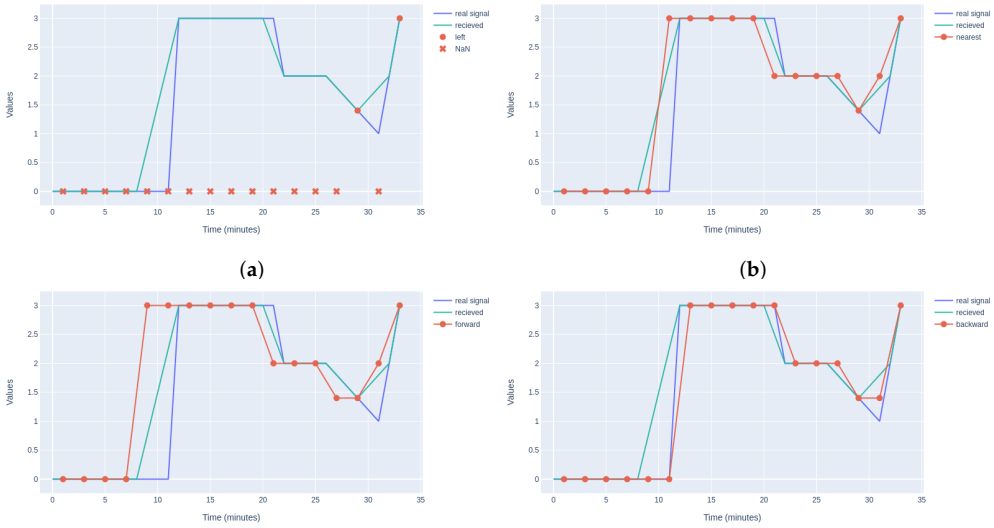


Figure 4. Application example results for different joining methods. (**a**) Left join. (**b**) Nearest join. (**c**) Forward join. (**d**) Backward join.

Table 6. Error values for different join methods in the application example.

Method	$NaEl(\hat{y})$	$MissEl(y,\hat{y})$	$DelEl(y,\hat{y})$	$DelT(y,\hat{y})$	$AntEl(y,\hat{y})$	$AntT(y,\hat{y})$	$Diff(y,\hat{y})$
left	0.882	0.818	0.0	0.0	0.0	0.0	1.0
nearest	0.0	0.0	0.471	0.031	0.412	0.031	0.045
forward	0.0	0.091	0.0	0.0	0.882	0.094	0.092
backward	0.0	0.091	0.882	0.094	0.0	0.0	0.084

Having established the significance of the measure of quality of a joining method, in the remainder of this contribution we leverage mathematical optimization techniques on training data to automatically determine which of the joining methods is most adequate for a given time series.

4. Experimental Setup

Two experiments were used in order to show the usefulness of the proposed smart join methodology.

The first one is a controlled application from simulated data and working with a unique time series to resample. Different distortion methods were applied to the data in order to have a practical use case with known theoretical result.

The second case is an application from an industrial chemical process. The aim of showing this example is to demonstrate the performance of the smart join method in a real scenario and the importance of adequately selecting the joining method and its implications.

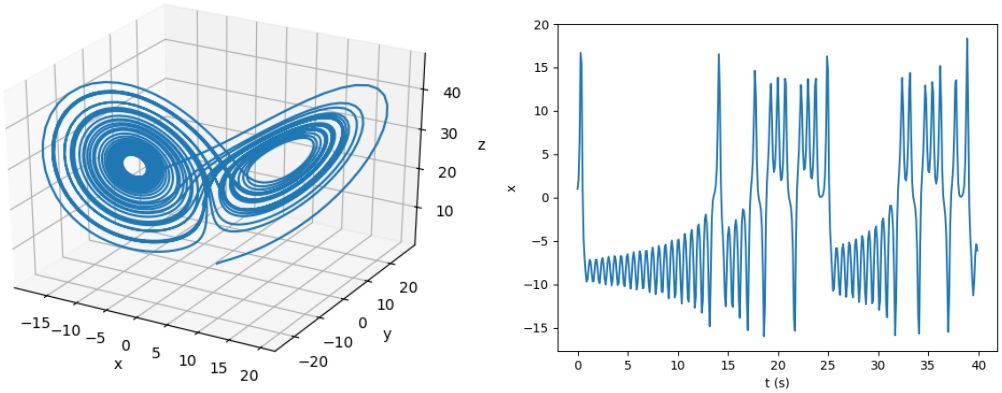
4.1. Experiments on Synthetic Data

The experiments on synthetic data are carried out on the x, y, z 3D curve generated in time t by a Lorenz system (originally a simplified model for atmospheric convection) [9].

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= x(\rho - z) - y \\ \frac{dz}{dt} &= xy - \beta z \end{aligned} \tag{10}$$

with parameters $\sigma = 10, \rho = 28$ and $\beta = 8/3$ and initial conditions $x(0) = y(0) = z(0) = 1$ and $t \in [0, 40]$. The time sampling interval selected for the time series was 0.1 time units.

The simulated data can be observed in Figure 5a. To apply the smart join methodology only dimension x was used. The time series is shown in Figure 5b.



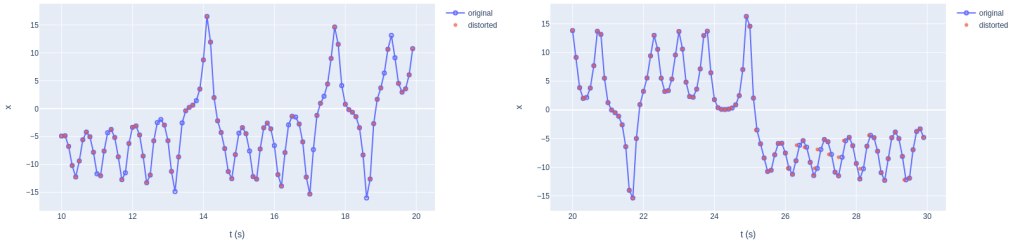
(**a**) (**b**)

Figure 5. Lorenz system result data. (**a**) Three-dimensional data. (**b**) First dimension time series.

In order to generate a distorted version of this time series in a controlled manner, some distortion methods were applied, inspired by from the work of Kreindler and Lumsden [10], which will be described later in the section.

This controlled experiment setup was used to demonstrate how errors change depending on the join method and on the type of distortion that is applied to each time window. The distortions have been selected in order to represent usual problems such as missing data or delays in receiving data points.

The series was divided into four parts of equal size. In the first part ($t \in (0, 10]$) the time series remains unaltered. In the range $t \in (10, 20]$, 20% randomly selected data points were removed. This distortion can be seen in Figure 6a. In the remaining part of the time series, 20% of data were shifted forward (in $t \in (20, 30]$) or backward (in $t \in (30, 40]$). The shifted quantity was selected by a random uniform variable, guaranteeing that data points cannot be disordered. In other words, the maximum possible shifted quantity was set by the sampling frequency value of the original simulation data. The distortion effect generated in the time series can be observed in Figure 6b.



(**a**) (**b**)

Figure 6. Zoomed distortions of Lorenz first dimension. (**a**) Removed data. (**b**) Shifted data.

The difference between modified and original data can be seen in Figure 7.

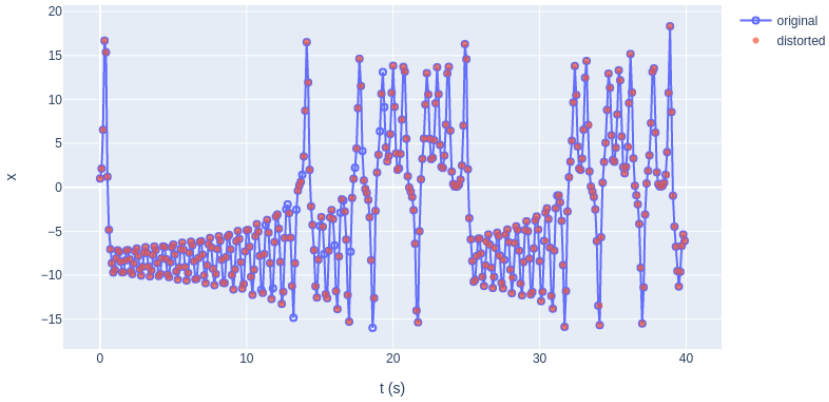


Figure 7. Original vs. distorted Lorenz first dimension.

4.2. Experiments on Real Industrial Dataset

The efficient management and the energy optimization of distillation units [11,12] in terms of both product quality and energy efficiency in both the petro-chemical and in the sustainable sector pose a great challenge to process and control engineers because of their complexity. The management, optimization and fault analysis of such units all require accurate process models, which in recent years have started to be generated directly from the data available in SCADA Historian databases by using machine learning methods [13–15] whose performance depends on the availability of properly pre-processed multi-variate data.

Suppose that the system captures and stores real-time sensor-based data. In this particular case, each sensor writes values in the database only when there is a significant change in the values of data. The decision on the significance of the difference between data points is based on the scale of each feature. The aim of this data recording strategy is reducing data volume. Consequently, if a feature becomes unstable the writing frequency augments drastically.

For machine learning applications, an alignment between features is needed. Each feature should be resampled to obtain a common desired temporal reference system previous to any feature extraction/selection algorithm application. Depending on the feature and the application system, the optimal joining method can be different.

Figure 8 shows the initial sampling for different features. Each column represents a feature and each row an hour time window. The number of samples is counted per hour and feature, and represented by the colour.

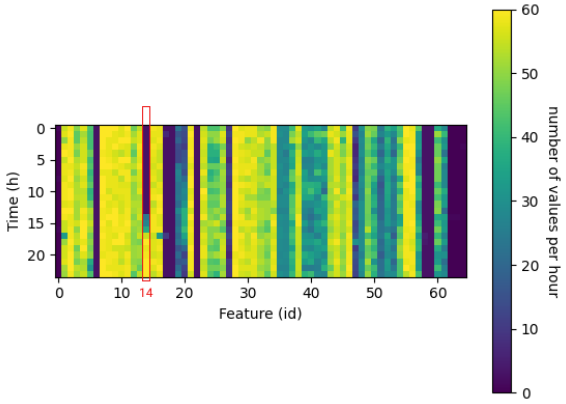


Figure 8. Original sampling of real industrial data from a distillation unit.

Figure 8 shows how, depending on the feature the frequency of data availability can be constant or variable, and the quantity of samples can be very different among features. For example, the feature with id 14 changes drastically from very low frequency to high frequency only in a couple of hours. The data points for this particular feature are shown in Figure 9, where the frequency change is observable.

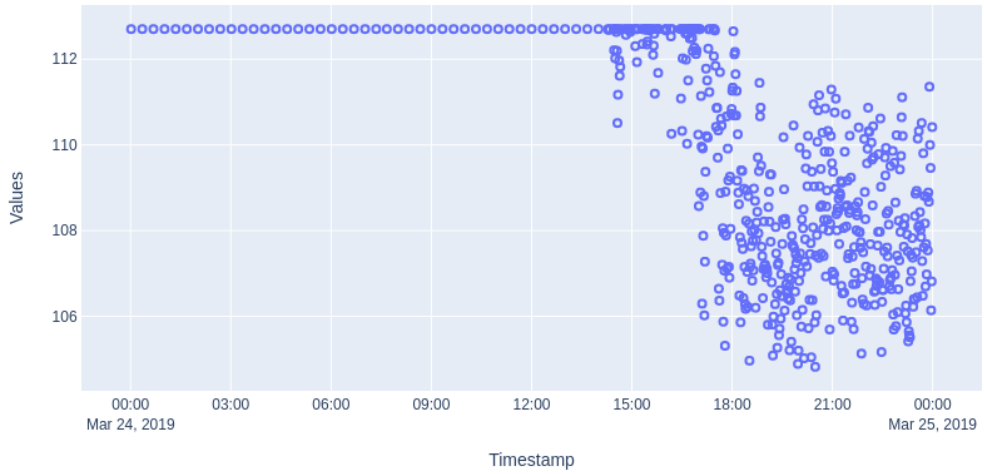


Figure 9. Original data of the feature with id 14 from Figure 8.

In this particular case, the desired time sampling interval is selected to be 15 min.

5. Experimental Results

This section presents experimental results for the aforementioned case studies.

5.1. Results on Synthetic Data

For synthetic data, different time series joining methods were used separately and the error, defined by Equation (2), was calculated for each method using windows of $t \in (p, p+2]$ with $p \in \{0, 2, \ldots, 38\}$. The selected values for the parameters of smart join methodology were $w_i = 1/7\ \forall i$ (i.e., the same importance for all different functions taking part in the error calculation) and the imputation function was $f = ffill$.

Table 7 shows the error values per method and time window. The optimal solution (minimum error) is marked in bold. An additional column labelled "theoretical" represents the theoretically optimal solution. Thus, the obtained optimal solution in each time window can be compared and contrasted with the theoretical solution. In Figure 10 numerical results are shown graphically.

As per Table 7, the optimal joining method (the one that has minimal error in each window) depends on the controlled distortion introduced. The proposed methodology is capable of obtaining as one of the optimal available results the theoretical solution. On the one hand, for $t \in (0, 10]$, the data was already available for the needed temporal reference system and for that reason all the methods were able to obtain a 0.0 value error. On the other hand, for $t \in (10, 20]$, as data points are removed randomly, there was no optimal theoretical solution, as from known data points the joining method should not be able to reconstruct the time series. For this range, the optimal solution for the joining method depends on $Diff(y, \hat{y})$, i.e., the distortion introduced is comparable to the one obtained with the lineal interpolation result. For $t \in (20, 30]$ and $t \in (30, 40]$ the optimal theoretical solutions were backward and forward join, respectively. However, in multiple windows, the nearest join method obtained the same solution as the theoretically optimal method, as the shifted data points

(t_{OI} introduced in the smart join system) are the nearest ones to the desired data points (t_{Dk} output temporal reference system).

Table 7. Results on synthetic data, in bold the method with minimal error (multiple solutions are possible).

t Range	Backward Join	Forward Join	Left Join	Nearest Join	Theoretical
0–2	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	all
2–4	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	all
4–6	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	all
6–8	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	all
8–10	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	**0.00** $\times 10^{0}$	all
10–12	**4.56** $\times 10^{-2}$	**4.56** $\times 10^{-2}$	**4.56** $\times 10^{-2}$	5.32 $\times 10^{-2}$	none
12–14	**5.08** $\times 10^{-2}$	**5.08** $\times 10^{-2}$	**5.08** $\times 10^{-2}$	**5.08** $\times 10^{-2}$	none
14–16	3.35 $\times 10^{-2}$	**2.77** $\times 10^{-2}$	3.35 $\times 10^{-2}$	4.06 $\times 10^{-2}$	none
16–18	4.55 $\times 10^{-2}$	4.55 $\times 10^{-2}$	4.55 $\times 10^{-2}$	**5.35** $\times 10^{-2}$	none
18–20	**4.36** $\times 10^{-2}$	4.83 $\times 10^{-2}$	**4.36** $\times 10^{-2}$	**4.36** $\times 10^{-2}$	none
20–22	**1.89** $\times 10^{-2}$	3.61 $\times 10^{-2}$	3.61 $\times 10^{-2}$	**1.89** $\times 10^{-2}$	backward
22–24	**1.50** $\times 10^{-2}$	3.61 $\times 10^{-2}$	3.61 $\times 10^{-2}$	**1.50** $\times 10^{-2}$	backward
24–26	**1.67** $\times 10^{-2}$	3.61 $\times 10^{-2}$	3.61 $\times 10^{-2}$	**1.67** $\times 10^{-2}$	backward
26–28	**7.69** $\times 10^{-2}$	1.21 $\times 10^{-1}$	1.92 $\times 10^{-1}$	9.39 $\times 10^{-2}$	backward
28–30	3.61 $\times 10^{-2}$	5.86 $\times 10^{-2}$	5.86 $\times 10^{-2}$	**3.00** $\times 10^{-2}$	backward
30–32	5.75 $\times 10^{-2}$	**3.55** $\times 10^{-2}$	7.22 $\times 10^{-2}$	4.44 $\times 10^{-2}$	forward
32–34	5.04 $\times 10^{-2}$	**2.72** $\times 10^{-2}$	5.04 $\times 10^{-2}$	**2.72** $\times 10^{-2}$	forward
34–36	8.90 $\times 10^{-2}$	**5.26** $\times 10^{-2}$	1.11 $\times 10^{-1}$	**5.26** $\times 10^{-2}$	forward
36–38	4.44 $\times 10^{-2}$	**2.86** $\times 10^{-2}$	4.44 $\times 10^{-2}$	**2.86** $\times 10^{-2}$	forward
38–40	3.61 $\times 10^{-2}$	**1.99** $\times 10^{-2}$	3.61 $\times 10^{-2}$	**1.99** $\times 10^{-2}$	forward

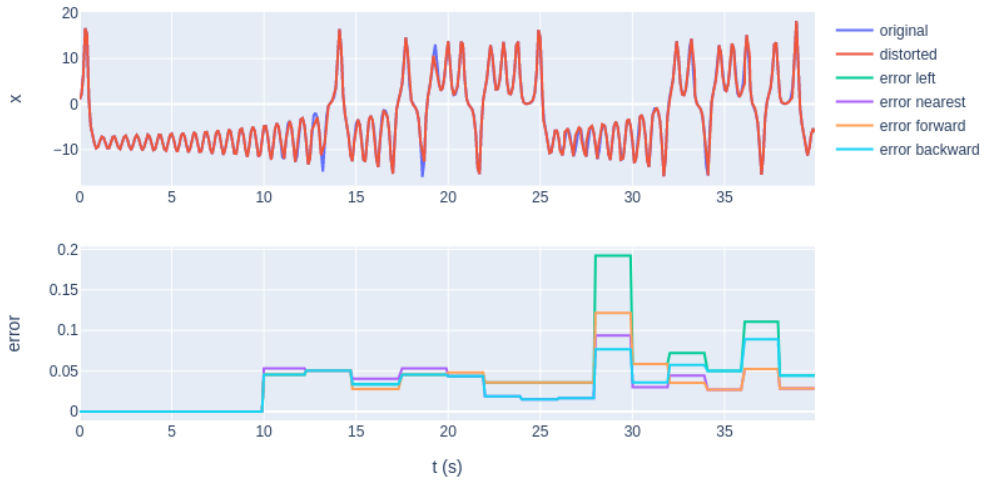


Figure 10. Synthetic data results.

5.2. Results on Real Industrial Datasets

With respect to the real dataset, all features had a common sampling distribution after the joining as per Figure 11. In this case, the common sampling distribution was represented by having the same colour by row for all the features (represented by columns). Furthermore, as the selected temporal reference system (t_D) had a constant sampling interval, the figure results in constant colour (four data points for each feature each hour).

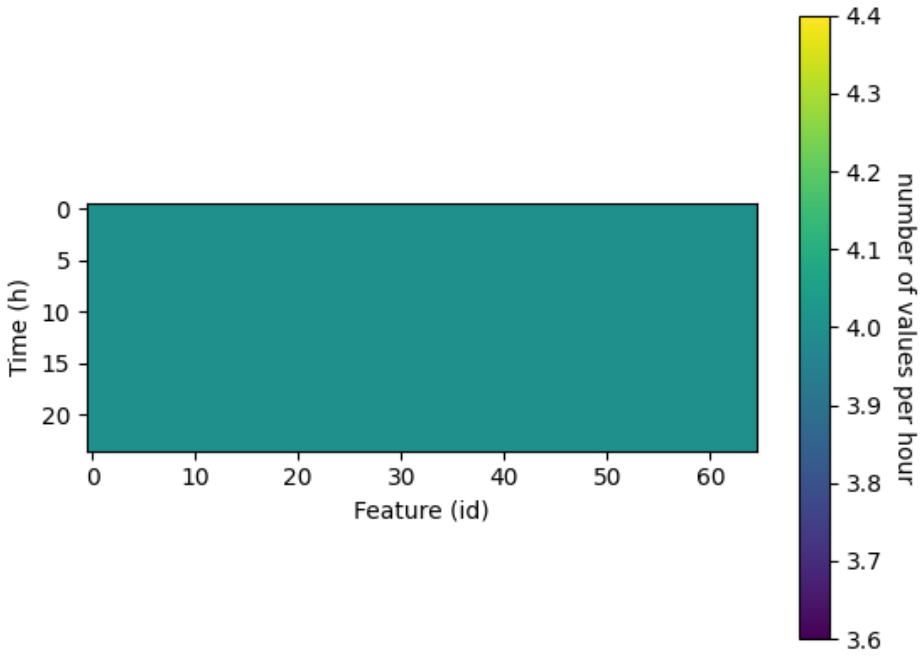


Figure 11. Result after the use of smart join. After the joining, all features have a common sampling distribution.

In Figure 12 the original time series (y) and the one obtained from the joining methodology ($\hat{y}$) are shown for making a visual comparison. Both time series (y and $\hat{y}$) had similar appearance until 16:00 where the feature became unstable. Due to the selected time sampling and the joins considered for finding the optimum being the ones operated by SQL database engines, only a data point near the needed sampling was selected.

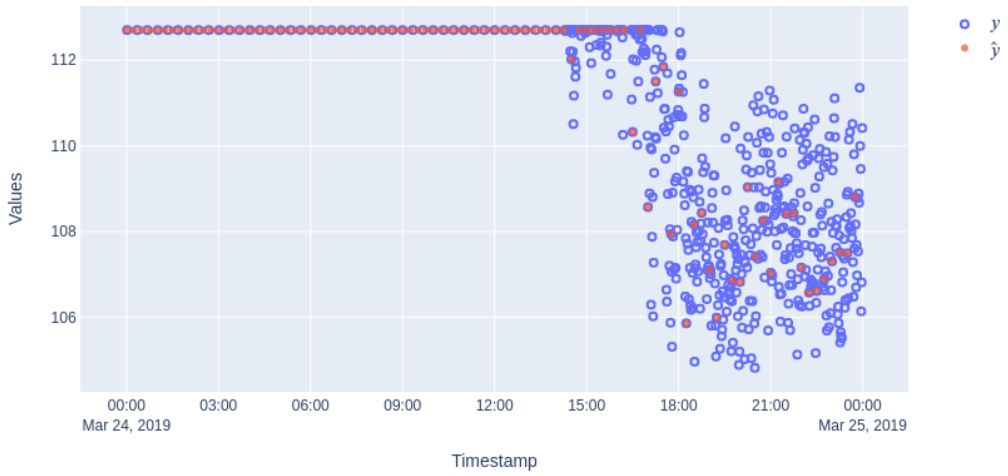


Figure 12. Comparison between original time series and after the use of smart join for the feature with id 14.

Figure 13 shows the alignment distortion for the feature with id 14. Negative values in this misalignment imply that anticipated time data were used in the join, whereas positive values imply

delayed time. The difference in the alignment could imply delays in prediction if anticipated data were used in the join or did not really have updated information of the process in order to make an adequate decision. In this particular case as the original time sampling initially writes nearly each 20 min and the desired time sampling is every 15 min, delays or anticipations of nearly 8 min become common. In the last part of the original time series, as data were available every minute or two, the delays or anticipations are drastically reduced for the joined time series.

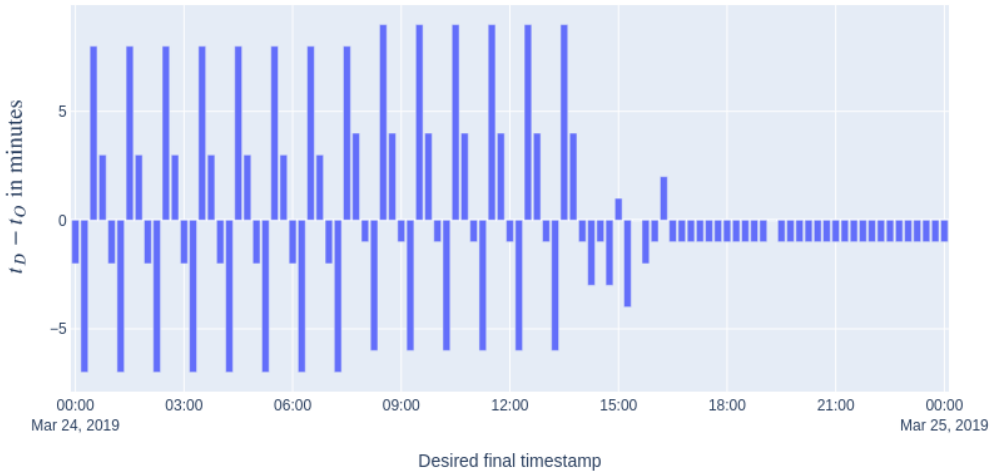


Figure 13. Alignment distortion for feature 14 between y and $\hat{y}$. Negative values in this misalignment imply that anticipated time data has been used in the join, whereas positive values imply delayed time. The difference in the alignment could imply delays in prediction if anticipated data was used in the join or did not really have updated information of the process in order to make an adequate decision.

Figure 14 shows used and unused points from the original time series in the join time series. Depending on the application the lost information could have a great impact. For time later that 16:00, as the selected time sampling (t_D) is slower than the dynamic of the original time series, a lot of data points are unused in the joining process, losing the information provided by those data points showed in blue in the figure. In some cases, different aggregation methods or rolling windows could be more adequate to use the data that otherwise will be lost.

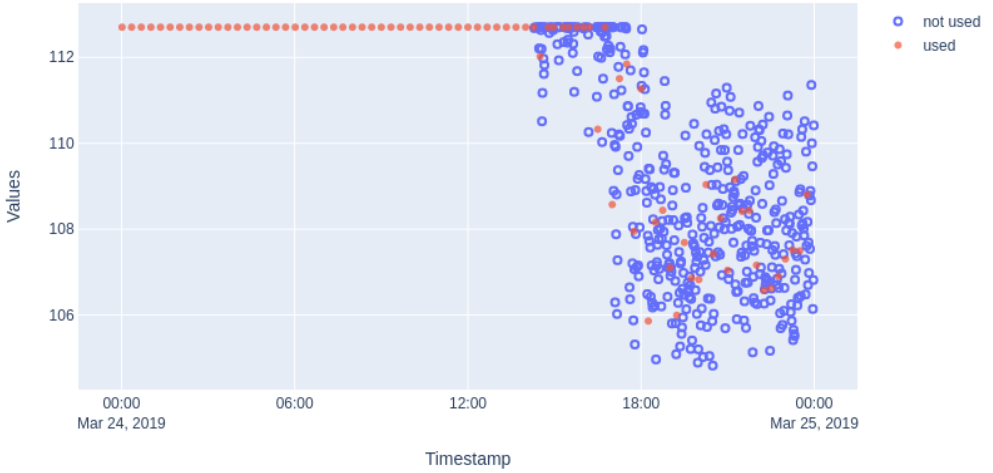


Figure 14. Data used and not used from y to generate $\hat{y}$.

The difference between the original time series and the joined one can help optimize the time sampling for a specific application. At the top of Figure 15 both the time series used for error calculation in part of $Diff(y, \hat{y})$ ($\hat{y}_{interp}$ and y_{interp}, i.e., generated by linear interpolation of time series y and $\hat{y}$ in order to have common data sampling distribution ($t_O \bigcup t_D$)) are shown, while the lower diagram shows the absolute error value calculated at each point. For a comparison of how the frequency selection can affect the desired time sampling, a similar diagram with a desired sampling frequency modified from 15 min to one minute is shown in Figure 16. In both figures, as initially the original time series has constant values, there is no difference between both interpolated time series. However, as time passes by and the time series becomes unstable, the difference is remarkable. This error is greater in Figure 15, as the desired time sampling frequency is slower than the real dynamic of the feature and data is not linear.

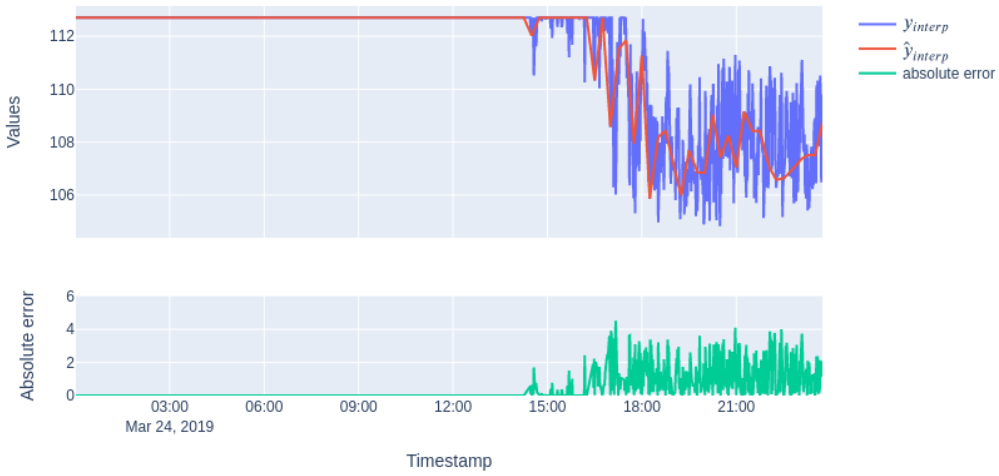


Figure 15. Difference between original data and joined data with a desired time sampling frequency 15 min.

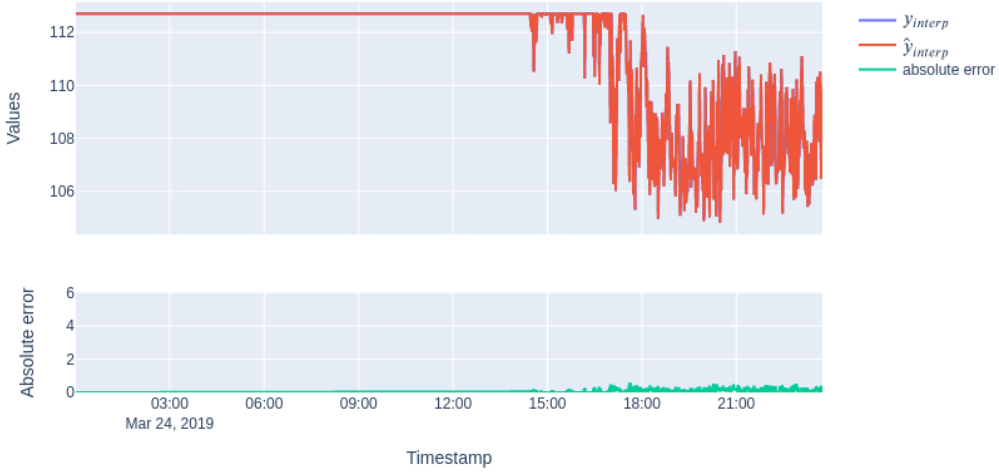


Figure 16. Difference between original data and joined data with desired time sampling frequency 1 min.

In Table 8 the effect in the error of different selections of desired sampling frequency are shown for comparison.

Table 8. Error values for the nearest joining method for different requested sampling frequencies.

Frequency	$NaEl(\hat{y})$	$MissEl(y,\hat{y})$	$DelEl(y,\hat{y})$	$DelT(y,\hat{y})$	$AntEl(y,\hat{y})$	$AntT(y,\hat{y})$	$Diff(y,\hat{y})$
1	0.0	0.494	0.333	0.007	0.667	0.006	0.004
5	0.0	0.851	0.322	0.007	0.678	0.005	0.036
10	0.0	0.903	0.315	0.007	0.685	0.005	0.054
15	0.0	0.921	0.333	0.007	0.667	0.004	0.058

6. Conclusions and Future Work

Standard data analysis pipelines often include resampling, interpolation and aggregation steps that are not optimized in the model learning procedure.

This paper introduced the definition of an optimization problem for data preprocessing, and in particular for data joining processes that imply a need for data resampling. The defined problem has been addressed by a method designed to efficiently solve it. The case studies introduced have demonstrated the applicability of the proposed method to time series data, using standard SQL-like data joining primitives as a basis to be optimized upon. The first case study, with simulated data and controlled distortions, means to provide insight into the methodology and its applicability. In the second experiment, the proposed methodology is applied in a real scenario, showing the impact of the decisions taken in the preprocessing step on the learning of data-based models.

Furthermore, the paper proposed an error function for its use in the optimization problem of joining time series. This error function allows comparisons across different features and time slices, which is needed to select among different join methods or to monitor their quality on different time series slices. As errors are comparable, selecting the optimal solution or knowing when there is a need for retraining is possible. Moreover, using the input parameters (w and f) of the proposed error function allows adapting the function to an adequate solution for different applications.

The approach presented in this paper has several new paths to follow as future works: on the one hand, the approach could be improved, adding automatic selection of the time window size, or applying B-Spline mode approximations of the missing values; on the other hand, the benefits of the proposed Smart Join method should be quantified on a diverse range of real world applications. Energy consumption, storage and production, supply transportation and storage management are candidates towards this end.

Author Contributions: A.G. and M.Q. designed and implemented the experimental testbed and algorithm, B.S. and I.G.O. supervised the experimental design and managed the project. M.Q. and B.S. reviewed the new approach of this research. A.G. performed the experimental phase. All authors contributed to the writing and reviewing of the present manuscript. All authors read and agreed to the published version of the manuscript.

Funding: This research has been partially funded by the 3KIA project (ELKARTEK, Basque Government).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Codd, E.F. *The Relational Model for Database Management*; Addison-Wesley Publishing Company: Boston, MA, USA, 1990.
2. Folgado, D.; Barandas, M.; Matias, R.; Martins, R.; Carvalho, M.; Gamboa, H. Time alignment measurement for time series. *Pattern Recognit.* **2018**, *81*, 268–279. [CrossRef]
3. Morel, M.; Achard, C.; Kulpa, R.; Dubuisson, S. Time-series averaging using constrained dynamic time warping with tolerance. *Pattern Recognit.* **2018**, *74*, 77–89. [CrossRef]
4. Sun, Y.; Zhang, C.; Gao, L.; Wang, X. Multi-objective optimization algorithms for flow shop scheduling problem: A review and prospects. *Int. J. Adv. Manuf. Technol.* **2011**, *55*, 723–739. [CrossRef]
5. Tawhid, M.A.; Savsani, V. Multi-objective sine-cosine algorithm (MO-SCA) for multi-objective engineering design problems. *Neural Comput. Appl.* **2019**, *31*, 915–929. [CrossRef]
6. Zuo, L.; Shu, L.; Dong, S.; Zhu, C.; Hara, T. A Multi-Objective Optimization Scheduling Method Based on the Ant Colony Algorithm in Cloud Computing. *IEEE Access* **2015**, *3*, 2687–2699. [CrossRef]

7. Zahedi, A.; Kargari, M.; Husseinzadeh Kashan, A. Multi-objective decision-making model for distribution planning of goods and routing of vehicles in emergency multi-objective decision-making model for distribution planning of goods and routing of vehicles in emergency. *Int. J. Disaster Risk Reduct.* **2020**, *48*, 101587. [CrossRef]
8. Yang, W.; Wang, J.; Niu, T.; Du, P. A hybrid forecasting system based on a dual decomposition strategy and multi-objective optimization for electricity price forecasting. *Appl. Energy* **2019**, *235*, 1205–1225, [CrossRef]
9. Lorenz, E.N. Deterministic Nonperiodic Flow. *J. Atmos. Sci.* **1963**, *20*, 130–141. [CrossRef]
10. Guastello, S.J.; Gregson, R.A. (Eds.) *Nonlinear Dynamical Systems Analysis for the Behavioral Sciences Using Real Data*; CRC Press Taylor & Francis Group: Abingdon, UK, 2011.
11. Ciric, A.R.; Miao, P. Steady state multiplicities in an ethylene glycol reactive distillation column. *Ind. Eng. Chem. Res.* **1994**, *33*, 2738–2748. [CrossRef]
12. Kumar, A.; Daoutidis, P. Modeling, analysis and control of ethylene glycol reactive distillation column. *AIChE J.* **1999**, *45*, 51–68. [CrossRef]
13. Osuolale, F.N.; Zhang, J. Energy efficiency optimisation for distillation column using artificial neural network models. *Energy* **2016**, *106*, 562–578. [CrossRef]
14. Tehlah, N.; Kaewpradit, P.; Mujtaba, I.M. Artificial neural network based modeling and optimization of refined palm oil process. *Neurocomputing* **2016**, *216*, 489–501. [CrossRef]
15. Mirakhorli, E. Fault diagnosis in a distillation column using a support vector machine based classifier. *Int. J. Smart Electr. Eng.* **2020**, *8*, 105–113.

ASSIST: Automatic Smart Selection of the Suitable Imputation Technique

Amaia Gil[1,2*], Marco Quartulli[1], Igor G. Olaizola[1] and Basilio Sierra[2]

[1]Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Mikeletegi 57, 20009 Donostia-San Sebastián, Spain.
[2]Department of Computer Sciences and Artificial Intelligence, University of the Basque Country (UPV/EHU), 20018 Donostia-San Sebastián, Spain.

*Corresponding author(s). E-mail(s): agil@vicomtech.org;
Contributing authors: mquartulli@vicomtech.org; iolaizola@vicomtech.org; b.sierra@ehu.eus;

Abstract

Dealing with data that is missing due to unknown causes is a common problem in industrial applications. A possibility to mitigate this issue is to effectively impute missing observations that adjust to the actual characteristics of the signal received. This paper proposes a new method called ASSIST for adaptively identifying optimal imputation strategies for time series data on a window-by-window basis. For this purpose, time series and gap distributions are characterized first, and a model is trained from those features in order to predict the best fit imputation strategy from the available ones. This proposed approach is validated in the entire UCR time series public data archive.

Keywords: data imputation, time series, machine learning, adaptive method

1 Introduction and description of the problem

Missing observations in data can be produced by multiple causes such as network communication problems, measurement errors or data acquisition problems. In general, the cause of a missing observation is unknown and its appearance is unpredictable. The data quality and availability are essential for applications where decisions are taken based on data [1–3]. Depending on the relative amount of missing data and the cause or multiple causes generating the missingness, the data obtained can be biased severely affecting performance of algorithms, machine learning models, etc [4, 5].

The processes of filling in the unknown observations is defined as imputation. This research is centered on univariate time series data and single imputation methods, in other words, there is no relation to other time series that can provide information about the actual missing observation being considered and each missing observation has a unique estimation value. Multiple imputation strategies have been proposed in the last years [6–10]. However, as most of these methods are specifically designed for data with certain given characteristics or specific application fields, undesired results can be obtained when applied to real data or previously unconsidered situations [11, 12].

A key challenge of imputation selection is filling the gaps in a controlled manner [13]. The imputation method should adapt to the actual signal and to the missing data pattern. This paper proposes a methodology that is able to select the imputation technique to the observed data and missing observations distribution.

The rest of the paper is structured as follows. Section 2 introduces different imputation methods. Then, the proposed approach is explained in Section 3. Section 4 provides a description of the experiments setup, whereas Section 5 shows the results of those experiments. Finally, conclusions and future work are presented in Section 6.

2 Theoretical background of data imputation

Zhang [14] presents a categorization of imputation methods based on the number of imputations generated for each missing observation: 1) single imputation, 2) multiple imputations, 3) fractional imputation and 4) iterative imputation. Single imputation methods provide a unique estimation for each missing value [15], while multiple imputation method makes multiple estimation values for each missing observation providing uncertainty measures in each case in order to combine them and reach a final imputation value [16]. In the fractional imputation case, several imputed values are generated together with conditional probabilities given a known observation [17]. Finally, iterative imputation techniques are capable of estimating missing values when data points are incomplete and by the use of multiple iterations obtaining refinement in the estimation values [18]. The rest of the Section is centered in the description of different single imputation techniques as this research is focused on their use.

The simplest imputation method is replacing missing observations with a constant value. This value could indicate that the data is missing or be replaced by a statistical property such as mean, median or mode. In the particular case of the time series, missing observations can be replaced with the nearest previous or following data points.

Interpolation methods [19] consist of constructing new observations from a discrete quantity of data points. Using known observations, interpolation function parameters are estimated, and later the interpolated function is evaluated to obtain missing data observations. The simplest interpolation function is the piecewise constant, which locates the nearest value and uses it to construct the function. The linear interpolation, and its generalization the polynomial interpolation, are also commonly used due to their simplicity. In the case of needing smooth functions, spline interpolation can be used. This variant uses lower degree polynomial functions as interpolate functions as in the previous case but this time guaranteeing that the resulting function is differentiable [20]. Akima splines [21] are capable to adjust smooth functions near outliers without creating oscillations due to its non-linearity interpolation function. Piece cubic Hermite interpolating polynomial [22] is capable to adapt better to flat areas compared to spline interpolation, due to slope interpolation differences. However, depending the position of flat areas, this could imply obtaining flat interpolation functions near the local extrema. Kriging [23] is a geostatistical interpolation method that is based on a regression method of Gaussian processes that can be used for point estimation.

In the cases when sufficient historical data is available, time series forecasting models can be generated. Once the model is trained, it is possible to forecast the missing observations and use those predictions to fill the gaps. For this purpose, classical statistical models can be used, such as exponentially weighted moving average [24] or its generalization Holt-Winters [25] model and autoregressive models variants [26]. Additionally, the application of machine learning models, such as decision trees [27], neural networks [28] and support vector machines [29], have been applied with success in time series forecasting. These models should be applied guaranteeing that the time series specific characteristics as temporal relations between observations [30] are met. Furthermore, hybrid models have been applied that include the statistical power of classical models joined to the capability of learning non-linear models of machine learning methods [31, 32].

Another possibility is using pattern recognition models [33] to identify similar patterns (or neighbours) in the historical data using previous observations to missing values and to fill the gaps them with those values. For example Sternickel [34] characterized the time series using wavelet transformation methods, and trained a neural network with those features in order to detect similar patterns in the complete time series data. In addition, clustering techniques [35] are capable of grouping similar signals that can be later be used for imputing missing data. These techniques use algorithms such as dynamic time warping (DTW) [36] that provide a measure of similarity between time series. Finally,

lazy learning techniques [37] are capable of adapting the number of neighbours needed depending on the results obtained by cross-validation methods.

3 Proposed approach

This paper proposes a method for adaptively identifying an optimal data imputation method for a specific time series on a window-by-window basis. Thus, this contribution focuses on quantifying the effect of the application of different imputation algorithms to a time series depending on the characterization of the observed data and gap distribution characteristics. The approach consists in training a machine learning model that is able to predict from signal and gap characterization which imputation technique to use in each case. The proposed approach is validated in all the datasets available at UCR time series public data archive [38].

4 Experimental setup

The schema showing the steps in the experimental setup is shown in Figure 1. Each step is detailed in the following subsections.

The section is structured as follows. Subsection 4.1 describes the dataset used and how missing observations are generated. Then, characterization features used in the experiments are detailed in Subsection 4.2. Subsection 4.3 provides a description of the selected imputation methods, whereas Subsection 4.4 shows the measure used for validating the imputation. Finally, the model generation is described in Subsection 4.5.

4.1 Dataset and missing values generation

The considered datasets are the ones available at the UCR Time Series Classification Archive [38]. The Archive contains 128 classification time series datasets of different types including sensor data, simulated data, motion data from several devices and health data such as electrocardiograph (ECG), electrooculography (EOG) and hemodynamic data. First, as UCR archive datasets are complete (without missing values) multiple observations should be deleted, in other words amputed, in order to later impute. The training set in each dataset is amputed, whereas the test set is used to generate descriptors of the dataset and as historical complete data for imputation methods that need it.

There are three possible types of missing data [39, 40]: missing completely at random (MCAR), missing at random (MAR) and not missing at random (NMAR). MCAR describes the situation when missing data occurs entirely at random, all data points in the time series have the same probability of becoming missing data. In other words, the missing observation cannot depend on the value of the data point itself, neither on the value of another variable. In the MAR type the missing probability for an observation is independent of the value of the observation itself, but it depends on the value of other variables. In the time series univariate case the values of the time can be used

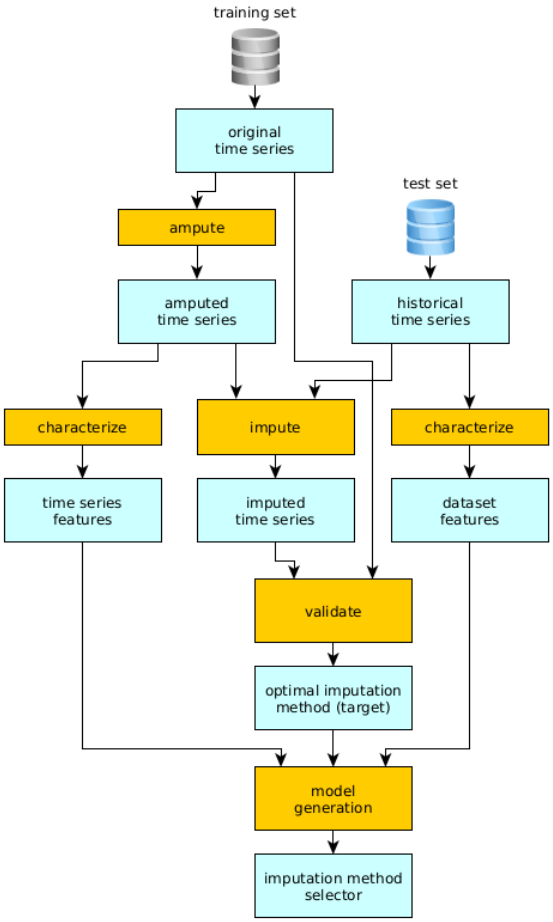


Fig. 1 Schema showing the experiment setup.

as other variable. Finally, in the NMAR case, the probability of a observation being missing depends on the value of the observation itself.

For simulating MCAR type missing values, time series were amputed using a uniform distribution in time index. For MAR case Poisson distribution was used to decide which observations were removed from the time series. $\lambda = len(x)/2$, being x the time series, was selected for for defining the distribution and to be adaptable with different length of the time series in the dataset. Finally, for replicating MNAR case and with the idea that the probability of some observation being removed depends on the value of observation itself, observation values are sorted and higher removing probability is given to low, central or high values depending on the option selected.

The amputation is done separately for each time series contained in each dataset. The amputation technique and the percentage are selected randomly between MAR and MNAR methods and missing percentage values (20% or 50%). Initially, MCAR amputation method was included in the experiment but

later, it was discarded as the majority of the gaps generated by this method were small (gap length less than 3); due to their limited length, there were minimal differences between imputed observations using different techniques.

4.2 Time series, dataset and gap characterization

Firstly, a gap is defined as continuous rank of missing observations. For each gap their position in time in the considered time window, length, and known previous and following observations of the actual gap being characterized are used as descriptors.

Let x_i be the observation value of the time series x in time i. Suppose $x'_i = x_{i+1} - x_i$ is the first derivative, $x''_i = x'_{i+1} - x'_i$ the second derivative, μ the mean and σ the standard deviation of observations values. Then, the features for characterizing the time series are defined next:

Coefficient of variation (CV)

$$CV = \frac{\sigma_x}{\mu_x} \tag{1}$$

Root mean square (RMS)

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \tag{2}$$

Mean absolute change (MAC)

$$MAC = \frac{1}{n-1}\sum_{i=1}^{n-1} \mathsf{x'\_i} \tag{3}$$

Mean distance (MD)

$$MD = \frac{1}{n-1}\sum_{i=1}^{n-1}\left(\sqrt{x'^2_i + 1}\right) \tag{4}$$

Shape factor (SF)

$$SF = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}}{\frac{1}{n}\sum_{i=1}^{n} abs(x_i)} \tag{5}$$

Mobility (M)

$$M = \frac{\sigma_{x'}}{\sigma_x} \tag{6}$$

Complexity (C)

$$C = \frac{M_{x'}}{M_x} = \frac{\sigma_{x''}/\sigma_{x'}}{\sigma_{x'}/\sigma_x} \tag{7}$$

The slope of the linear regression of the known observations and the R^2 indicating the linearity of the observations are also calculated in the time series characterization. As a predictability measure of the time series, autocorrelation is another feature considered.

Continuing with dataset descriptors, statistical distribution descriptors are used as features. Concretely the mean, median, standard value, skewness, kurtosis and maximum and minimum values. Gap characterization features is used to select the imputation method together with time series characterization.

4.3 Data imputation

This research focuses on single imputation due to simplicity needed for streaming or IOT applications. The list of selected imputation techniques is the following: Akima and Krige interpolation methods and k-nearest method using for imputing the observation the mean value in the same position of 3 curves that are the most similar using DTW score as distance measure.

On one hand, Akima should be the adequate imputation method for small gaps and smooth time series but depends totally on the previous and posterior gap nearby observations. On the other hand, Krige takes into account gaussian processes not centering all the attention of the interpolation technique in the gap nearby observations. Finally, k-nearest method allows looking patters that had occurred previously enabling considering different imputation shapes in bigger gaps.

4.4 Imputation validation measure

The optimal imputation between selected imputation methods is validated by the method obtaining the minimal error indicated by Percentage root mean square difference (PRD) measure in in the amputed observations. Let x_i be the observation value of the time series x in time i and $\hat{x}_i$ the observation value of the imputed time series in the same time i, the the PRD measure is defined by the following equation:

$$PRD(\hat{x}, x) = \left(\frac{\sum (x_i - \hat{x}_i)^2}{\sum (x_i)^2} \right)^{1/2} \tag{8}$$

4.5 Model generation

The training dataset used for model generation is created by the characterization of first 20 amputed time series of the training set of each dataset as described in Subsections 4.1 and 4.2. Then, each time series is imputed with all available imputation methods (Krige, k-nearest and Akima) and error value defined in the subsection 4.4 are calculated comparing imputed observations of each time series with original time series values from the dataset.

The optimal imputation method for each time series in considered the one with the lowest error value. The feature selection is done by point biserial correlation coefficient [41] between the features obtained in the characterization

and optimal imputation method obtained for each time series. A combination of two SVC models are trained in order to predict which imputation method is the most adequate regarding time series known values characterization and gaps distribution. The first model is trained to separate Krige and k-nearest imputation methods vs Akima interpolation and the second trained to separate the methods Krige and k-nearest.

5 Experimental results

The results shown in this section are generated using the PRD error method to select the optimal imputation method in each time series. The number of time series for each imputation methods is shown in Table 1. The most correlated features obtained in this specific case are the largest length of gaps in each time series and the mobility value of the amputed time series. The largest length of gaps feature is changed to logarithmic scale to separate the imputation methods more significantly. The two models are generated for predicting the optimal imputation method for all the datasets contained in the UCR time series archive. With those two features, the first model reached accuracy is 78% and the second model with accuracy of 67%. The combination of both models reaches an accuracy of 65% and the confusion matrix is shown in Figure 2. Finally features selected for model generation and separated by each imputation and coloured depending on the prediction of each method are shown in Figure 3. In general terms, the classification model recommends Akima for small gaps imputation and k-nearest for bigger gaps and Krige when the mobility reaches high values.

Table 1 Model training dataset distribution by optimal imputation method for PRD error

Imputation method	Optimal method times count
Akima	1304
k-nearest	649
Krige	588

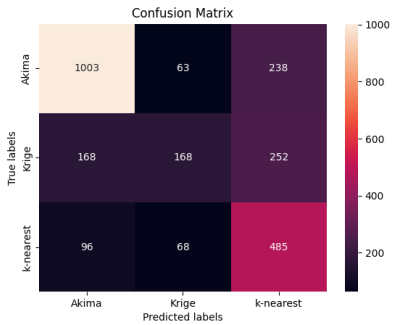


Fig. 2 Confusion matrix of the generated model

Fig. 3 Selected features log_length_holes_max vs mobility

In the Tables 2 and 3 the mean PRD imputation value aggregated by dataset is shown. The datasets are ordered in ascending order from the minimum mean error value obtained using the optimal imputation method for each time series (Optimal column). The Akima, k-nearest and Krige columns show the mean error value when a fixed imputation method is used for imputing the dataset and finally the Predicted column shows the quality of the imputation with the combination of models' predicted imputation method is used in each time series. The Optimal column is used as reference and from the rest of the columns the method obtaining the minimum mean error value is marked in bold.

A final total count indicating how many times each of the studied methods are optimal for a dataset is provided at the end of Tables 2 and 3 as summary. In both tables proposed model's predictions is optimal most of the times. As the mean error increases using Akima as unique interpolation technique appears to be adequate and in the last half Krige method appears considerable times. Rarely k-nearest imputation method is selected as optimal imputation technique for time series and this maybe due to the fact that the amputed time

series is pre-imputed by linear interpolation before calculating DTW distance with time series of the test set. This pre-imputation may devalue the distance score between time series.

Table 2 Mean PRD error values aggregated by each dataset from the UCR time series classification archive

Dataset	Optimal	Akima	k-nearest	Krige	Predicted
InsectEPGRegularTrain	0.005	**0.007**	0.016	0.008	0.013
InsectEPGSmallTrain	0.026	0.035	0.063	0.034	**0.032**
GestureMidAirD2	0.032	**0.036**	0.181	0.067	0.073
GunPointMaleVersusFemale	0.033	0.056	0.101	0.068	**0.043**
GunPointOldVersusYoung	0.039	0.058	0.099	0.099	**0.053**
PigArtPressure	0.039	0.082	0.132	0.139	**0.045**
GunPointAgeSpan	0.042	0.057	0.125	0.08	**0.043**
Meat	0.042	0.227	0.059	0.209	**0.046**
GunPoint	0.057	0.084	0.178	0.12	**0.057**
OliveOil	0.071	0.677	0.101	0.627	**0.073**
Rock	0.073	0.102	0.265	**0.095**	0.097
Wine	0.089	0.555	**0.111**	0.608	0.12
FreezerRegularTrain	0.094	0.18	**0.106**	0.62	0.107
FreezerSmallTrain	0.096	0.243	**0.112**	0.775	0.115
ProximalPhalanxTW	0.099	0.292	0.185	0.278	**0.107**
DistalPhalanxTW	0.103	0.296	0.187	0.28	**0.109**
EthanolLevel	0.105	0.152	0.165	0.283	**0.105**
DiatomSizeReduction	0.109	0.529	0.143	0.344	**0.122**
StarLightCurves	0.114	0.234	0.265	0.276	**0.142**
Crop	0.12	**0.141**	0.203	0.21	0.188
ProximalPhalanxOutlineCorrect	0.12	0.561	0.202	0.446	**0.123**
PickupGestureWiimoteZ	0.121	0.134	0.376	0.182	**0.125**
ProximalPhalanxOutlineAgeGroup	0.127	0.649	0.206	0.516	**0.129**
Trace	0.129	**0.164**	0.382	0.248	0.242
Strawberry	0.131	0.339	**0.151**	0.432	0.171
Coffee	0.136	0.54	0.157	0.538	**0.142**
GestureMidAirD3	0.153	**0.17**	0.613	0.219	0.272
AllGestureWiimoteZ	0.156	0.186	0.34	0.194	**0.169**
TwoLeadECG	0.157	0.322	0.277	0.41	**0.16**
EOGHorizontalSignal	0.162	**0.164**	0.665	0.2	0.203
GestureMidAirD1	0.165	**0.167**	0.975	0.382	0.415
Chinatown	0.171	**0.186**	0.256	0.223	**0.186**
DistalPhalanxOutlineCorrect	0.172	0.462	0.292	0.471	**0.199**
NonInvasiveFetalECGThorax1	0.175	0.44	0.257	0.721	**0.196**
MiddlePhalanxOutlineAgeGroup	0.176	0.764	0.231	0.569	**0.194**
NonInvasiveFetalECGThorax2	0.191	0.425	0.24	0.724	**0.201**
DistalPhalanxOutlineAgeGroup	0.191	0.95	0.284	0.594	**0.201**
MiddlePhalanxTW	0.192	0.829	0.231	0.64	**0.2**
MiddlePhalanxOutlineCorrect	0.193	1.123	0.232	0.658	**0.193**
MelbournePedestrian	0.198	**0.216**	0.275	0.396	0.228
PLAID	0.2	0.408	0.49	**0.239**	0.361
InlineSkate	0.201	0.248	0.414	0.313	**0.243**
UWaveGestureLibraryY	0.201	**0.272**	0.515	0.385	0.294
Adiac	0.205	0.73	0.266	0.459	**0.217**
Symbols	0.208	0.375	0.351	0.479	**0.276**
PigCVP	0.208	0.892	0.364	0.282	**0.225**
Beef	0.214	0.554	0.292	0.546	**0.247**
PhalangesOutlinesCorrect	0.222	0.648	0.3	0.577	**0.253**
UMD	0.226	**0.336**	0.465	0.37	0.337
Herring	0.227	0.609	0.294	0.413	**0.244**
UWaveGestureLibraryZ	0.229	0.437	0.498	**0.374**	0.379
Mallat	0.238	0.644	0.276	0.708	**0.238**
Fish	0.238	0.721	0.337	0.38	**0.263**
BME	0.242	**0.286**	0.512	0.33	0.425
EOGVerticalSignal	0.244	**0.306**	0.833	0.328	0.516
SwedishLeaf	0.25	0.477	0.442	0.303	**0.267**
DodgerLoopWeekend	0.255	0.755	0.287	**0.278**	0.291
AllGestureWiimoteY	0.257	**0.269**	0.905	0.345	0.462
DodgerLoopGame	0.26	0.428	0.299	**0.294**	0.297
ShapesAll	0.26	**0.284**	0.817	0.372	0.561
Car	0.262	0.628	0.397	0.405	**0.309**
DodgerLoopDay	0.266	0.595	0.293	0.334	**0.287**
ECG5000	0.273	0.371	0.325	0.449	**0.301**
BirdChicken	0.275	**0.287**	0.809	0.488	0.555
Best approach total count	-	14	4	5	**40**

Table 3 Mean PRD error values aggregated by each dataset from the UCR time series classification archive

Dataset	Optimal	Akima	k-nearest	Krige	Predicted
MedicalImages	0.276	0.346	0.54	0.458	**0.344**
MixedShapesRegularTrain	0.282	0.545	0.785	**0.424**	0.456
FiftyWords	0.295	0.568	0.821	0.502	**0.402**
Wafer	0.298	0.425	0.414	0.622	**0.358**
ArrowHead	0.299	0.592	0.444	0.426	**0.362**
WordSynonyms	0.302	**0.392**	0.997	0.489	0.419
UWaveGestureLibraryX	0.306	0.537	0.655	**0.369**	0.498
Yoga	0.307	**0.413**	0.648	0.45	0.546
Haptics	0.323	**0.42**	0.576	0.694	0.422
Ham	0.339	0.997	0.447	0.765	**0.352**
ECG200	0.35	0.57	0.53	0.549	**0.378**
PigAirwayPressure	0.35	0.452	0.504	0.534	**0.45**
HandOutlines	0.361	0.791	0.451	0.818	**0.393**
UWaveGestureLibraryAll	0.371	0.89	0.744	0.57	**0.461**
MoteStrain	0.378	0.56	0.571	0.561	**0.478**
ItalyPowerDemand	0.385	0.461	0.662	0.481	**0.46**
AllGestureWiimoteX	0.396	0.521	0.832	0.494	**0.485**
MixedShapesSmallTrain	0.416	0.789	0.825	**0.594**	0.617
PowerCons	0.418	0.479	0.689	0.476	**0.457**
Worms	0.421	0.605	0.844	0.569	**0.494**
BeetleFly	0.423	0.822	1.037	0.635	**0.532**
ShakeGestureWiimoteZ	0.44	0.926	0.735	**0.55**	0.644
ECGFiveDays	0.449	0.651	0.717	0.845	**0.57**
HouseTwenty	0.451	**0.474**	0.751	0.542	0.564
SmoothSubspace	0.457	0.62	0.496	**0.487**	0.495
OSULeaf	0.461	0.674	0.975	**0.598**	0.619
Plane	0.465	0.771	0.641	0.704	**0.597**
SemgHandSubjectCh2	0.474	2.705	0.553	**0.475**	**0.475**
SonyAIBORobotSurface1	0.474	0.673	0.616	0.755	**0.543**
InsectWingbeatSound	0.489	1.092	1.098	**0.652**	0.803
Fungi	0.49	**0.545**	0.741	0.688	0.618
GesturePebbleZ1	0.491	**0.547**	0.945	0.605	0.591
SemgHandMovementCh2	0.494	1.526	0.62	**0.496**	**0.496**
WormsTwoClass	0.5	1.087	0.905	0.624	**0.599**
CricketZ	0.52	0.756	0.808	0.709	**0.608**
SemgHandGenderCh2	0.527	1.955	0.609	**0.533**	0.535
FordA	0.55	1.826	1.202	1.063	**0.641**
ToeSegmentation1	0.553	1.031	0.913	0.754	**0.617**
GesturePebbleZ2	0.56	0.87	1.013	0.743	**0.61**
LargeKitchenAppliances	0.562	2.053	1.818	**0.859**	1.6
SonyAIBORobotSurface2	0.563	0.787	0.779	0.869	**0.748**
ToeSegmentation2	0.578	0.813	1.048	**0.66**	0.743
CBF	0.596	0.909	0.736	0.684	**0.668**
Lightning7	0.598	0.819	0.825	0.657	**0.627**
FaceFour	0.604	0.877	0.785	0.934	**0.655**
CinCECGTorso	0.606	0.787	1.122	**0.657**	0.784
Computers	0.612	**0.702**	1.256	0.72	1.063
FacesUCR	0.619	1.063	0.904	0.975	**0.716**
CricketY	0.637	0.781	0.913	0.82	**0.685**
ScreenType	0.642	0.812	0.986	**0.734**	0.795
CricketX	0.649	0.81	0.84	0.793	**0.699**
Phoneme	0.655	2.406	1.111	0.944	**0.737**
ChlorineConcentration	0.687	0.916	**0.718**	0.849	0.766
RefrigerationDevices	0.707	0.972	1.154	0.974	**0.849**
Lightning2	0.734	1.399	1.049	**0.814**	0.861
TwoPatterns	0.738	0.894	1.124	**0.858**	0.895
FaceAll	0.759	1.355	0.998	1.017	**0.908**
FordB	0.773	2.738	1.176	1.117	**0.904**
ElectricDevices	0.818	**1.432**	2.5	1.489	1.52
SmallKitchenAppliances	0.938	**0.956**	1.443	1.038	1.043
Earthquakes	1.015	1.142	1.125	**1.015**	**1.015**
ACSF1	1.021	15.607	1.186	**1.032**	**1.032**
ShapeletSim	1.073	4.58	1.147	**1.086**	**1.086**
SyntheticControl	1.074	1.38	1.153	**1.145**	1.149
Best approach total count	-	9	1	20	**39**

6 Conclusions and future work

This paper has introduced a novel method called ASSIST aimed at adaptively identifying optimal imputation strategies for time series data, on a window-by-window basis. A method to characterize time series and gap distributions has

been presented, together with a procedure to train a model in order to predict a best-fit imputation strategy. This approach has been validated in the entire UCR time series public data archive.

In the actual experiment only 20 time series of each dataset are amputed, characterized and used for generating the model. The features obtained from in each time series of all the datasets in the UCR Archive are combined in an unique dataset for its later use as training data for model generation. From the defined characterization features two of them reach biserial point correlation values greater than 0.25 for selected imputation methods. Finally, the selected model has been trained with default parameter values and limited and imbalanced data for classification strategies. Even though the previously mentioned details, the proposed methodology is able to select best fitting imputation method from the available ones for each time series. This experiment shows that the gap characterization and actual observations values are valuable for selecting adequate imputation strategies.

Future work to be considered includes characterizing all the time series of training sets of the URC Archive datasets obtaining enough data to train more complex classification models or even considering separating different time series datasets depending on their characteristics. In this way, more accurate models could be generated allowing imputation method prediction with a controlled risk assessment technique. Finally, the imputed observations can be validated previous to final imputation method selection by the use of the characteristics of the datasets that are not currently used in none of the steps of the proposed methodology.

Declarations

Funding This research has been partially funded by the 3KIA project (ELKARTEK, Basque Government).

Competing interests The authors declare no competing interests.

Availability of data and material Not applicable.

Code availability Not applicable

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Author's contributions A.G. and M.Q. designed and implemented the experimental testbed and algorithm, B.S., and I.G. supervised the experimental design and managed the project. M.Q. and B.S. reviewed the new approach of this research. A.G. performed the experimental phase. All authors contributed to the writing and reviewing of the present manuscript. All authors read and agreed to the published version of the manuscript.

References

[1] Peng, T., Sellami, S., Boucelma, O.: Iot data imputation with incremental multiple linear regression. Open Journal of Internet Of Things (OJIOT)

5(1), 69–79 (2019)

[2] Liu, Y., Dillon, T., Yu, W., Rahayu, W., Mostafa, F.: Missing value imputation for industrial iot sensor data with large gaps. IEEE Internet of Things Journal **7**(8), 6855–6867 (2020). https://doi.org/10.1109/JIOT.2020.2970467

[3] Ding, Z., Mei, G., Cuomo, S., Li, Y., Xu, N.: Comparison of estimating missing values in iot time series data using different interpolation algorithms. International Journal of Parallel Programming **48**(3), 534–548 (2020)

[4] Gorelick, M.H.: Bias arising from missing data in predictive models. Journal of clinical epidemiology **59**(10), 1115–1123 (2006)

[5] Pombo, N., Rebelo, P., Araújo, P., Viana, J.: Design and evaluation of a decision support system for pain management based on data imputation and statistical models. Measurement **93**, 480–489 (2016). https://doi.org/10.1016/j.measurement.2016.07.009

[6] Oehmcke, S., Zielinski, O., Kramer, O.: knn ensembles with penalized dtw for multivariate time series imputation. In: 2016 International Joint Conference on Neural Networks (IJCNN), pp. 2774–2781 (2016). IEEE

[7] Xu, X., Liu, X., Kang, Y., Xu, X., Wang, J., Sun, Y., Chen, Q., Jia, X., Ma, X., Meng, X., *et al.*: A multi-directional approach for missing value estimation in multivariate time series clinical data. Journal of Healthcare Informatics Research **4**, 365–382 (2020)

[8] Bertsimas, D., Orfanoudaki, A., Pawlowski, C.: Imputation of clinical covariates in time series. Machine Learning **110**(1), 185–248 (2021)

[9] Wilson, R.E., Eckley, I.A., Nunes, M.A., Park, T.: A wavelet-based approach for imputation in nonstationary multivariate time series. Statistics and Computing **31**(2), 1–18 (2021)

[10] Li, Y., Bao, T., Chen, H., Zhang, K., Shu, X., Chen, Z., Hu, Y.: A large-scale sensor missing data imputation framework for dams using deep learning and transfer learning strategy. Measurement **178**, 109377 (2021). https://doi.org/10.1016/j.measurement.2021.109377

[11] Dong, Y., Peng, C.-Y.J.: Principled missing data methods for researchers. SpringerPlus **2**(1), 1–17 (2013)

[12] Schouten, R.M., Lugtig, P., Vink, G.: Generating missing values for simulation purposes: a multivariate amputation procedure. Journal of Statistical Computation and Simulation **88**(15), 2909–2930 (2018)

[13] Khayati, M., Lerner, A., Tymchenko, Z., Cudré-Mauroux, P.: Mind the gap: an experimental evaluation of imputation of missing values techniques in time series. Proceedings of the VLDB Endowment **13**(5), 768–782 (2020)

[14] Zhang, S.: Shell-neighbor method and its application in missing data imputation. Applied Intelligence **35**(1), 123–133 (2011)

[15] Somasundaram, R., Nedunchezhian, R.: Evaluation of three simple imputation methods for enhancing preprocessing of data with missing values. International Journal of Computer Applications **21**(10), 14–19 (2011)

[16] Rubin, D.B.: Multiple imputation after 18+ years. Journal of the American statistical Association **91**(434), 473–489 (1996)

[17] Yang, S., Kim, J.K.: Fractional imputation in survey sampling: A comparative review. Statistical Science **31**(3), 415–432 (2016)

[18] Liu, J., Gelman, A., Hill, J., Su, Y.-S., Kropko, J.: On the stationary distribution of iterative imputations. Biometrika **101**(1), 155–173 (2014)

[19] Steffensen, J.F.: Interpolation. Courier Corporation, ??? (2006)

[20] McKinley, S., Levine, M.: Cubic spline interpolation. College of the Redwoods **45**(1), 1049–1060 (1998)

[21] Akima, H.: A new method of interpolation and smooth curve fitting based on local procedures. Journal of the ACM (JACM) **17**(4), 589–602 (1970)

[22] Fritsch, F.N., Carlson, R.E.: Monotone piecewise cubic interpolation. SIAM Journal on Numerical Analysis **17**(2), 238–246 (1980)

[23] Oliver, M.A., Webster, R.: Kriging: a method of interpolation for geographical information systems. International Journal of Geographical Information System **4**(3), 313–332 (1990)

[24] Gardner Jr, E.S.: Exponential smoothing: The state of the art. Journal of forecasting **4**(1), 1–28 (1985)

[25] Chatfield, C., Yar, M.: Holt-winters forecasting: some practical issues. Journal of the Royal Statistical Society: Series D (The Statistician) **37**(2), 129–140 (1988)

[26] Shibata, R.: Selection of the order of an autoregressive model by akaike's information criterion. Biometrika **63**(1), 117–126 (1976)

[27] Ma, R., Boubrahimi, S.F., Hamdi, S.M., Angryk, R.A.: Solar flare prediction using multivariate time series decision trees. In: 2017 IEEE

International Conference on Big Data (Big Data), pp. 2569–2578 (2017). IEEE

[28] Yoo, T.-W., Oh, I.-S.: Time series forecasting of agricultural products' sales volumes based on seasonal long short-term memory. Applied Sciences **10**(22) (2020). https://doi.org/10.3390/app10228169

[29] Cao, L.-J., Tay, F.E.H.: Support vector machine with adaptive parameters in financial time series forecasting. IEEE Transactions on neural networks **14**(6), 1506–1518 (2003)

[30] Bontempi, G., Taieb, S.B., Le Borgne, Y.-A.: Machine learning strategies for time series forecasting. In: European Business Intelligence Summer School, pp. 62–77 (2012). Springer

[31] de O. Santos Júnior, D.S., de Oliveira, J.F.L., de Mattos Neto, P.S.G.: An intelligent hybridization of arima with machine learning models for time series forecasting. Knowledge-Based Systems **175**, 72–86 (2019). https://doi.org/10.1016/j.knosys.2019.03.011

[32] Maghrour Zefreh, M., Torok, A.: Single loop detector data validation and imputation of missing data. Measurement **116**, 193–198 (2018). https://doi.org/10.1016/j.measurement.2017.10.066

[33] Xing, Z., Pei, J., Philip, S.Y.: Early prediction on time series: A nearest neighbor approach. In: Twenty-First International Joint Conference on Artificial Intelligence (2009)

[34] Sternickel, K.: Automatic pattern recognition in ecg time series. Computer methods and programs in biomedicine **68**(2), 109–115 (2002)

[35] Aghabozorgi, S., Shirkhorshidi, A.S., Wah, T.Y.: Time-series clustering–a decade review. Information Systems **53**, 16–38 (2015)

[36] Müller, M.: Dynamic time warping. Information retrieval for music and motion, 69–84 (2007)

[37] Zhang, M.-L., Zhou, Z.-H.: Ml-knn: A lazy learning approach to multi-label learning. Pattern recognition **40**(7), 2038–2048 (2007)

[38] Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.-C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The UCR time series archive. IEEE/CAA Journal of Automatica Sinica **6**(6), 1293–1305 (2019)

[39] Rubin, D.B.: Inference and missing data. Biometrika **63**(3), 581–592 (1976)

[40] Little, R.J., Rubin, D.B.: Statistical Analysis with Missing Data vol. 793.

John Wiley & Sons, ??? (2019)

[41] Lev, J., *et al.*: The point biserial coefficient of correlation. Annals of Mathematical Statistics **20**(1), 125–126 (1949)

energies

MDPI

Article

Ensemble Surrogate Models for Fast LIB Performance Predictions

Marco Quartulli [1,*], Amaia Gil [1], Ane Miren Florez-Tapia [1], Pablo Cereijo [2], Elixabete Ayerbe [2] and Igor G. Olaizola [1]

[1] Vicomtech, Basque Research and Technology Alliance, Mikeletegi 57, 20009 Donostia-San Sebastián (ES), Spain; agil@vicomtech.org (A.G.); amflorez@vicomtech.org (A.M.F.-T.); iolaizola@vicomtech.org (I.G.O.)
[2] CIDETEC, Basque Research and Technology Alliance, Pº Miramón 196, 20014 Donostia-San Sebastián (ES), Spain; pcereijo@cidetec.es (P.C.); eayerbe@cidetec.es (E.A.)
* Correspondence: mquartulli@vicomtech.org; Tel.: +34-943-309230

Abstract: Battery Cell design and control have been widely explored through modeling and simulation. On the one hand, Doyle's pseudo-two-dimensional (P2D) model and Single Particle Models are among the most popular electrochemical models capable of predicting battery performance and therefore guiding cell characterization. On the other hand, empirical models obtained, for example, by Machine Learning (ML) methods represent a simpler and computationally more efficient complement to electrochemical models and have been widely used for Battery Management System (BMS) control purposes. This article proposes ML-based ensemble models to be used for the estimation of the performance of an LIB cell across a wide range of input material characteristics and parameters and evaluates 1. Deep Learning ensembles for simulation convergence classification and 2. structured regressors for battery energy and power predictions. The results represent an improvement on state-of-the-art LIB surrogate models and indicate that deep ensembles represent a promising direction for battery modeling and design.

Keywords: Li-ion battery; surrogate modeling; deep learning ensembles

Citation: Quartulli, M.; Gil, A.; Florez-Tapia, A.M.; Cereijo, P.; Ayerbe, E.; Olaizola, I.G. Ensemble Surrogate Models for Fast LIB Performance Predictions. *Energies* **2021**, *14*, 4115. https://doi.org/10.3390/en14144115

Academic Editor: Marcin Kamiński

Received: 12 June 2021
Accepted: 6 July 2021
Published: 8 July 2021

1. Introduction

This paper considers the task of estimating the energy and power density of LIB designs across a range of characteristics and parameters and focuses on the problem of obtaining these results without incurring the considerable computational costs of detailed physical simulations in a recurrent manner. While state-of-the-art contributions such as [1] exploit Deep Learning neural networks trained on simulated data as surrogate models for this objective, to the best of our knowledge, no literature contribution yet has evaluated the performance of so-called ensemble models for this task. We address this gap in the literature by comparing state-of-the-art models with a number of ensemble surrogates.

Even though Lithium-Ion Batteries (LIB) have progressively been improved since their market introduction in 1991 by Sony, their massive deployment requires them to be further optimized in terms of performance, durability, and safety. From the physical point of view, to obtain these results, in the battery design phase, it is important to reduce limitations to ion transport, thus avoiding undesirable cell polarization effects. Consequently, an extensive literature describes the effect each design parameter has on transport mechanisms, and therefore on cell performance. On the one hand, the effects of several parameters (including electrode thickness, particle size, tortuosity and discharge rate) have been investigated by experimental methods. On the other hand, cell design has also been explored through modeling approaches, demonstrating that model-based design can be used to reduce the number of experiments, while accurately describing battery performance and providing guidance for battery characterization.

In general, the battery models presented in the literature mainly fall into two categories: physics-based electrochemical models and empirical ones. In addition, the recent

development of physics-based equivalent circuit models aims to combine the high accuracy of physical models with the reduced computational cost of empirical ones.

Physics-based electrochemical models [2–4] use partial differential equations to describe the phenomena taking place within the battery. They can be used to forecast its electrochemical state, and to provide accurate information about variables such as lithium concentrations and over-potentials, which can be used to understand the phenomena that are limiting performance or durability. For example, the widely used Pseudo-two-Dimensional (P2D) electrochemical model developed by Doyle and Newman [5,6] is based on porous electrode theory, concentrated solution theory, and kinetics equations. Further electrochemical modeling approaches rely on Single Particle (SP) simplifications, in which the properties of the electrolyte are normally ignored [7–9]. Further recent approaches such as SPMe [10] consider electrolyte dynamics in SP models. Other possibilities include Simplified Models building on polynomial profiles [11,12], Galerkin approximations [13], transfer function modeling [14], and the like. Concerning limitations, a drawback of detailed electrochemical models is the significant computational cost of simulations [15]. To overcome this limitation, various reduced-order models have been developed, which are either distributed-parameter models [10,13] or lumped-parameters models [14,16]. Distributed-parameter models are normally expressed in the form of ordinary differential-algebraic equations (DAEs), derived, for example, using the Galerkin projection methods or Proper Orthogonal Decomposition, trying to preserve the physical meaning of all model parameters. Lumped-parameter models, which mimic the output voltage of the battery using electrical components such as resistors and capacitors, are described below.

Empirical models are typically either based on equivalent circuits [16,17] or on data-driven approaches [18,19]. battery control algorithm rather than in cell design optimization applications. They are composed of an open-circuit voltage source connected to a set of electric elements, such as resistors and capacitors, to model the electrical behavior of a battery. While these models are intuitive and relatively simple to use in control system design and implementation, they do not provide insights on the internal behavior of the battery. In this regard, new approaches have recently been explored, such as the developed distributed-parameter ECMs [20,21], in which the models are normally expressed in the form of DAEs, which can be solved rapidly using the proposed method with high accuracy. This represents an improvement on existing physics-based Li-ion battery models, especially in real-time, dynamic environments. Physics-based equivalent circuit models combine the benefits of high accuracy physical models with the lower computational cost of empirical ones, for instance, by combining a concise transmission line structure with partial differential equations for the mass transport processes that describe the concentration distributions and that are solved with the finite difference method, avoiding simplifications or approximations, thus guaranteeing the accuracy of the results [20]. Online estimation and prediction of the Remaining Useful Life also often use data-driven empirical methods which, however, have not been commonly exploited for cell design purposes.

So-called 'surrogate' models are obtained by Machine Learning methods from data generated by simulations. Once the training phase is completed, these models can provide estimations of battery performance indicators with a lower computational cost and with an accuracy similar to that of physical simulators. Probabilistic surrogate models are often used for optimization: by running the simulations at a set of points (experimental design), one obtains fast surrogates for otherwise expensive objective functions [22]. In methods such as Upper and Lower Confidence Bound [23], Expected Improvement [24], DYCORS [25], and SOP [26], the optimization jointly performs both exploration and exploitation, looking for an optimum, while, at the same time, sampling by simulation the most uncertain parameter regions. In this sense, surrogate optimization schemes can efficiently leverage the ability of probabilistic surrogate model classes, from Gaussian Processes to Sequential Radial Basis Functions, to jointly generate estimates for both the local value and the local uncertainty of an objective function of interest. Relevant efforts for LIB aging modeling include those based on Gaussian processes by Liu et al. [27–29].

Contributing to the domain of such surrogates, Wu et al. [1] propose to address cell design characterization by combining Machine Learning classifiers and regressors based on deep feed-forward neural network models. While the first neural network is a classifier that predicts whether a set of input design variables would result in a physically realizable cell, the second neural network estimates the specific energy and specific power of the design. Both neural networks are trained and validated using data from finite-element, thermo-electrochemical simulations.

Although ensemble models are well represented in the literature, to the best of our knowledge, no contribution yet has evaluated their performance as surrogates of physical LIB simulators. This paper addresses this gap by comparing a state-of-the-art LIB surrogate model based on deep feed-forward networks with a set of ensemble models integrating deep classifiers and regressors. In this sense, we demonstrate the accuracy of composite surrogate models for the estimation of the density of power and energy of a given LIB parameter set. The input and output parameters taken into consideration for the electrochemical simulations are listed in Table 1. Their values are measured by proprietary electrochemical and physicochemical protocols for a set of proprietary electrodes in CIDETEC, and vary between the minimum and maximum values shown in Table 1.

Table 1. Input and output parameters for both the simulators and the proposed surrogate models. The simulated dataset consists of 1000 records with 11 input and 3 output attributes.

Variable	Type		Units	Range
Anode/Cathode				
thickness	input	L_n, L_p	µm	45–75/60–90
porosity	input	ϵ_n, ϵ_p	-	0.2–0.3/0.18–0.28
particle radius	input	r_n, r_p	µm	3.5–8.5/8–11
Separator				
thickness	input	L_s	µm	15.0–30.0
porosity	input	ϵ	-	0.35–0.45
Electrolyte				
ionic conductivity	input	k_e	S/m	0.1–1.0
initial concentration	input	c_0	mol/m^3	1750–2250
Whole Cell				
applied C-rate	input	C_{rate}	1/h	0.33–3.0
convergence status	output	Boolean	-	N.A.
energy	output	E	Wh	N.A.
power	output	P	W	N.A.

In terms of application, the present paper does not directly focus on optimization. Instead, as also previously done in [1], it considers surrogates that can be used to estimate the performance of battery designs across a wide range of material characteristics and parameters. This applicative objective reduces the importance of using surrogate models that can output estimated values and their uncertainties, for instance by exploiting techniques such as Monte Carlo Dropout [30] and Variational Inference [31]. Consequently, in the rest of our treatment, we limit ourselves to non-probabilistic surrogates.

The methodology we introduce and detail in the sections below is based on the general framework put forward by [32,33], integrating and extending the Pseudo-2D surrogate proposed by [1] to evaluate the performance of composite surrogate models integrating both classification and regression. Through classification, a Machine Learning model evaluates the convergence of the simulator, while, through regression, the algorithms predict the output parameters of the Ragone plot [34] (energy and power, as per Figure 1). We extend the state of the art in [1] by showing the advantage of adopting respectively ensemble methods for convergence classification and structured regression for the estimation of the Ragone parameters, rather than simpler feed-forward networks for both tasks. Furthermore, from the methodological point of view, we apply a quantitative performance evaluation proce-

dure that is based on K-fold validation [35] rather than on simple train/test/validation splitting as in [1]. The accuracy of some of the ensemble surrogate models we introduce compares favorably to that of the state-of-the-art method introduced in [1]. Furthermore, the approach in the reference is extended by using a K-fold cross-validation method to evaluate if the model can generalize the quality of its results.

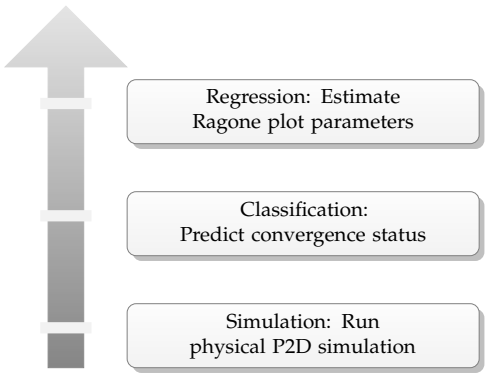


Figure 1. Global workflow. The methodology progresses from physical simulations to simulator convergence classification to the estimation by regression of a specific subset of Li-ion battery parameters of interest.

2. Methods

The construction and validation of the P2D-based surrogate model have involved several steps:

- Selection of the appropriate input and output parameters. In this regard, measurable electrode and electrolyte transport properties are selected as input parameters, while the parameters of the very-well known Ragone plot (energy and power) are selected as outputs to be used for battery characterization purposes, as shown in Table 1, where the minimum and maximum values for each of the parameters are also reported.
- Selection of electrochemical model and Design of Experiments. A proprietary P2D model is used to generate the data set.
- Development of surrogate models. Each surrogate is composed of three models: a binary simulation convergence classifier and two separate regression models that estimate integrated power and energy density values from the input parameters (see Figure 2).
- Model validation, typically carried out similarly as for the data-driven models.

2.1. Electrochemical Model and Design of Experiments

The electrochemical model used for building the data-set is expressed in terms of four conservation relations described in Smith et al. [36] by partial differential equations and their corresponding boundary conditions. The numerical approach considered for solving the system of equations is based on Finite Elements Methods (FEM) for space-discretization as implemented in the open-source FEniCS toolkit (https://fenicsproject.org/, accessed on 8 May 2021), and on Implicit Euler methods for time-discretization.

The data set is generated by running the model with the selected input variables and their ranges presented in Table 1.

As in Wu et al. [1], in this paper, we focus on the variables that are controllable during battery manufacturing. Among all the design variables, the electrode thickness, porosity, and particle radius are chosen because they can be controlled within the material or the electrode manufacturing steps. For example, electrode thickness and porosity can be experimentally adjusted during the calendering step of the cell manufacturing process. Separator thickness and porosity ranges are defined considering commercially available components. In addition, the initial electrolyte concentration and ionic conductivity are

further important tunable variables and are considered as they play an important role in determining thick electrode performance limitations due to the dry-out of the electrolyte. When it comes to the electrolyte, two parameters are selected for the design of experiments. First, the ionic conductivity (k_e) is selected as it can be modulated by doping salts, acids, metals, alkali, etc. to the solvent matrix and the initial salt concentration for a similar reason (i.e., c_0). For the sake of simplicity, we have assumed that the ionic conductivity has no dependence on the electrolyte concentration. In further developments, we envision that this dependency will be taken into account. Finally, the applied C-rate is also selected as an input variable, as this parameter may vary from very low values to very high ones depending on the battery application. Constant–current discharge protocols are used as a baseline, with the aim of the characterization of the cell in terms of energy and power during discharge. Even so, the testing protocol could be adapted to other battery applications. The rest of the parameters needed to simulate battery performance using a P2D model are summarized in Table 2 and Figure 3.

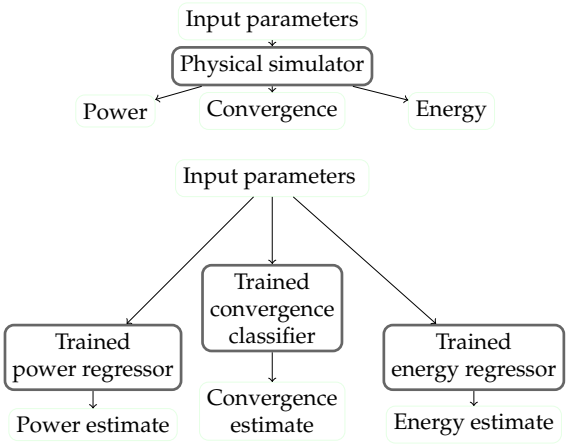
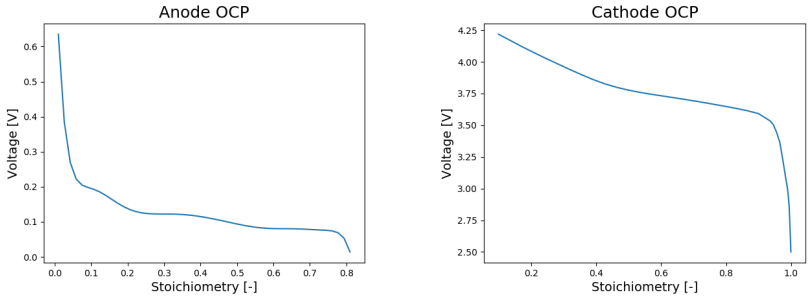


Figure 2. The general structure of the model. A simulator produces values for energy and power density integrals, together with a convergence flag indicating the successful execution of a run, up to a fixed end time. Run-stopping criteria corresponding to non-convergence involve, for instance, physically meaningless negative ion density values. A classifier and two regressors learned from the simulated data can be exploited to generate estimates. The classifier estimates whether a simulation will produce physically unrealizable conditions, and therefore stop without completing a run. The two regressors estimate integrated energy and power for converging runs.



(**a**) (**b**)

Figure 3. Open circuit potentials for the electrodes of the P2D model simulator. (**a**) Anode OCP curve: voltage response on each stoichiometry; (**b**) Cathode OCP curve: voltage response on each stoichiometry.

Table 2. Further parameters for the P2D model simulator.

Variable	Units	Value
Anode/Cathode		
active material volume fraction	%	56/70
effective electrical conductivity	S/m	3.0/2.0
maximum lithium concentration	mol/m^3	73,900.0/28,100.0
initial lithium concentration (100% SOC)	mol/m^3	59,859.0/3653.0
reference reaction rate coefficient	$m^{2.5}/mol^{0.5}$ s	$6.1 \cdot 10^{-9}/1.0 \cdot 10^{-10}$
reference solid diffusion coefficient	m^2/s	$1.0 \cdot 10^{-14}/1.0 \cdot 10^{-13}$
open circuit potential (OCP)	V	Figure 3a/Figure 3b
Electrolyte		
transference number	-	0.4
diffusion coefficient	m^2/s	$4.0 \cdot 10^{-6}$
Whole Cell & Constants		
Area	m^2	$2.16 \cdot 10^{-4}$
Universal gas constant	J/K mol	8.31
Faraday constant	C/mol	96,485.34

Considering the previous inputs, we generate a simulated data set by running a P2D model implemented in Python. As previously mentioned, simulations consider a given set of input parameter values. They propagate a set of variables describing the electrochemical status of the battery up to a well-defined and fixed time limit. Upon both convergence and premature termination of the simulation, Ragone energy and power estimates are integrated as per

$$\begin{aligned} E = & \quad \int_0^{t_d} IV \, dt \quad [Wh] \\ P = & \quad \int_0^{t_d} IV/t_d \, dt \quad [W] \, . \end{aligned}$$

where t_d is the discharge time. The Ragone plot [34] is commonly used to illustrate the performance of energy storage devices, and is widely used to compare technologies catering to specific demands. Accordingly, in this work, we generate a Ragone plot for a wide range of electrode/cell design parameters aimed at a variety of future designs of electrochemical energy storage devices for different usage applications. Constant currents for discharge are used as a baseline since the Ragone plot is usually generated with a constant C-rate. Accordingly, we do not consider variable currents.

Reasons for failures in simulating up to the specified end time include reaching physically meaningless or non-realizable situations, such as negative values for ion concentrations. In these cases, the run will stop prematurely, producing only partial estimates for the above Ragone integrals, and appropriately setting a negative convergence status flag.

2.2. Surrogate Modeling

We generate a surrogate model to reproduce the results of the P2D model simulator. Once trained, this empirical model can approximate simulation results in a fraction of the running time of the physical simulator, keeping the approximation errors under control. The input parameters taken into account correspond to the inputs of simulations. As previously mentioned, the output variables include the convergence status of the simulation, and the Ragone plot variables corresponding to energy and power integrals. The proposed surrogate model structure is a composite one: the convergence status is modeled by a binary classifier (Section 2.2.1) and the energy and power output integrals are approximated by regressors (Section 2.2.4).

2.2.1. Simulation Convergence Classification

We consider that, in a Pseudo-2D electrochemical model, the completion status of a simulation can either be full (corresponding to propagation up to the intended complete

temporal extension) or partial (corresponding to early termination). This status can depend on several phenomena with varying degrees of complexity. Those phenomena range from the consumption of chemical species to variations in the separator/electrode interface resistance due to parasitic ion deposition and layering reactions. Modeling the range of phenomena involved requires adding flexibility to the surrogate model. To that end, we combine the prediction capabilities of committees of ML classifiers by using ensemble methods [37].

The key hypothesis is that, by exploiting extended ensembles of classifiers, we can implicitly learn about the different mechanisms that can stop the execution of the simulator. The resulting composite models can then better reproduce the varying degrees of complexity of the physical phenomena that lead a simulation to an early stop, thereby improving on state-of-the-art approaches such as the one in [1].

To evaluate this hypothesis, we compare a single feed-forward network (indicated by the 'single' label in subsequent tables and diagrams), corresponding to the state of the art in [1], to a set of ensemble binary classifiers based on different approaches: a voting ensemble composing a small number of tree-based classifiers ('voting'), an efficient Gradient Boosting ensemble implementation ('xgboost'), and a stacking ensemble of models including feed-forward networks ('stacking'). The list of classifiers is in Table 3.

Table 3. Convergence classifier types.

Classification Model Type	Label
Feed-forward deep network	'single'
Voting ensemble	'voting'
Gradient boosting ensemble	'xgboost'
Deep stacking ensemble	'stacking'

We briefly introduce each model in the present section and detail implementation parameters in Section 3, dedicated to experiments.

2.2.2. Feed-Forward Deep Network

We start by establishing a baseline for the performance of convergence estimators, by considering a deep learning classifier as per the state of the art in [1], as a reference against which the performance of the rest of the models can be quantitatively evaluated. To that end, we define a fully connected network with a single hidden layer, considering a log-entropy loss function suited for classification [38], and an ADAMW optimizer [39] to address possible weight decay issues [40]. The limited structural complexity of the network corresponds to the results published in the literature.

2.2.3. Voting Ensemble of Classifiers

We build a first ensemble classifier by combining three simple models. The combination operates by basic voting among level-1 members, listed in Table 4. We describe them briefly. A pruned decision tree is one in which sections that are not critical to reducing an empirical loss function are iteratively eliminated. The pruning operates by considering the Minimum Description Length principle to reduce the risk of over-fitting [41]. The second type of level-1 model is a decision stump, a basic single-level decision tree [42]. The third type of level-1 model considered is a random forest, in itself an ensemble of decision trees [43] that outputs the mode of the output of the trees that compose it.

Table 4. Voting ensemble instance types.

Voting Ensemble Member Type	Instances
Pruned decision tree	1
Decision stump optimized by AdaBoost	1
Random forest	1

We then consider a Gradient Boosting implementation in XGBoost [44]. This specific model is selected on account of its flexibility and of the robustness of available implementations, which draw on efficient linear model solvers and effective tree learning algorithms. We complete the set of classifiers by taking into account a further model ensemble that includes instances of the feed-forward network. In this last case, the stacking is performed by a second-level classifier that takes as input the outputs of the first-level classification models that compose the committee. In this sense, the predictions of the first-level models are combined by an ensemble bagging/boosting combination mechanism that is based on a 'stacking' second-level classifier.

The models considered in the set are listed in Table 5. The three level-1 instances of the feed-forward network in the ensemble are similar in structure yet independent of one another, in the sense that they are independently trained on non-overlapping subsets of the data set.

Table 5. Stacking ensemble instance types.

Stacking Ensemble Member Type	Instances
Feed-forward network	3
Pruned decision tree	1
Decision stump optimized by AdaBoost	1
Random forest	1

Detailed descriptions of the models, together with the results obtained on a test data set, are reported and compared in Section 3.

2.2.4. Ragone Variable Regression

Regression is carried out separately on the integrated energy and power values generated by the simulation. The input data set is filtered considering only the samples that correspond to complete propagation of the model through time. The integrated energy and power values are approximated by regressors whose functional structure has a basis in the physical model. In particular, a polynomial structure is extended to a more general one, by considering an exponential transformation of the ionic conductivity of the electrolyte

$$\hat{V}(\{\vartheta_i\}_{i\in\Theta}) = \sum_{j=0}^{N} a_j \vartheta_i^j \qquad \text{with } a_j \in \mathbb{R}, \;\; \vartheta_i \in \{\theta_i, \exp(\theta_i)\}_{i\in\Theta} \tag{1}$$

so that an estimate $\hat{V}$ for one of the integrated Ragone integrals (energy or power) is expressed in terms of a polynomial of order N of the Θ input parameters θ_i of the simulation as well as of their exponential $\exp(\theta_i)$. This inclusion of the exponentiation of the parameters of the simulation is motivated by the structural form of the Butler–Volmer equation, as well as by the solution implied by the conservation and diffusion equations in the model. Note that, for simplicity, neither temperature nor concentration dependence is assumed for parameters such as the ionic conductivity in the present contribution. We plan to address this relation in future extensions of this work.

3. Results

Simulated data consists of 11 input and 3 output attributes, as detailed in Table 1. The outputs are not temporally located, since they represent the integration of the state of a simulation that is successfully extended until a given time limit. Concerning the available data and parameter measurements, we observe that uncertainties from sensors and experimental measurements are not considered at this stage. We believe that sensor noises are more critical in BMS development, as they could have a big impact on the performance of the numerical algorithms used to estimate the state variables. They can be incorporated in treatments such as the present one by e.g., Variational Inference techniques, modifying the output layers of the network architectures considered and the loss functions specified.

3.1. Simulation Convergence Prediction by Classification

We partition the data set into separate training, testing and validation sets by 60/20/20% stratified sampling without replacement. Consequently, the distribution of the simulation convergence status is preserved in the generated sets. The data points from each set are shown in Figure 4. We use K-fold validation [35] as a strategy for the measurement of performance of the classification methods, with the K parameter set to 8. The input variables are normalized by a Z-score transformation in the preprocessing step. The randomized data are organized in batches of 32 samples to speed up the learning [45,46]. While the validation subset is fixed, the train/test split varies by $k \in \{1 \dots K\}$ in the K-fold validation procedure. The values of the input simulation parameters tend to be uniformly distributed in the input data set, with a maximum absolute value of the Pearson correlation between variables around 0.19.

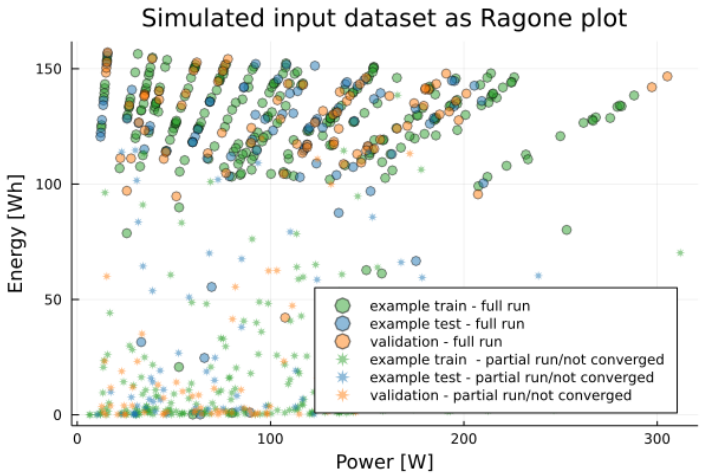


Figure 4. Simulated input data set as Ragone power/energy plot, describing an example partition of the data into model training, testing, and validation sets by stratified sampling without replacement. Asterisks are used to mark partial and non-converging runs. Samples in the 20% validation set are in orange. Samples from an example training/testing 60%/20% split are depicted in green and blue, respectively. While the validation subset is fixed, the train/test split varies by k in the K-fold validation procedure.

To establish baseline results, we first consider a state-of-the-art single fully connected feed-forward network as in [1]. The hyper-parameters for the surrogate are summarized in Table 6. The number of epochs is experimentally limited to 15, observing the convergence plots produced during the learning phase. As a consequence of the limited number of training epochs, training time is limited to five seconds on a single Intel i7-8550U CPU device running at 1.80 GHz. Though a GPU can be used as a learning device, the fast convergence of the learning procedure means that learning directly on the CPU is practically usable. The Confusion Matrix for the single feed-forward network for simulation convergence estimation as trained with the whole training/testing set and measured on the validation set is reported in Table 7. The results for this fully connected deep classifier are included and compared in Figure 5.

Table 6. Feed-forward convergence classifier configuration.

Batch size	32
Optimizer	ADAMW [47]
Number of epochs	15
Loss function	Mean Absolute Error
Hidden layer neurons	$\Theta \times 4$
Number of hidden layers	1
Activation functions	tanh
Dropout	0.2

Table 7. Confusion Matrices for a single run of the simulator convergence classification models: feed-forward network (a), voting ensemble (b), gradient boosting ensemble (c), deep stacking ensemble (d). The results for the single run may not be representative of the full results obtained via the K-fold validation in Figure 5.

(a)				
		True convergence status		
		Positive	Negative	Total
Convergence estimate	Positive	90.0	7.0	97.0
	Negative	9.0	94.0	103.0
	Total	99.0	101.0	200.0
(b)				
		True convergence status		
		Positive	Negative	Total
Convergence estimate	Positive	93.0	10.0	103.0
	Negative	6.0	91.0	97.0
	Total	99.0	101.0	200.0
(c)				
		True convergence status		
		Positive	Negative	Total
Convergence estimate	Positive	95.0	10.0	105.0
	Negative	4.0	91.0	95.0
	Total	99.0	101.0	200.0
(d)				
		True convergence status		
		Positive	Negative	Total
Convergence estimate	Positive	92.0	7.0	99.0
	Negative	7.0	94.0	101.0
	Total	99.0	101.0	200.0

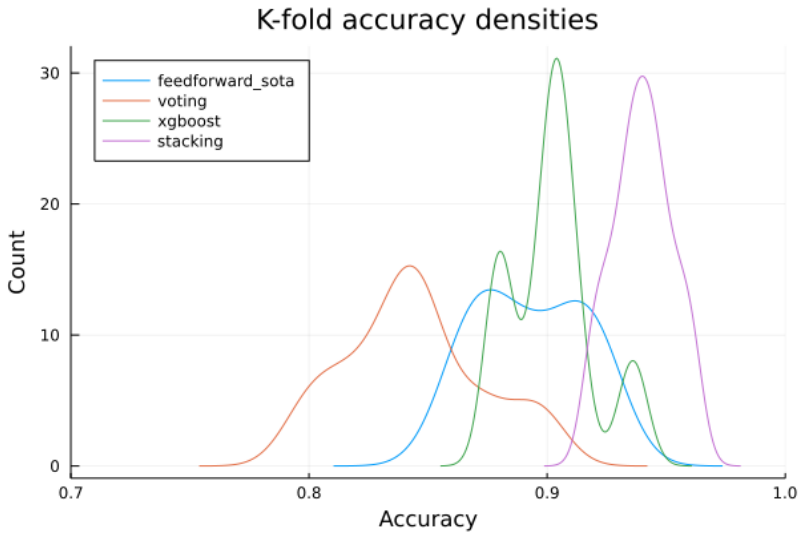


Figure 5. K-fold validation accuracy densities for the simulation convergence classifiers considered in the experiments. The curves indicate that a stacking deep ensemble ('stacking') can outperform a state of the art fully connected network ('feedforward_sota') as well as Gradient Boosting and voting ensembles (respectively 'xgboost' and 'voting').

A first ensemble model implements a voting committee with simple tree-based models as described in the above Section 2.2.1. The confusion matrix is reported in Table 7, while K-fold and validation results are summarized and compared in Figure 5.

After that, a gradient boosting ensemble classifier is configured with the hyper-parameter values in Table 8. At each training cycle, an ensemble of decision trees with a predefined maximum depth is improved through adding estimators for residuals to globally optimize a given objective function [48]. In our case, the maximum depth and the number of training cycles for the classifier are both set to two, while the objective taken into account for the loss function minimization is a binary logistic function. The step size shrinkage hyper-parameter used in the updating steps to prevent over-fitting is set to one, to indicate that no shrinking should take place. The Confusion Matrix for the gradient boosting simulation convergence estimator trained with the whole training/testing data and validated on the validation set is reported in Table 7. The resulting performance indicators for the gradient boosting ensemble classification model are included in Figure 5.

Table 8. Boosting ensemble classifier configuration.

Number of rounds	2
Maximum depth	2
Step size shrinkage	1
Objective function	binary logistic.

Finally, we consider an ensemble convergence model operating by a second-level classifier stacking the results obtained by first-level classifiers as per Section 2.2.1. The Confusion Matrix for the stacking ensemble simulation convergence model is reported in Table 7. The performance for the simulator convergence prediction of the stacking ensemble model is again reported in Figure 5.

3.2. Regression

Regression is carried out separately on the integrated energy and power values generated by the simulation. The input data set is filtered considering only the samples

that correspond to the successful propagation of the model through time. For both models, we consider 70% of the samples for training and the remaining 30% of the samples for testing.

3.2.1. Energy

The distribution of the energy integral is shown in Figure 6. Values lower than 80 J are considered as outliers and filtered out. By considering the correlations between the energy and the input variables, a first simple model is proposed to predict the values of the energy integral, $\hat{E}$, in the form of Equation (2):

$$\hat{E} = a \cdot L_p + b \cdot e^{c \cdot k_e} + d \tag{2}$$

L_p being the thickness of the positive electrode and k_e the ionic conductivity of the electrolyte. The resulting mean squared error and the R^2 score are $MSE = 32.83$ and $R^2 = 0.848$, respectively. In order to improve these results, a more complete model is introduced, as shown in Equation (3):

$$\hat{E} = a \cdot L_p + b \cdot e^{c \cdot k_e} + d \cdot L_n + e \cdot C_{rate} + f \tag{3}$$

where L_n represents the thickness of the negative electrode and C_{rate} is the applied current as a C-rate. Doing so results in a mean squared error of $MSE = 19.30$ and a $R^2 = 0.911$. Both models can be compared as in Figure 7a.

To ensure that the models have the desired accuracy and variance, some cross-validation is needed. To avoid the possibility of high bias in cases of limited data, a K-Fold cross-validation technique is used with $K = 30$, which means that the data set is split into 30 different groups. The value for K is chosen so that each train/test group of data samples is large enough to be statistically representative of the broader data set. The R^2 and the MSE are compared to those obtained when creating the model in Figure 7b,c.

From Figure 7b,c, it can be concluded that, in the Simple Model, around 20 out of 30 times, the R^2 values are higher than 0.8, and the same number of times, the MSE is below 25. For the Complete Model, around 20 of 30 times, the R^2 is higher than 0.9 and the MSE lower than 20. The models behave as expected and are validated by comparing these values with the ones obtained first in the models.

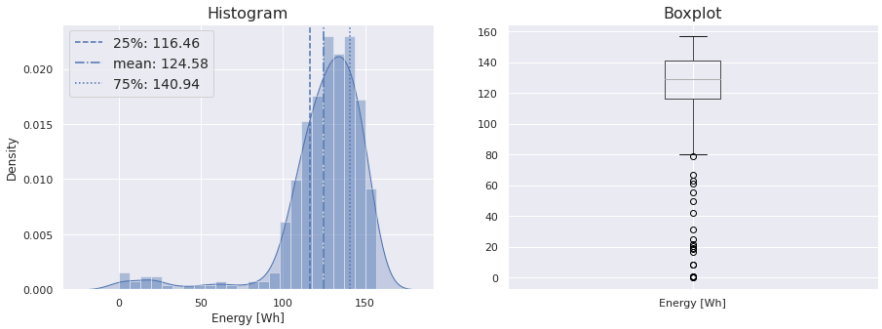


Figure 6. Distribution of the integrated energy variable (output of the simulation). Histogram (**left**) and box-plot (**right**) of the variable. As shown in the box-plot, values lower than 80 J are considered as outliers in this distribution.

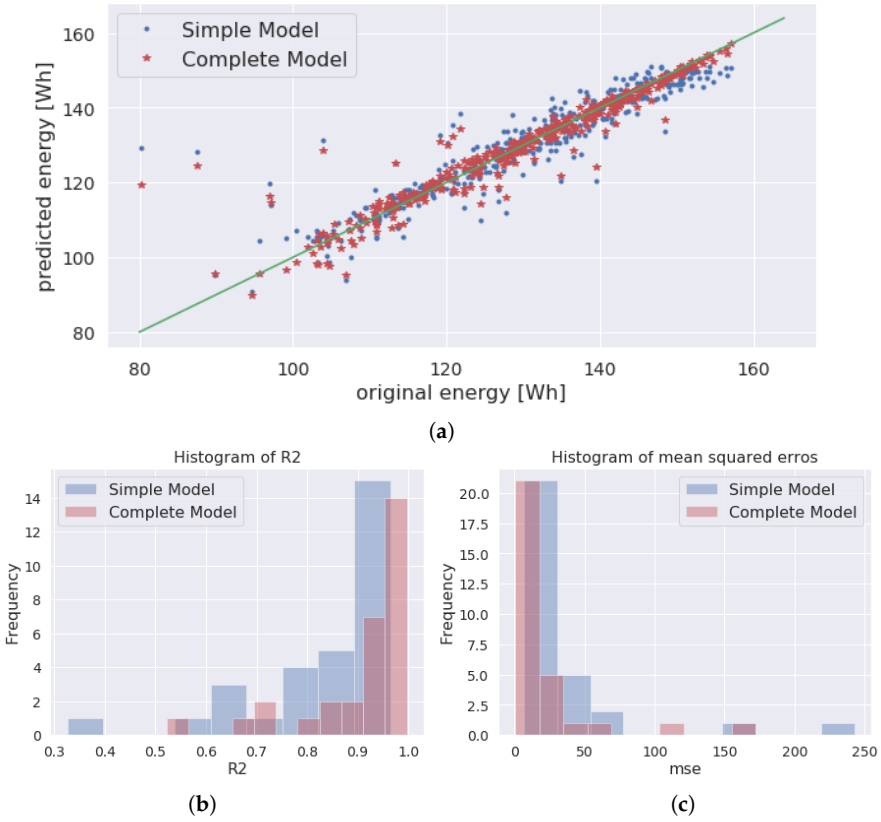


Figure 7. Results of the proposed regression models for the energy output integral (**a**) and histograms of the MSE (**b**) and R^2 (**c**) of 30 K-Folds performed to the model for the energy.

3.2.2. Power

The distribution of the power integral of simulation outputs is shown in Figure 8. Values higher than 285 W can be considered as outliers, and, therefore, the dataset is again filtered for these values. The power integral presents a significantly higher correlation with the applied current and the thickness of the positive electrode, compared to that of other input variables. Thus, a first simple linear regression is proposed to predict the values of the power integral, $\hat{P}$ based on the following equation:

$$\hat{P} = a \cdot C_{rate} + b \cdot L_p + c \tag{4}$$

with C_{rate} the C-rate and L_p the thickness of the positive electrode. The results obtained by optimizing the constants in Equation (4) can be observed in Figure 9a. The resulting mean squared error and the R^2 score would be $MSE = 183.49$ and $R^2 = 0.988$, respectively.

As in the case of energy estimates, K-fold cross-validation is performed in the developed model, with $K = 30$ meaning that the dataset is again split into 30 different groups. R^2 and MSE values are compared to those obtained when creating the model. The results for the power model are shown in Figure 9b,c.

From Figure 9b,c, it can be concluded that, 20 out of 30 times, the R^2 values are above 0.985, and that, although there are instances for which the MSE's values are high, 20 out of 30 times, this MSE is below 60.

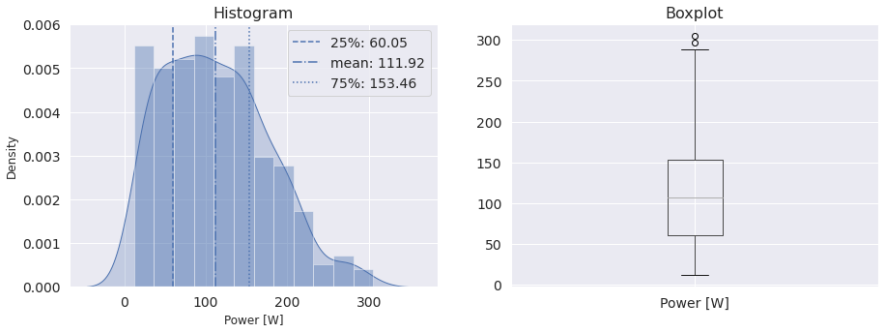


Figure 8. Distribution for the integrated power simulation output.

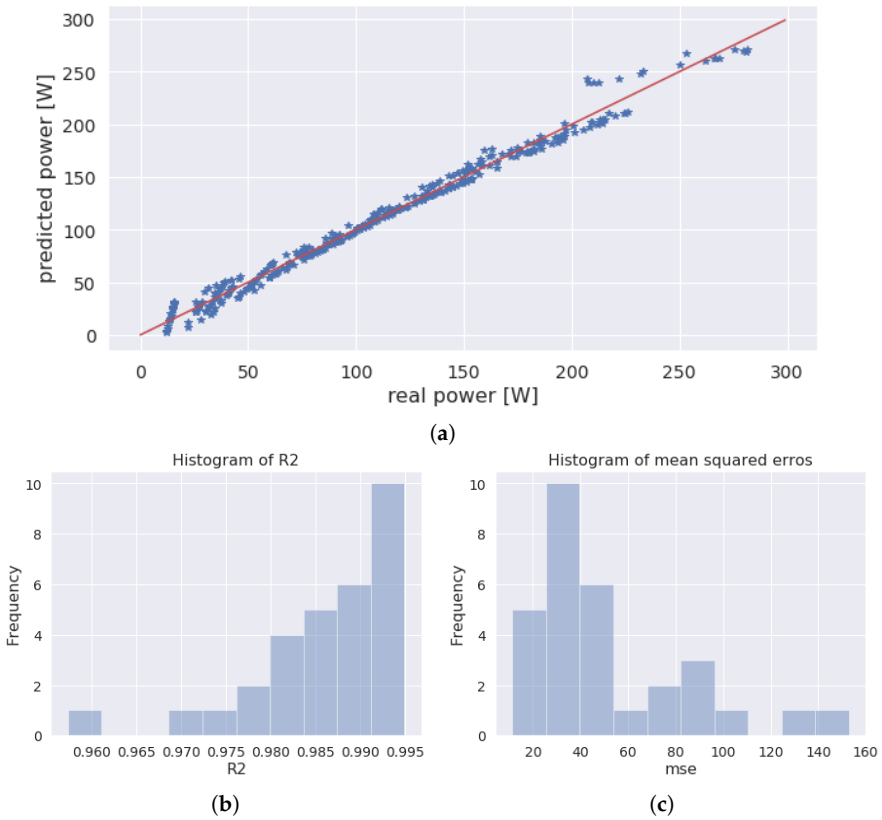


Figure 9. Results of the proposed regression model for the power integral (**a**) and histograms of the MSE (**b**) and R^2 (**c**) of 30 K-folds performed to the model for the power integral estimation.

4. Discussion

The results above indicate that an effective way to speed up the characterization of LIB designs involves complementing simulations with empirical, ML surrogate models learned from simulation results [33]. In terms of computational performance, P2D simulations using CIDETEC's proprietary code take, on average, 20 seconds for a 1C discharge. Once the surrogate models are trained, instead, predictions can be obtained in a few hundredths of a second. Deep ensembles and structured regression models represent an interesting direction for the development of efficient surrogate models capable of rapidly and accurately predicting Ragone plot variables such as energy and power integrals. Those results can

be integrated with design workflows and extended to other fields such as management and control. In the specific case of the prediction of early termination of the simulator by classification, the results seem to indicate that ensemble models can indeed outperform state-of-the-art surrogates based on deep learning fully connected networks, exploiting their flexibility in implicitly learning the multiple mechanisms that can stop the execution of the simulator.

On the other hand, it is perhaps worth mentioning that an important point that is sometimes overlooked [49] is that physical simulations are complemented rather than substituted by such data-based modeling: some of the computational costs are moved in time rather than eliminated, since some simulated data need to be generated first for the learning to take place.

5. Conclusions and Future Work

The present contribution has introduced and evaluated composite surrogate models for the prediction of the performance of Lithium-Ion Batteries which combine classifiers based on deep ensembles and structured regressors for the prediction of energy and power densities. We have compared a single state-of-the-art feed-forward network to three types of ensemble binary classifiers for the prediction of simulation convergence. The quantitative results obtained indicate that ensemble models can indeed outperform state of the art models based on a single deep network. Furthermore, we have quantitatively validated the applicability of structured regressors to the estimation of LIB energy and power. Future work to be considered includes analyzing feature and parameter sensitivities [50] and the use of ML explainability/interpretability methods to ensemble models from an electrochemical perspective. Furthermore, model and data uncertainty management [51,52] should also be analyzed: the quantification of the risk from different sources is valuable for the real-world applicability of the developed methodology.

Author Contributions: Physical simulation, P.C. and E.A.; classification surrogates: M.Q.; regression models, A.G., A.M.F.-T.; funding acquisition, I.G.O. and E.A; text revision: all. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to their proprietary nature.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wu, B.; Han, S.; Shin, K.G.; Lu, W. Application of artificial neural networks in design of lithium-ion batteries. *J. Power Sources* **2018**, *395*, 128–136. [CrossRef]
2. Bizeray, A.M.; Zhao, S.; Duncan, S.R.; Howey, D.A. Lithium-ion battery thermal-electrochemical model-based state estimation using orthogonal collocation and a modified extended Kalman filter. *J. Power Sources* **2015**, *296*, 400–412. [CrossRef]
3. Doyle, M.; Newman, J.; Gozdz, A.S.; Schmutz, C.N.; Tarascon, J.M. Comparison of modeling predictions with experimental data from plastic lithium ion cells. *J. Electrochem. Soc.* **1996**, *143*, 1890. [CrossRef]
4. Kumaresan, K.; Sikha, G.; White, R.E. Thermal model for a Li-ion cell. *J. Electrochem. Soc.* **2007**, *155*, A164. [CrossRef]
5. Doyle, M.; Fuller, T.F.; Newman, J. Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell. *J. Electrochem. Soc.* **1993**, *140*, 1526. [CrossRef]
6. Newman, J.; Thomas-Alyea, K.E. *Electrochemical Systems*; John Wiley & Sons: Hoboken, NJ, USA, 2012.
7. Santhanagopalan, S.; Guo, Q.; Ramadass, P.; White, R.E. Review of models for predicting the cycling performance of lithium ion batteries. *J. Power Sources* **2006**, *156*, 620–628. [CrossRef]
8. Bernardi, D.; Pawlikowski, E.; Newman, J. A general energy balance for battery systems. *J. Electrochem. Soc.* **1985**, *132*, 5. [CrossRef]
9. Bandhauer, T.M.; Garimella, S.; Fuller, T.F. A critical review of thermal issues in lithium-ion batteries. *J. Electrochem. Soc.* **2011**, *158*, R1. [CrossRef]

10. Moura, S.J.; Bribiesca Argomedo, F.; Klein, R.; Mirtabatabaei, A.; Krstic, M. Battery State Estimation for a Single Particle Model with Electrolyte Dynamics. UC Berkeley: Energy, Controls, and Applications Lab. Available online: https://escholarship.org/uc/item/5xt11817 (accessed on 7 July 2021) .
11. Subramanian, V.R.; Ritter, J.A.; White, R.E. Approximate solutions for galvanostatic discharge of spherical particles i. constant diffusion coefficient. *J. Electrochem. Soc.* **2001**, *148*, E444. [CrossRef]
12. Majdabadi, M.M.; Farhad, S.; Farkhondeh, M.; Fraser, R.A.; Fowler, M. Simplified electrochemical multi-particle model for LiFePO4 cathodes in lithium-ion batteries. *J. Power Sources* **2015**, *275*, 633–643. [CrossRef]
13. Dao, T.S.; Vyasarayani, C.P.; McPhee, J. Simplification and order reduction of lithium-ion battery model based on porous-electrode theory. *J. Power Sources* **2012**, *198*, 329–337. [CrossRef]
14. Smith, K.A.; Rahn, C.D.; Wang, C.Y. Control oriented 1D electrochemical model of lithium ion battery. *Energy Convers. Manag.* **2007**, *48*, 2565–2578. [CrossRef]
15. Jokar, A.; Rajabloo, B.; Désilets, M.; Lacroix, M. Review of simplified Pseudo-two-Dimensional models of lithium-ion batteries. *J. Power Sources* **2016**, *327*, 44–55. [CrossRef]
16. Nejad, S.; Gladwin, D.; Stone, D. A systematic review of lumped-parameter equivalent circuit models for real-time estimation of lithium-ion battery states. *J. Power Sources* **2016**, *316*, 183–196. [CrossRef]
17. Seaman, A.; Dao, T.S.; McPhee, J. A survey of mathematics-based equivalent-circuit and electrochemical battery models for hybrid and electric vehicle simulation. *J. Power Sources* **2014**, *256*, 410–423. [CrossRef]
18. Wang, Y.; Yang, D.; Zhang, X.; Chen, Z. Probability based remaining capacity estimation using data-driven and neural network model. *J. Power Sources* **2016**, *315*, 199–208. [CrossRef]
19. Wei, J.; Dong, G.; Chen, Z. Remaining useful life prediction and state of health diagnosis for lithium-ion batteries using particle filter and support vector regression. *IEEE Trans. Ind. Electron.* **2017**, *65*, 5634–5643. [CrossRef]
20. Geng, Z.; Wang, S.; Lacey, M.J.; Brandell, D.; Thiringer, T. Bridging physics-based and equivalent circuit models for lithium-ion batteries. *Electrochim. Acta* **2021**, *372*, 137829. [CrossRef]
21. Li, Y.; Vilathgamuwa, M.; Farrell, T.; Choi, S.S.; Tran, N.T.; Teague, J. A physics-based distributed-parameter equivalent circuit model for lithium-ion batteries. *Electrochim. Acta* **2019**, *299*, 451–469. [CrossRef]
22. Jones, D.R. A taxonomy of global optimization methods based on response surfaces. *J. Glob. Optim.* **2001**, *21*, 345–383. [CrossRef]
23. Forrester, A.I.; Keane, A.J. Recent advances in surrogate-based optimization. *Prog. Aerosp. Sci.* **2009**, *45*, 50–79. [CrossRef]
24. Zhang, Y.; Han, Z.H.; Zhang, K.S. Variable-fidelity expected improvement method for efficient global optimization of expensive functions. *Struct. Multidiscip. Optim.* **2018**, *58*, 1431–1451. [CrossRef]
25. Regis, R.G.; Shoemaker, C.A. Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. *Eng. Optim.* **2013**, *45*, 529–555. [CrossRef]
26. Krityakierne, T.; Akhtar, T.; Shoemaker, C.A. SOP: Parallel surrogate global optimization with Pareto center selection for computationally expensive single objective problems. *J. Glob. Optim.* **2016**, *66*, 417–437. [CrossRef]
27. Liu, K.; Li, Y.; Hu, X.; Lucu, M.; Widanage, W.D. Gaussian process regression with automatic relevance determination kernel for calendar aging prediction of lithium-ion batteries. *IEEE Trans. Ind. Inform.* **2019**, *16*, 3767–3777. [CrossRef]
28. Liu, K.; Hu, X.; Wei, Z.; Li, Y.; Jiang, Y. Modified Gaussian process regression models for cyclic capacity prediction of lithium-ion batteries. *IEEE Trans. Transp. Electrif.* **2019**, *5*, 1225–1236. [CrossRef]
29. Liu, K.; Shang, Y.; Ouyang, Q.; Widanage, W.D. A data-driven approach with uncertainty quantification for predicting future capacities and remaining useful life of lithium-ion battery. *IEEE Trans. Ind. Electron.* **2020**. [CrossRef]
30. Gal, Y.; Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 1050–1059.
31. Graves, A. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*; Neural Information Processing Systems Foundation, Inc. (NIPS): Granada, Spain, 12–15 December 2011; Volume 24, pp. 2348–2356.
32. Ascione, F.; Bianco, N.; De Masi, R.F.; De Stasio, C.; Mauro, G.M.; Vanoli, G.P. Artificial neural networks for predicting the energy behavior of a building category: A powerful tool for cost-optimal analysis. In *Cost-Effective Energy Efficient Building Retrofitting*; Elsevier: Amsterdam, The Netherlands, 2017; pp. 305–340.
33. Ascione, F.; Bianco, N.; De Stasio, C.; Mauro, G.M.; Vanoli, G.P. CASA, cost-optimal analysis by multi-objective optimisation and artificial neural networks: A new framework for the robust assessment of cost-optimal energy retrofit, feasible for any building. *Energy Build.* **2017**, *146*, 200–219. [CrossRef]
34. Ragone, D.V. *Review of Battery Systems for Electrically Vehicles*; SAE Technical Paper Series; SAE International: Warrendale, PA, USA, 1968.
35. Fushiki, T. Estimation of prediction error by using K-fold cross-validation. *Stat. Comput.* **2011**, *21*, 137–146. [CrossRef]
36. Smith, K.; Wang, C.-Y. Solid-state diffusion limitations on pulse operation of a lithium ion cell for hybrid electric vehicles. *J. Power Sources* **2006**, *161*, 628. [CrossRef]
37. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1249. [CrossRef]
38. MacKay, D.J.; Mac Kay, D.J. *Information Theory, Inference and Learning Algorithms*; Cambridge University Press: Cambridge, UK, 2003.
39. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv* **2017**, arXiv:1711.05101.
40. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.

41. Breslow, L.A.; Aha, D.W. Simplifying decision trees: A survey. *Knowl. Eng. Rev.* **1997**, *12*, 1–40. [CrossRef]
42. Iba, W.; Langley, P. Induction of one-level decision trees. In *Machine Learning Proceedings 1992*; Elsevier: Amsterdam, The Netherlands, 1992; pp. 233–240.
43. Friedman, J.; Hastie, T.; Tibshirani, R. *The Elements of Statistical Learning*; Springer Series in Statistics New York; Springer: New York, NY, USA, 2001; Volume 1.
44. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; KDD '16; ACM: New York, NY, USA, 2016; pp. 785–794.
45. Keskar, N.S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; Tang, P.T.P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv* **2016**, arXiv:1609.04836.
46. Masters, D.; Luschi, C. Revisiting small batch training for deep neural networks. *arXiv* **2018**, arXiv:1804.07612.
47. Lydia, A.; Francis, S. Adagrad—An optimizer for stochastic gradient descent. *Int. J. Inf. Comput. Sci.* **2019**, *6*, 566–568.
48. Friedman, J.H. Stochastic gradient boosting. *Comput. Stat. Data Anal.* **2002**, *38*, 367–378. [CrossRef]
49. Breen, P.G.; Foley, C.N.; Boekholt, T.; Zwart, S.P. Newton vs. the machine: Solving the chaotic three-body problem using deep neural networks. *arXiv* **2019**, arXiv:1910.07291.
50. Liu, K.; Hu, X.; Zhou, H.; Tong, L.; Widanalage, D.; Marco, J. Feature analyses and modelling of lithium-ion batteries manufacturing based on random forest classification. *IEEE/ASME Trans. Mech.* **2021**. [CrossRef]
51. Kläs, M.; Vollmer, A.M. Uncertainty in machine learning applications: A practice-driven classification of uncertainty. In *International Conference on Computer Safety, Reliability, and Security*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 431–438.
52. Hüllermeier, E.; Waegeman, W. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Mach. Learn.* **2021**, *110*, 457–506. [CrossRef]