POLITECNICO DI TORINO

Master's Degree in ICT For Smart Societies

Master's Degree Thesis

Toward a zero defects manufacturing process with Artificial Intelligence applications

Supervisors

Prof. Danilo GIORDANO

Prof. Marco MELLIA

Prof. Tania CERQUITELLI

Candidate

Gennaro RENDE

16th April 2021

Summary

Nowadays, the concept of "*Industry 4.0*" is becoming more relevant, and one of its innovations is the shift from periodic maintenance to predicted one. In this context, manufacturing companies extensively use Artificial Intelligence and Machine Learning to predict when any of their machines require maintenance [1]. A machine's failure could translate into producing waste or into time losses to recover the machines; predictive maintenance forecasts eventual needs of technical assistance and allows scheduling maintenance before a failure occurs.
This thesis project evaluates predictive maintenance (PdM) feasibility in laser welding processes, focusing on applying it to car components manufactured by Fiat Chrysler Automobiles (FCA) using machine learning algorithms. Here, the assessment to implement PdM is evaluated with the analysis of *power signals* and their relative *metadata* recorded in one of FCA's factories during production. Correctly detecting when a machine is about to output an inadequate weld would consent to predict and schedule maintenance.
The methodology used is the data-driven one. *Power signals* and their *metadata* such as mean, standard deviation, the signal's energy, and the signals' outcome (defined with a label of *good* or *bad*), are examined to determine if a weld is more likely to have an unsuccessful output. In the first half of the project, the focus is on the *metadata*: our goal is to evaluate if they are meaningful enough to classify *good* and *bad* signals correctly. The most relevant *metadata* are selected to test labeling signals with supervised analysis using linear classification (Decision-Tree) and non-linear classification (Support Vector Machine). Additional features are extracted from the raw *power signals* by comparing a signal's subsets' statistics with its reference of the same production day. Afterward, to compare these statistics and determine if they are proper of a particular label or another, a specifically tailored heatmap visualization is developed.
The *metadata* analysis showed no correlation between features and labels: it is impossible to precisely identify faulty welds given how much *good* and *bad* signals' features are similar. The analysis of *power signals* shows similar statistics both at the beginning and end of the time series, but no difference is found between *good* and *bad* signals.

The analysis of the *metadata* and the comparison methodology developed for the *power signals* contribute to exploratory analysis of the given dataset and evaluation of PdM's feasibility in the context of laser weldings.
FCA positively evaluated this thesis's results, finding it interesting how the multiple time-series statistics were represented and how they proved the presence of commonalities between the same production day signals.

Table of Contents

Acknowledgements

I cannot begin to express my thanks to Professor Danilo Giordano for his unwavering guidance and insightful suggestions, for which I will always be grateful. His teachings helped me learn more not only about the world of data analysis but also on how to be a constructive and thoughtful leader. I would also like to extend my deepest gratitude to Professor Marco Mellia for giving me the chance to work with him and be part of a stimulating environment such as "Smart Data." Thanks also to Eng. Giorgio Pasquettaz who was instrumental in this project's development with FCA.

This thesis represents the conclusion of my academic career, and is dedicated to my dad Antonio and my mother, Anna Maria. This achievement would not have been possible without your support and unconditional love. Thank you for always putting me first and teaching me how to be strong.

I could never be grateful enough to my sister, Alida. She encouraged me, taught me to be proud of myself, and motivated me to battle in the most challenging times. You are my role model.

Thanks to my family and especially to my grandmothers Assunta e Palmina for always making me feel loved and appreciated.

Thanks to all my friends who helped me on the way of this incredible journey. I love you and you are always in my heart.

List of Tables

List of Figures

Acronyms

PdM
Predictive Maintenance

CM
Corrective Maintenance

PM
Time-based Preventive Maintenance

CBM
Advanced Condition-Based Maintenance

AI
Artificial Intelligence

LV
LabView™

ML
Machine Learning

CSV
Comma-Separated values

TXT
TeXT

CRF
Centro Ricerche Fiat

DB

DataBase

DT

Decision Tree

SVM

Support Vector Machine

RBF

Radial Basis Function

FT

Fourier Transform

IFT

Inverted Fourier Transform

FFT

Fast Fourier Transform

LPF

Low Pass Filter

LR

Linear Regression

SSR

Sum of Squared Residuals

Chapter 1

Introduction

1.1 Background of the project

Throughout history, the world of industry went through four revolutions and introduced groundbreaking technologies and innovations, from steam machines enabling mechanized processes to informatics and electronics for automatizing operations. In the last decade, the rapid advances in digitalization and automation methods radically evolved industrial working pipelines, bringing a fourth revolution: the "Industry 4.0". The term "Industry 4.0" was introduced in 2011 at the "Hannovermesse" (Hannover's fair) by Henning Kagermann, Wolf-Dieter Lukas and Wolfgang Wahlster [2] who publicly used it in their publication called: "*Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4 industriellen Revolution*" (Industry 4.0: With the Internet of Things on the way to the 4th industrial revolution)[2][3]. Industry 4.0, including a technological mix of sensors, robotics, connection protocols, and programming, represents a new revolution in manufacturing products and organizing work. By exploiting new production models based on cloud-interconnected systems and big data management, mass production is destined to become more efficient and customizable. The continuous evolution of technologies diversifies the dissemination of 4.0 across multiple levels and operational areas associated with Artificial Intelligence and all digital derivatives. One of the main concepts of Industry 4.0 is the shift from the periodic maintenance of industrial machines to a predicted one. Predicting when a machine will require maintenance in the future is still a relevant challenge. However, the benefits are multiple: the enhancement of machine downtime, costs, control, and quality of production [4]. Fiat Chrysler Automobiles, one of Italy's most important car manufacturers, employs industry 4.0's ' innovations, including predictive maintenance. In collaboration with FCA's research center (Centro Ricerche Fiat), this project evaluated PdM's feasibility in laser welding operations based on data-driven analyses and classifications with

Artificial Intelligence tools.

1.2 Objective of the project

This thesis project aims at developing a methodology that would correctly identify erroneous weld outputs; detecting when a weld is not proper is an essential indicator of a need for maintenance for the welding machine. Therefore, to comprehend the specifics under which a weld is considered to be not sufficiently good, the analysis focused on a dataset of power signals (time-series) and their relative metadata. This dataset was built by Centro Ricerche Fiat (CRF) and contains signals recorded during laser welding of components for heavy vehicles' gearboxes such as Fiat Professional, Fiat Ducato, or IVECO. For this project's end, it is crucial to organize and select the data that could be the most informative and beneficial to extract knowledge. The "*Knowledge Discovery in Databases*" is a methodology extensively adopted in the Data Science field for the objectives reported previously. The KDD incorporates operations for data arrangement and selection, data cleaning, and finally, Data Mining. Wikipedia defines Data Mining as *"the process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems"*[5]. Data mining is fundamental to extract implicit information with machine learning algorithms and represent those in organized patterns to analyze and retrieve latent knowledge. In the first place, the analyses focused on classifying the metadata linked to the power signals, looking for meaningful information that could help label the outcome of a weld. Given the availability of the validated labels, a supervised approach has been utilized. A second methodology consisted on performing data analytic on power signals, plotting them with different modalities, and analyzing those plots, searching for common trends and range values for each output label.

1.3 State of the Art

PdM is becoming a concept of primary importance in factory plants to improve their efficiency and maintenance cost due to its constant rise over the decades [6]. As reported by the analyses of this article written by Maurizio Bevilacqua [7], 15 - 70% of production costs are linked to maintenance, considering several types of industrial settings. Given the economic impact of maintenance, this topic has become more relevant in many industrial environments, motivating research growth towards new and better ways to perform maintenance [8]. The main evolutional phases of maintenance are the following [9]:

1. Corrective maintenance (CM);

2. Time-based preventive maintenance (PM);

3. Advanced condition-based maintenance (CBM);

4. Predictive maintenance (PdM).

The last step of the maintenance's evolution is drastically improving industrial workflows. As it is reported in [10]: "PdM has the capability of improving system reliability and reduce the quantity and severity of in-service system failures". PdM stepped into the industrial sector mainly after 2011 with the start of the revolution "Industry 4.0", which increased the adoption of innovative technologies, such as Artificial Intelligence (AI) and Cloud Computing. Industrial companies used to perform maintenance operations mostly in emergency conditions as a corrective function. However, today there are new demands, such as sustaining larger productions with a better output quality maintaining plant's safety, efficiency, and reliability. PdM is a possible solution to all of these new demands and some of its applications are reported in the following.

The project work presented by Hongfei Li [11] is an example of how machine learning is applied to predictive maintenance. This study uses a data-driven analysis with ML to include PdM in rail network monitoring systems. The inspiration comes from preexisting PdM applications in railroads where networks of wayside mechanical condition detectors are used to monitor the rolling-stock as it passes by and alerts in case of deterioration. This considered project analyzes historical detector data, failure data, maintenance action data, inspection schedule data, train type data, and weather data. The different analytical approaches consist of correlation analysis, causal analysis, time series analysis, and machine learning techniques to learn rules and build failure prediction models automatically. Those models will subsequently be applied against historical and real-time data to predict conditions leading to failure in the future, thus avoiding service interruptions and increasing network velocity.

Another noteworthy project is the one introduced by Mikel Canizo [12] which implements predictive maintenance strategies using a Big Data approach in the field of wind turbines, relying on a Cloud Computing infrastructure. The Energy Roadmap 2050, as reported in the just-mentioned article, projects that Europe's electricity supplies, around 31.6% to 48.7%, will be provided by wind turbines. Therefore it is crucial to adopt PdM methods to reduce the economic impact of maintenance due to the extensive adoption of wind turbines. With the rapid growth of the wind turbines' market, there's a proportional growth in terms of the systems' produced monitoring data. Nowadays, the daily data volumes generated by wind turbines are too large to be processed with traditional technologies. This project opted for Big Data frameworks to analyze data more efficiently, therefore improving decision-making processes. It is shown that there's a 3% increment in a company's

productivity if Big Data Frameworks are deployed. The PdM method proposed by this project is executed on a cloud computing environment such as Amazon EMR 1 or Microsoft HDInsight 2, which can be scaled horizontally as the data to be processed increases in volume. The final results of this project showed that, besides achieving a more rapid and scalable methodology, PdM's results improved thanks to a more accurate and sensitive rate of the systems.

The project MINICON [13] aims at developing a cost-effective integrated sensor processing units (SPUs) for inclusion of on-line condition monitoring systems on all range of industrial and civil machinery. This includes two objectives: the consecution of a cost-effective hardware monitoring unit (or sensor processing unit-SPU) and the availability of automated systems to perform decision support on fault diagnosis and troubleshooting. This project applies predictive maintenance to reduce the relevant maintenance costs by performing cost-effective monitoring operations and maintenance to small elements of industrial facilities (i.e. machine tools) and civil infrastructure (i.e. elevators and escalators).

The main parts of this project have been the analysis of *metadata* and *power signals*. For the latter are available multiple methodologies in literature that show how timeseries' analysis is fundamental for predictive maintenance's implementation. In this paper by Georgios Makridis [14], it is evaluated how to implement predictive maintenance in the enivornment of maritime industry, to minimize vessels' maintenance's downtime and reduce the costs of this operation. PdM is a solution to provide an optimized scheduling of maintenance processes, while extending vessels' at reduced costs. This work focuses on anomaly detection by analyzing timeseries data with machine learning, and then forecasting the condition of vessels's main engine components. The machine learning models used in this project are:

- Gradient Boosting Classification;
- Multivariate LSTM Model;
- One Class Support Vector Machine;
- Weighted Permutation Entropy.

These models have been employed all together, reveling to be an efficient tool for forecastig anomalies with timeseries analysis.

In this work [15] is evaluated the use of PdM to find out when a target device (TD) is about to die and to proceed with maintenance to avoid unexpected down times. In this case the Time Series Prediction (TSP) algorithm is used to predict the TD's remaining useful life (RUL) TSP uses historical data to forecast future trends, thus predicting when the TD is about to turn-off. The used algorithm turned out to be the best solution for prediction, since exponential models could not handle timeseries' steep rises or steep drops, making the model not useful.

Chapter 2

Dataset overview and management

This chapter presents an overview of the dataset, of the inner folders and files, along with a description of the approaches used for data management.

In the first section I describe the two file typologies contained in the Dataset:

1. Signal Files: Time series of power signals recorded during the laser welding process;

2. Metadata Files: Table containing features extracted from the signal files right after their creation.

Moreover, in the second section will be presented the adopted methodology for re-organizing, cleaning, and selecting the meaningful portions of the Dataset.

2.1 Files and Folders overview

2.1.1 Signal Files

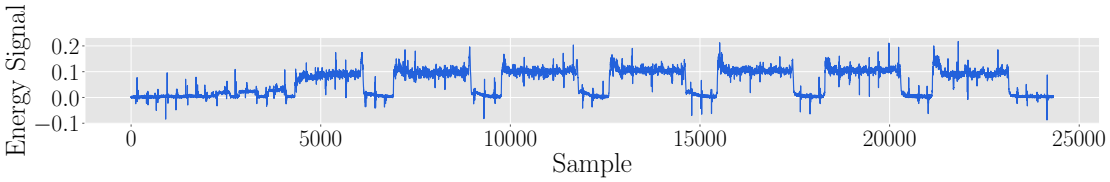
As mentioned in the introduction of this chapter, signal files are recorded during the welding of mechanical components, which can belong to 3 categories (also known with the name of velocities or geometries):

- `Velocity A` - Geometry 1;
- `Velocity B` - Geometry 2;
- `Velocity C` - Geometry 3.

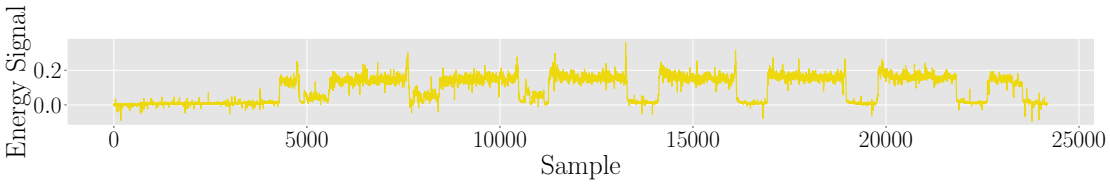
It is also available a `D Velocity` (Geometry 4), but it was not considered in the analyses for this project's purpose.

Figure 2.1 depicts the time-series for each type of geometry/velocity. One commonality they share is the presence of low power values at the first ~ 4500 samples because the welding machine is set on a "low penetration mode" to avoid thermal excursions. Moreover, at the last ~ 500 samples in the time-series, there are again low power values given the power reduction at the end of the welding. This is due to the laser beam being turned off to prevent cracks on the welded material with an excessive overheating.

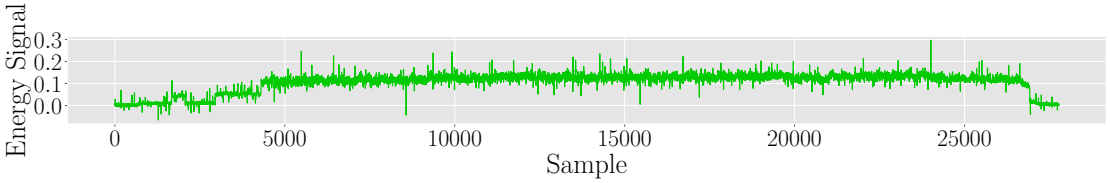
If we look instead at the intermediate part of `Velocity A` 2.1a and `Velocity B` 2.1b signals, we can notice that they have a similar trend with the presence of multiple gores, which is not a characteristic of `Velocity C`. For all of our analyses and each type of geometry, only the intermediate part will be considered (eventual gores are included). The initial and final sections of a signal are not descriptive of its characteristics.



(a) Geometry 1 Signal (`Velocity A`)



(b) Geometry 2 Signal (`Velocity B`)



(c) Geometry 3 Signal (`Velocity C`)

Figure 2.1: All the typologies of signals included in the Dataset

Besides belonging to one of the already mentioned three geometries/velocities, signal files are labeled according to the welding output they represent. If a signal represents a *good* or *bad* weld output, then its name file will report that information

accordingly. These are the type of output a signal can have:

- **Good**: the signal is representative of a good weld output;
- **Good with over penetration**: signal of a good weld with excessive over penetration;
- **Bad**: the signal is representative of a bad weld output;
- **Bad with over penetration**: signal of a bad weld with excessive over penetration;
- **Warning**: the signal is representative of a degraded weld output;
- **Warning with over penetration**: signal of a degraded weld with excessive over penetration.

If a signal is labeled with the additional information of "over penetration", it means that during the welding process, the component's material was overly penetrated by the laser. This will increase the probability of having induced thermal deformation onto the material.

The `Velocity C`

Most of this project's analyses were performed on the `Velocity C` category. This is linked mostly to two reasons:

1. Looking at 2.1c, the intermediate part of the signal is not affected by gores, which instead are present in the time-series of `Velocity A` 2.1a and `Velocity B` 2.1b. Not having gores in a time-series makes signals' analysis easier because of the reduced dynamic.
2. It includes more of all the available labels. Having a larger number of signals representing each label, is fundamental to improve the model to classify signals (this will be explained in this chapter 3).

Figure 2.2 shows an example of a `Velocity C` signal. X-axis reports the acquired samples while the Y-axis reports the energy value of each sample. The main sections of the signal are highlighted with different colors. As we described previously, the initial section (the one plotted in green) has lower power values than the intermediate part (the one plotted in blue), the same is for the final section in red, which has low power values. The intermediate section will be fundamental for the analyses of signals, since it represents the actual welding process. The initial and final sections are representative of the warming-up and cooling-down of the laser process, they do not provide information about the outcome of the weld.

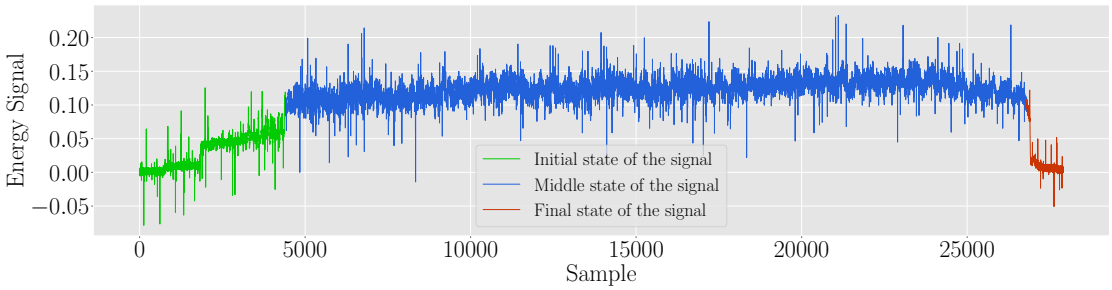


Figure 2.2: Highlighting of the initial, intermediate and final section of a `Velocity C` energy signal

Signal files recording

Signals are recorded during laser weld processes with a photo-diode in the proximity of the point of weld, see figure 2.3. The optic radiation between the laser and the material is converted into an electric signal while acquired at 2048 Hz. The acquisition of the signal is achieved with LabView's PCI 6115 capture card installed on a computer. Once all the signals of a production day are recorded, they are transferred to a CRF proprietary software. It analyzes the signals and from them extracts features such as the mean, the standard deviation, the energy and many more (see table 2.1.2 for the complete list). The new features are subsequently saved in a metadata file, along with those of the same geometry and production day. The metadata file, once completed, is named after the signals' velocity category.

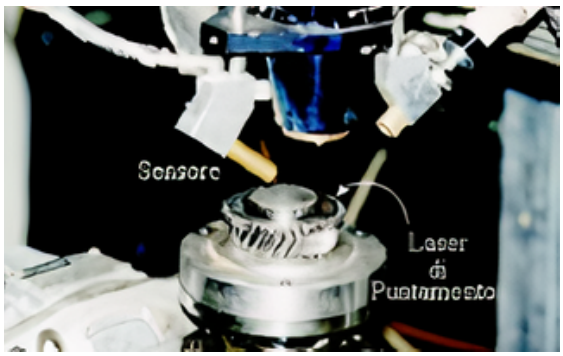


Figure 2.3: Welding Laser with the additional sensor to capture energy's radiation

Signal files conversion

The PCI 6115 capture card is produced by LabView and converts an analogical input into a proprietary digital binary format called `LabView DTLG`. This file format is only readable with the LabView software tools. Therefore, signals are not usable

right after being exported to digital; they need a conversion to a readable format such as the `.txt` one.

Figure 2.4 depicts the main blocks of the LabView script to convert signal files. The algorithm reads all the signal files' names from a folder selected by the user, creates a list with these names, and cycles over this list to convert one file at each iteration. One iteration completes a set of operations:

- Reading the file's name from the list;
- Determining the file's path in the computer;
- Selecting the file;
- Converting the file to text format;
- Appending the `".txt"` string to the file's name;
- Exporting the file to the same path determined in step 2.

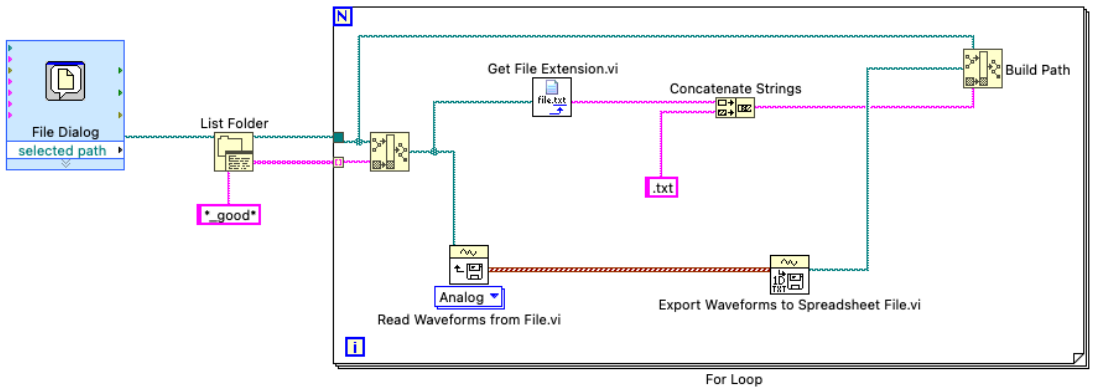


Figure 2.4: Workflow for converting a trial file in LabView

For better usage after being converted into text format, signals have been converted into `CSV` with a Python script. LabView cannot perform conversions into `CSV` directly, therefore it was mandatory to convert them first into `txt`.

2.1.2 Metadata Files

Metadata files are created once a welding session for a specific geometry is completed. All the signals recorded during a session are analyzed by Centro Ricerche Fiat (CRF) software, and multiple features for each signal are derived. A metadata file is a table of 10 columns with many rows as the number of signal files recorded for that specific welding session. All the Metadata files are exported with the `.txt` file format. In Table 2.1 is shown a preview of what is contained inside of a metadata

file. Each column is referred to a specific feature of the signals. Every row is linked to a specific signal file. One of the main tasks of this project, in terms of data management, was to to correctly associate metadata with their relative signal files, since no relation between the two is expressed explicitly.

Quality	Constant Penetration	Percentage Porosity	Energy Signal	Mean Signal	Deviance	Type Acquisition	Length Acquired Signal	Length Analyzed Signal	Reference
0.000000	NaN	NaN	NaN	NaN	0.000000	10.000000	0.000000	0.000000	1.000000
1.000000	100.000000	0.555421	2.927530	0.034158	0.006864	10.000000	16564.000000	16564.000000	1.000000
0.000000	NaN	NaN	NaN	NaN	0.000000	10.000000	0.000000	0.000000	1.000000
0.000000	0.000000	0.000000	0.707289	0.141385	0.017861	10.000000	18075.000000	0.000000	0.000000
1.000000	93.521595	0.000000	0.691527	0.144608	0.015687	10.000000	18059.000000	18059.000000	1.000000
1.000000	86.356340	0.010696	0.694444	0.144000	0.019217	10.000000	18699.000000	18059.000000	0.000000
					...				

Table 2.1: Example of what is contained inside a Metadata file

In the list below are reported the definitions of the metadata's columns. The entry "quality" reports the goodness of a weld, which is what we want to estimate with machine learning. In this scenario, the feature "quality" is indeed extracted from the signal, but only when the entire welding session is over. There's no quality-evaluation in real-time, which is what this project aims to achieve. The earlier the systems detects from the signal a sign of a potential "bad quality", the faster maintenance can be achieved by re-calibrating the machine or cleaning up the optics of the laser.

- **Quality**: it is a flag which could be either *1.0*, if it refers to a signal labelled as *good/good w. over penetration/warning/warning w. over penetration*, or *0.0* if the label of the signal is set to *bad/bad w. over penetration*;
- **Constant penetration**: percentage of correct penetration over the whole weld;
- **Percentage porosity**: indicates the percentage of porosity, which takes place when 2 consecutive samples have a value greater than the 20% of the signal's average;
- **Energy signal**: total amount of energy required during the welding process;
- **Mean signal**: it is the average of all the values reported in a single signal file;
- **Deviance**: it is the standard deviation of all the values reported in a single signal file;
- **Type acquisition**: flag set automatically to value "10" when the welding process starts;
- **Length acquired signal**: it reports the total length of the signal, which consist of the whole number of samples inside a signal file;

- **Length analyzed signal**: it reports the length of the signal without considering the initial state and final state which have a magnitude around zero;

- **Reference**: it is a flag which could be either *1.0*, if the trial is used as a *reference* for its parameters to label accordingly the following signals, or *0.0* if the signal is not considered as a *reference*.

Reference

One of the essential features of a metadata file is the *reference* entry. In case a signal is flagged as *reference*, it will be used as a judgment parameter to label signals that refer to it. According to CRF, any signal that stands out from its *reference*'s parameters, it is labeled *bad*. A signal designated as a *reference*, which can be labeled either *good* or "good over penetration", will indicate the standard requirements its later referred signals should have to be labeled *good*. The *reference*'s standards are not explicit or defined; it will be investigated on what they are based on and if they are a proper parameter to recognize *bad* signals. Signals flagged as *reference* can be more than one during a production day. Those labeled as *good* or "good over penetration", and produced in one of the following two scenarios, are always flagged as *reference*:

- SCENARIO 1: When manual maintenance is performed, the signal recorded right after is flagged as *reference* and maintains this role for all the following signals of that welding session, until a new one is set;

- SCENARIO 2: When a new production session is started, the first signal is always a *reference*.

2.1.3 Dataset's directories

In Figure 2.5 it is shown the first level of the directories. The first folders are those whose names report the day their content was archived and included in the dataset. Each of these contain two sub-folders, one for the signals and one for the metadata (the inside is shown in figure 2.6). As it can be seen from the figure, supposedly correlated signals and metadata are grouped in the same backup folder. In the next section (2.2), it will be explained that this is not always straightforward. Some signal folders will have to be moved inside the dataset to maintain consistency in the relation between signals and metadata.

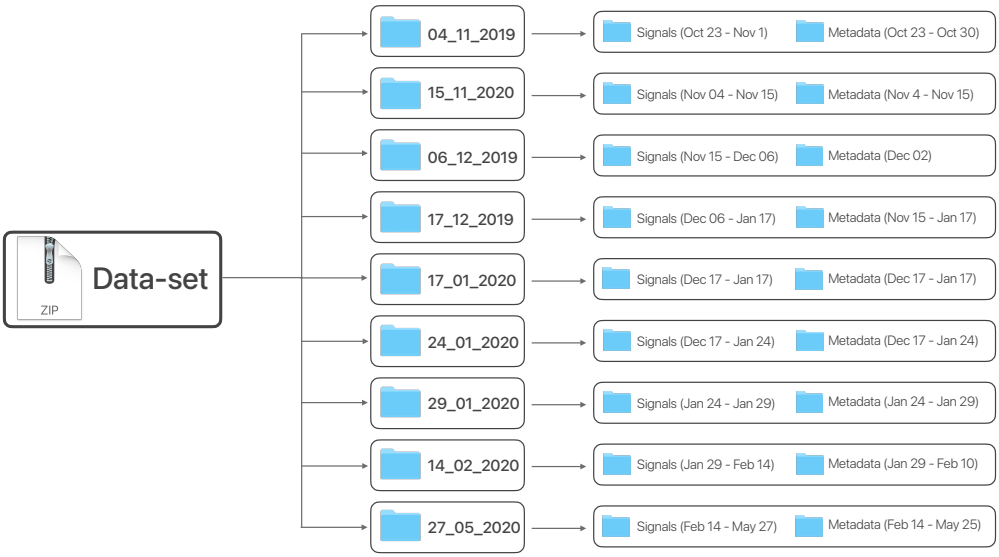


Figure 2.5: First level of folders' hierarchy

In figure 2.6 it is shown the second level of directories. On this level are located signal and metadata folders. Signal folders contain sub-folders named after their date of creation, their geometry code, and the velocity label. Inside of these are located signal files that need to be converted, as mentioned in the subsection [2.1.1]. All signal files have a defined pattern for their nomenclature: number + label of the weld quality output. Metadata folders contain sub-folders named after their creation date. These contain metadata files in the `.txt` format and are named with the geometry code and type of velocity of the signals they refer to. As it can be seen, there could be different metadata files with different geometry codes, and this is because on that same production day were welded different geometries.

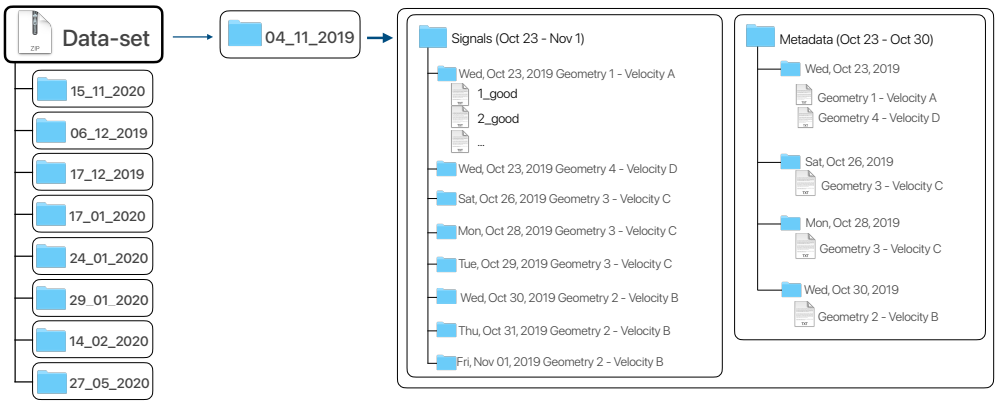


Figure 2.6: Second level of folders' hierarchy

2.2 Dataset Management

One of the most critical steps of the project was reorganizing the dataset to index signals with their relative metadata more easily. As explained in section 2.1.2, metadata tables contain pre-computed signals' features, and each table's row contains all the information referred to a specific signal file. To simplify the management of all the shared information between signals and metadata, a relational database was built with the intent of having a complete overview of files' associations.

2.2.1 Re-Organization

The methodology adopted for the reorganization consisted of moving the metadata files into the directories where their relative signals are located. This methodology's approach was purely manual and consisted of selecting the metadata files and moving them in their related signal files' directories. This moving operation of metadata files, has to be done with these three criteria in mind:

1. **Geomerty Code - Velocity**: metadata and the signals have to share the same geometry code - type of velocity;
2. **Metadata rows' length - Number of signals**: the metadata's table needs to have a number of rows perfectly equal to the number of signal files present in the folder where it is moved to;
3. **Metadata' creation date**: the creation date of the metadata file has to be subsequent to all the signal files' creation date.

The 2ND criteria is essential to alert when a match between signals and metadata is not available. We could have three scenarios while comparing metadata's number of rows with the number of related signals:

1. **Number of Metadata's rows > number of signals**: signal files are missing, and it is necessary to find them in other directories that still satisfy the 3 previous criteria. Sometimes, an entire signal folder or folders need to be retrieved or just a portion.
2. **Number of Metadata's rows < number of signals**: signal files are in excess, and in cases like this, only a portion of a signal folder is linked to a specific metadata file. The signal files which are not included, will certainly be required somewhere else.
3. **No signals available**: no signals are available in the entire dataset, most likely due to a backup problem.

In Figure 2.7 is represented how the directories' hierarchy changed from what is shown in figure 2.6. Now the metadata are located in the same folder of the signals.

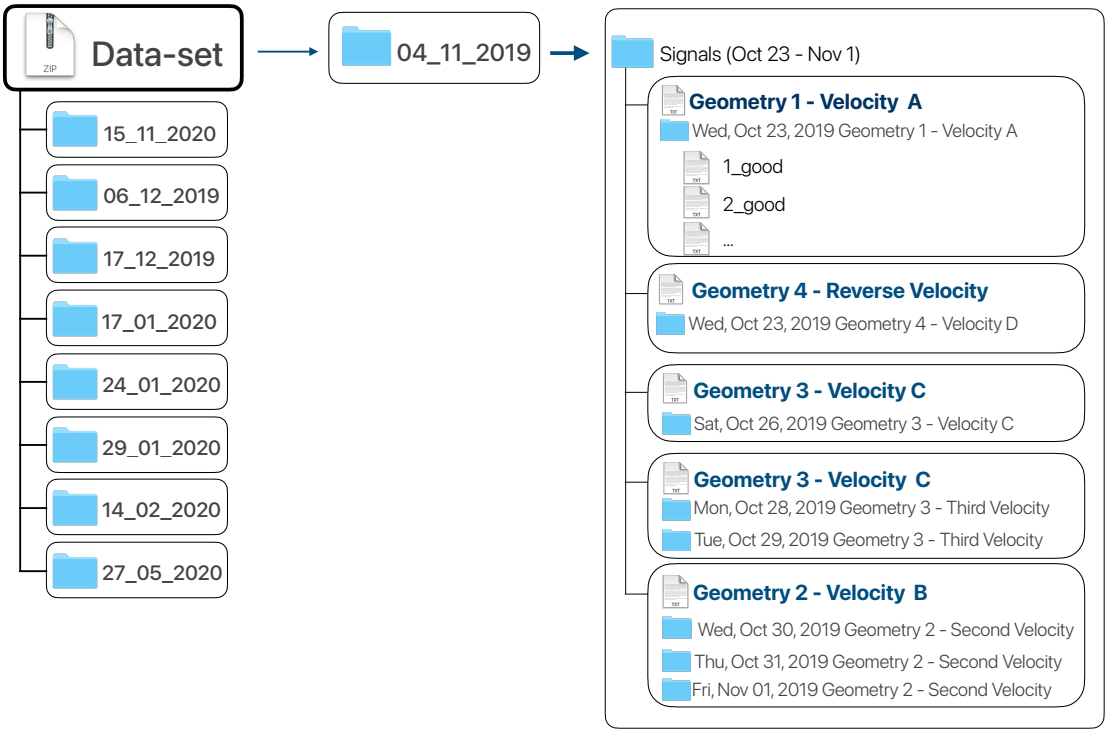


Figure 2.7: Folders structure after reorganization

2.2.2 Data-selection

Once the dataset had been re-organized, it came out that many signal files were missing, and the metadata-signal association was not possible. Most of the signals were missing in the backups from 04/11/2019 to 17/01/2020 (see figure 2.5 for reference). No signals were missing in the backups from 24/01/2020 to 27/05/2020. Therefore these folders were selected for the analyses. This selected portion of the dataset consists of 17973 signals and 39 metadata files with 17973 rows.

2.2.3 Relational Database

By moving the metadata into the directories of their referred signal, retrieving the features of the $n-th$ signal, is possible by reading its metadata $n-th$ row. To make the workflow more efficient, build a relational database for the signals. It consists of all the metadata files concatenated chronologically. New columns-features were added to this database, such as:

- **Name of the File**: It reports the nomenclature of the file to which the metadata's row refers to;
- **Path of the referred signal file**: It reports where the signal file is located, so that it is easier to retrieve it and read it;
- **Index of the signal's *reference***: It reports the index at which its *reference* is located;
- other more specific fields.

The relational database is fundamental for all the analyses that will be shown in the next chapter. The entire construction of the database, from concatenating the metadata files to adding new feature-columns, was done with the Python tool `Pandas`. Pandas is particularly useful to manage databases and to export them in different formats such as `.CSV`. Before performing our analyses, it was necessary to validate the relational database and make sure that the concatenation of the metadata files and new columns' addition did not invalidate the consistency of the information.

File Name	Type	Date Creation	Path	Index Reference	Quality	const_penetration	percentage_porosity	energy_signal	mean_signal	deviance	type_acquisition	length_acquired_signal	length_analyzed_signal	reference
1_good	good	2020-01-17 15:18:32	Files/24_01_2020/...	0	1.0	95.638629	0.025973000000000003	0.911094	0.109758	0.012864	10.0	19251.0	19251.0	1.0
2_good	good	2020-01-17 15:19:08	Files/24_01_2020/...	0	1.0	98.119122	0.020898	0.8799620000000001	0.113641	0.013096	10.0	19141.0	19251.0	0.0
3_good	good	2020-01-17 15:19:44	Files/24_01_2020/...	0	1.0	94.968553	0.015722999999999997	0.878403	0.113843	0.014537	10.0	19080.0	19251.0	0.0
4_good	good	2020-01-17 15:20:20	Files/24_01_2020/...	0	1.0	97.955975	0.167776	0.8884040000000001	0.112561	0.013999	10.0	19073.0	19251.0	0.0
5_good	good	2020-01-17 15:20:56	Files/24_01_2020/...	0	1.0	95.85327	0.010632	0.8917450000000001	0.11214	0.013806	10.0	18812.0	19251.0	0.0
6_good	good	2020-01-17 15:22:12	Files/24_01_2020/...	0	1.0	97.464542	0.037000000000000005	0.875856	0.114174	0.013354	10.0	18919.0	19251.0	0.0
7_good	good	2020-01-17 15:22:48	Files/24_01_2020/...	0	1.0	97.652582	0.031283	0.882653	0.113295	0.013431	10.0	19180.0	19251.0	0.0
8_good	good	2020-01-17 15:23:26	Files/24_01_2020/...	0	1.0	95.084485	0.040984	0.8780110000000001	0.113894	0.014753	10.0	19520.0	19251.0	0.0
9_good	good	2020-01-17 15:24:02	Files/24_01_2020/...	0	1.0	98.899371	0.015712	0.8830129999999999	0.113249	0.012522	10.0	19094.0	19251.0	0.0
10_good	good	2020-01-17 15:24:38	Files/24_01_2020/...	0	1.0	96.012759	0.037202	0.8684860000000001	0.115143	0.013816	10.0	18816.0	19251.0	0.0
11_good	good	2020-01-17 15:25:14	Files/24_01_2020/...	0	1.0	95.288754	0.045623	0.90501	0.110496	0.014772	10.0	19727.0	19251.0	0.0
12_good	good	2020-01-17 15:25:54	Files/24_01_2020/...	0	1.0	94.242424	0.08086499999999999	0.910605	0.109817	0.015059	10.0	19786.0	19251.0	0.0
13_good	good	2020-01-17 15:26:30	Files/24_01_2020/...	0	1.0	94.954128	51.0	0.8893399999999999	0.112443	0.014571	10.0	19608.0	19251.0	0.0
14_good	good	2020-01-17 15:27:06	Files/24_01_2020/...	0	1.0	99.03537	0.021439	0.863243	0.115842	0.012538	10.0	18658.0	19251.0	0.0
15_good	good	2020-01-17 15:27:42	Files/24_01_2020/...	0	1.0	94.054878	0.06604	0.8942700000000001	0.111823	0.015317	10.0	19685.0	19251.0	0.0
16_good	good	2020-01-17 15:28:18	Files/24_01_2020/...	0	1.0	98.726115	0.031842	0.878258	0.113862	0.012332	10.0	18843.0	19251.0	0.0
17_good	good	2020-01-17 15:28:54	Files/24_01_2020/...	0	1.0	98.548387	0.026882	0.887493	0.112677	0.01349	10.0	18600.0	19251.0	0.0

Table 2.2: Preview of the relational database

2.2.4 Validation: quality label

To validate the relational database it was compared the name of the files with the information related to them, such as the quality label.

The workflow consisted in:

1. Reading the signal's file name in the relational database and retrieving the quality label (for example for the file `23_good`, the quality label is *good*);
2. Reading the quality flag column in the relational database for that signal;
3. Checking if the label and the flag are coherent (i.e. if the signal is labeled as *good*, the correct flag is has to be equal to "1.0");

In Figure 2.8 are shown the results of this validation with a "line-scatter" plot. If the comparison between the signal's label and its quality entry returns a match, a value equal to 1 is plotted, otherwise 0. Among 17937 signal files, only 26 did not match the entry in the relational database, and they can be seen at the bottom of

the plot. The mismatches are present in only two folders, and in both folders, they refer to signals produced on the same production day for the same welding session. The mismatch is most likely due to an error in the original metadata file.

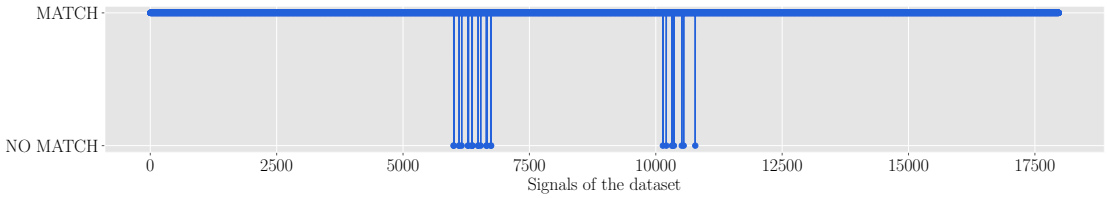


Figure 2.8: Result of the validation for the data organization

2.2.5 Data-cleaning

Once the validation of the relational database was completed, all the 26 errors were discarded. The cleaning operation consisted in simply removing from the DB the rows of the erroneous files, along with those rows containing "Not-a-Value" (NaN) entries. The cleaning process was carried out by importing the DB in Python as a Pandas Data-Frame, and then by applying a filter for discarding the just mentioned unwanted entries. After the cleaning the DB was saved and exported as a `.CSV` file for further utilization in the project.

Chapter 3

Working Pipeline and methodologies

This chapter will discuss the adopted methodologies to analyze the two file typologies of the dataset described in the previous chapter.

3.1 Methodologies for Metadata Analysis

The database is compounded of metadata files concatenated in order of creation's date. It holds all signal file information in one place, making analyses much more effortless. The relational database is a CSV file and is handled in a Python environment as a Pandas data-frame, which significantly simplifies the management and selection of data with required specifics. The following will describe how all the signals' features have been extracted from the DB and classified with Machine Learning algorithms. The process of classifying data is a standard activity in the field of machine learning, and it can be performed with the development of a model that, once properly trained, can analyze and segregate data into discrete values such as:

- True or False;
- pre-defined label classes.

A classifier to work correctly needs to be trained and tested; thus, selecting a portion of data from the dataset for training and one for testing is required. Usually, as shown in Figure 3.1, we reserve 80% of the dataset for training and the residual 20% for testing.

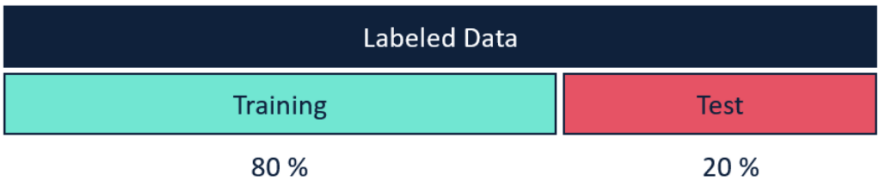


Figure 3.1: Division of the dataset for machine learning [16]

Classification methodologies are supervised learning, meaning that the correct data labels are handled to the model to verify its guesses and adjust parameters accordingly during the training phase. Once the model has been trained, the dataset for testing is employed to provide an unbiased evaluation of a final model fit on the training dataset [17]. Using just two sets for model training and testing makes the model not completely reliable, as training and test data do not always have the same variation as the original dataset. The K-Fold Cross-Validation solves this issue by dividing the input data into multiple groups, rather than just two (test and training). The letter "K" refers to how many groups the dataset of training and test will be split. In this way, the model will be trained and tested K times, using one fold as test data and the rest as training data for each iteration. This technique reduces over-fitting as most of the data are also used in the validation set. Another advantage of K-Fold Cross-Validation is that "it helps to generalize the machine learning model, resulting in better predictions on unknown data" [18].

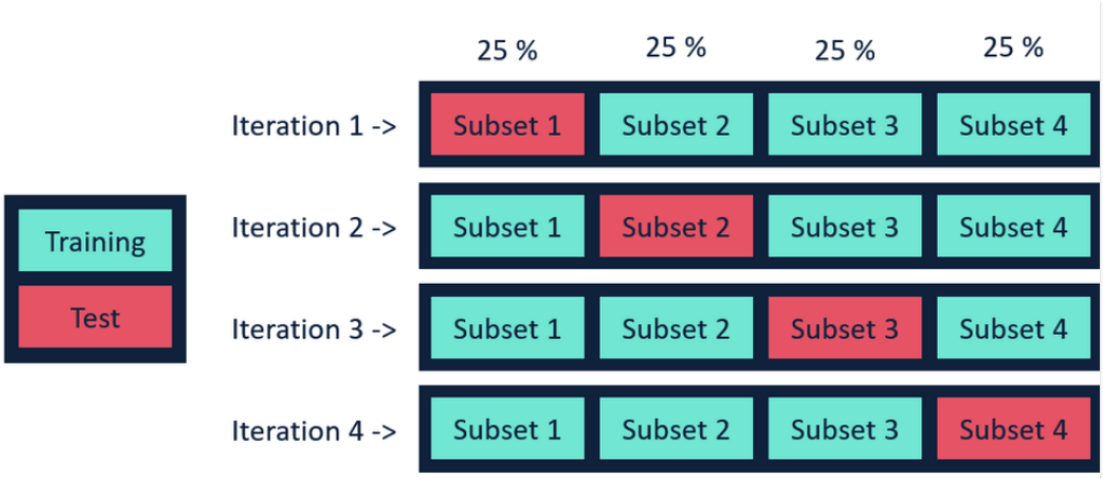


Figure 3.2: Example of a 4-Fold Cross-Validation [16]

The concept of classification is extensively used nowadays and some of the main applications are [19]:

- Spam detection: classification of e-mails with the goal of detecting spam

e-mails;

- Speech recognition: classification of speech words.
- Object recognition: classification of visual objects.

In our case study, the aim is to correctly label signals with two categories (*good* and *bad*) by analyzing their metadata collected in the DB. The considered features for classifying signals are their correct label and statistics (mean, standard deviation, and more). Only `C Velocity` is considered for the classification with metadata: given its larger number of signals labeled as *bad*, it allows to fit the classifier more precisely.

3.1.1 Decision Tree classification (linear)

Decision Tree is a tool used to represent classification and regression. It is based on conditional control statements, and on sequential decisions that will help assigning a label/class to input data [20]. A sequence of decisions will lead to the tree's leafs which hold the label/class to assign. In Figure 3.3 is shown an example of a Decision Tree.

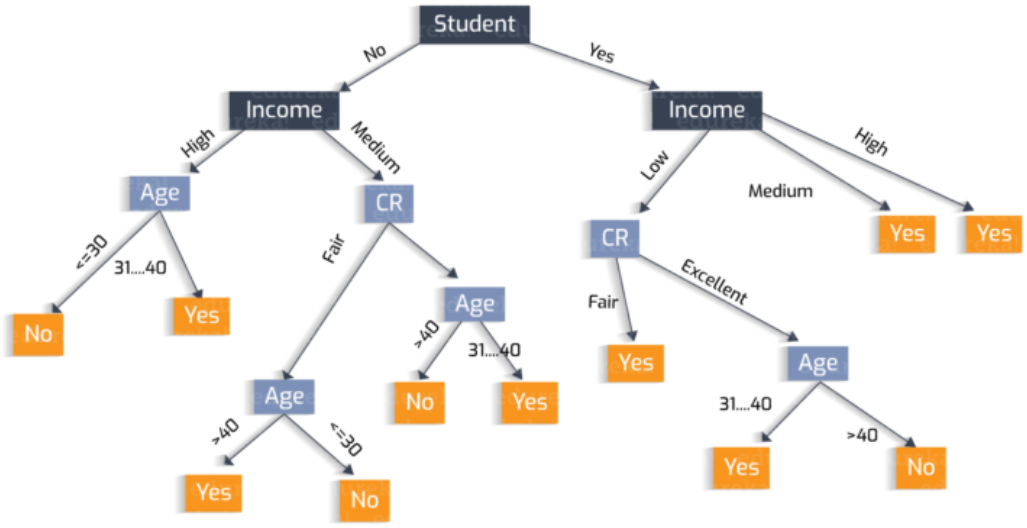


Figure 3.3: Example of a Decision Tree [21]

For this classification solely the `C Velocity` signals were taken into account. These are the considered features, taken from the relational database, to build the classifier's model:

- Constant Penetration;
- Percentage Porosity;
- Energy of the signal;
- Mean of the signal;
- Deviance of the signal.

The model's accuracy is evaluated by comparing the classifier's output with the correct signals' label reported in the *Quality* entry of the relational database. For the subgroup of the training data, the database of signals flagged as a *reference* has been selected. On the other hand, for the subgroup of the test data, have been selected all the `C Velocity` signals included in the relational database. As mentioned in the introduction, using just two sub-sets affects the model's accuracy, thus K-Fold Cross-Validation was included in the process, considering $K = 10$. Cross-Validation can be executed with different scoring techniques, meaning that when assigning a label we can push to maximize the overall accuracy or the precision. The main scoring techniques are:

- **Recall Score**: ratio of positive instances correctly detected by the classifier;
  $$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives+False Negatives}}$$
- **F1 Score**: is the harmonic mean of precision and recall;
  $$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$
  This is how precision is calculated $\text{Precision} = \frac{\text{True Positives}}{\text{True Positives+False Positives}}$
- **Accuracy Score**: accuracy of the positive prediction
  $$\text{Accuracy} = \frac{\text{True Positives+True Negatives}}{\text{True Positives+True Negatives+False Positives+False Negatives}}$$

True Positives and True Negatives are respectively the *good* and *bad* signals correctly labeled by the classifier, while False Positives and False Negatives are the *good* and *bad* signals classified incorrectly.

When classifying metadata, all three scoring techniques will be employed. Moreover, it will be evaluated the precision of each classifier employed during the tests. Decision Tree is a linear classifier, which means that the features shown in 3.1.1 are combined linearly.

3.1.2 Support Vector Machine (SVM) classification (non linear)

An additional methodology to classify data is Support-Vector-Machine (SVM). Like Decision Tree, SVM is a supervised linear classifier with the difference that it can also handle non-linear models with the aid of kernels.

This classifier uses a hyper-plane or set of hyper-planes to separate data points transferred on a N-dimensional plane. Hyper-planes are $(N-1)$-dimensional sub spaces deployed to separate data in a ND space. In Figure 3.4 it is shown an example of how an hyper-plane depends on the dimension of the space; it is a line in a 2D space or a plane in a 3D space.

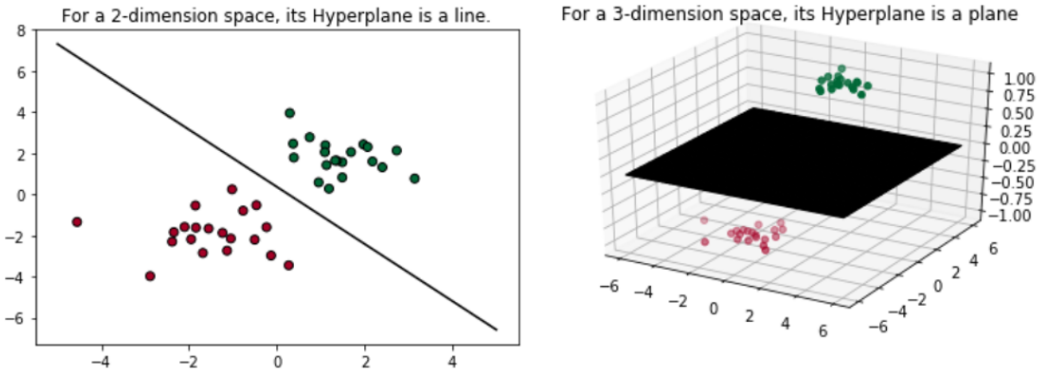


Figure 3.4: Example of a different hyper-planes [22]

An important consideration to take into account is how an hyper-plane is chosen among others. The criteria to chose the best hyper-plane, is to select the one that divides groups of data with the maximum distance, or margin. An example is now proposed to show what actually makes an hyper-plane better than another. In Figure 3.5 are shown training data points which can be either black or white, and 3 hyper-planes (H1,H2 and H3). Data points that have the same color belong to the same class and the classification/division with each hyper-plane is evaluated. H1 does not separate the points, while H2 and H3 succeed but with different results. The best division is the one of hyper-plane H3, since it has "the largest separation, or margin, between the two classes" [23]. A good separation is achieved when the margin between the hyper-plane and the nearest training data points of any class is the largest possible. "The bigger is the distance between the hyper-plane and the data, the lower the generalization error of the classifier" [24].

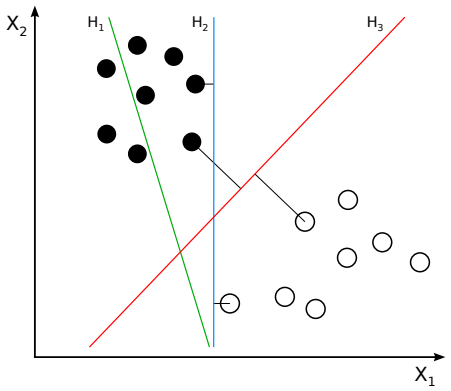


Figure 3.5: Example of a linear SVM [23]

The just showed example considers the case of a linearly separable dataset. Data points that are not separable with a line or a plane, require a transformation. The idea is to map data into a higher-dimensional space to make separation easier (see Figure [**3.6**] for an example).

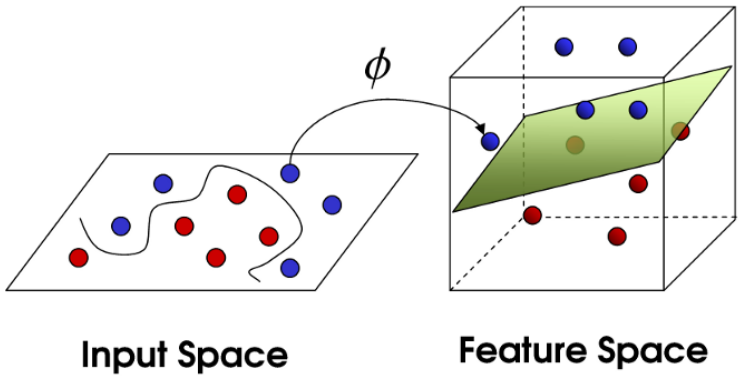


Figure 3.6: Mapping of data to a higher-dimensional space [25]

Kernel functions are employed in SVM to take data as input and transform it into the required form in a higher-dimensional space. The kernels commonly adopted in the SVM for non-linear classifications are:

- Radial Basis Function (RBF) - kernel for non linear models
- Polynomial - kernel for non liner models

In Figure 3.7 are depicted examples of how kernels allow SVM to separate data:

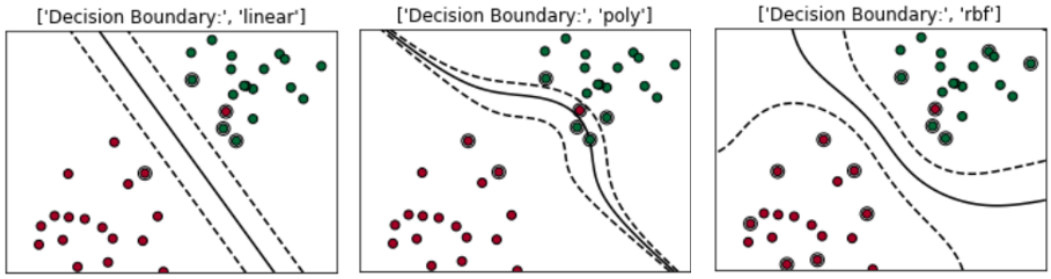


Figure 3.7: Example of SVM kernels [23]

For SVM's classifications, it was only utilized the `C Velocity`. It was performed a linear classification with the "linear" kernel, and a non-linear one with the RBF and Polynomial kernel. These are the considered features, taken from the relational database, to build the classifier's model:

- Constant Penetration;
- Percentage Porosity;
- Energy of the signal;

The model's accuracy is evaluated by comparing the classifier's output with the correct signals' label reported in the *Quality* entry of the relational database. For the subgroup of the training data, the database of signals flagged as a *reference* has been selected. On the other hand, for the subgroup of the test data, have been selected all the `C Velocity` signals included in the relational database. As mentioned in the introduction, using just two sub-sets affects the model's accuracy, thus K-Fold Cross-Validation was included in the process, considering $K = 10$.

3.2 Methodologies for Signal Analysis

This section will present the methodologies adopted for signal file analysis. The following working pipeline is changed from the one of the previous section since the type of file is entirely different. Previously classification was based on the signal's features; now, the focus shifts to signals' time trends and how they change from the *reference*. The first analyses consisted of examining the time-series in the frequency domain and later in the time domain, performing feature extraction based on the technique of *windowing*.

3.2.1 Frequency-domain analysis

In signal processing, frequency-domain analysis is a powerful tool, and it is extensively used in communications, image processing, and many other fields. The time-based analysis examines how a signal transforms over time, while frequency-domain analysis shows how a signal's energy is disseminated across a range of frequencies. The Fourier Transform (FT) is a mathematical tool that allows the transformation of a signal from a time-based domain to the frequency one. The vice versa is possible with the Inverse Fourier Transform (IFT). When the FT tool is employed, the input signal is decomposed into the sum of N-sine wave frequency components, each one with a certain magnitude (see Figure 3.8). Time-series, once converted with the FT in the frequency domain, can be represented with the spectrum of their frequency components.

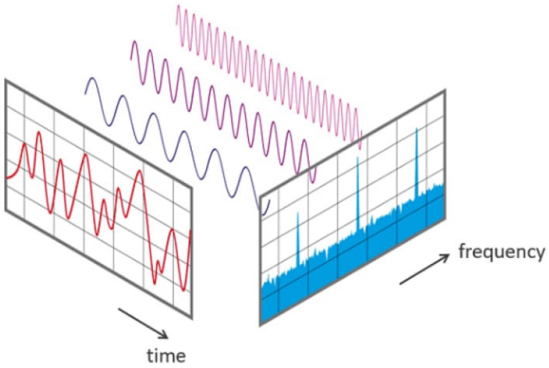


Figure 3.8: Example of a transformation from time-domain to frequency-domain [26]

The Fourier transform is used for continuous signals, for sampled ones it is used the Discrete Fourier Transform (DTF) defined as follows:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \quad k = 0, \ldots, N-1 \tag{3.1}$$

The DTF of a discrete signal is the the multiplication of the signal sample $x[n]$ with the Nth primitive root, which is $e^{-i2\pi kn/N}$. N is the total number of the signal's samples. The final result is the sum of all multiplications for any n.

The DTF has a complexity equal to $O(N^2)$ because there are N(multiplications) $\times$ N(additions). This operation would have a huge complexity if applied to signals with many samples, therefore it is used the Fast Fourier Transform (FFT). FFT's algorithm determines DTF for an input greatly faster than computing it directly, lowering the computation complexity for a problem of size N from $O(N^2)$ to $O\left(N \log_2 N\right)$ [27].

For our analyses, signals were converted from the time-domain to the frequency-domain with the FFT and finally visualized with a plot of their spectrum.

3.2.2 Time-domain analysis

After analyzing signals in the frequency domain, they have been examined in the time-domain; they were not subject to any prior transformation since they were already in time-format. As it was explained in the previous chapter, signals have three sections, and only the intermediate one is interesting to be subject to analysis (see Figure 2.2). The general idea applied to all following time-domain analyses is *feature extraction* and signal comparison based on windows. The concept of "*windowing* consists of dividing signal's intermediate part into windows of a fixed size (in Figure 3.9 the chosen dimension is 256 samples per window) and proceed with the analysis focusing on single windows and not on the whole signal. The following analyses evaluate how the features extracted from every window of a signal diverge from those of their *reference*'s windows; different windows size, according to the analysis' requirements, will be tried.

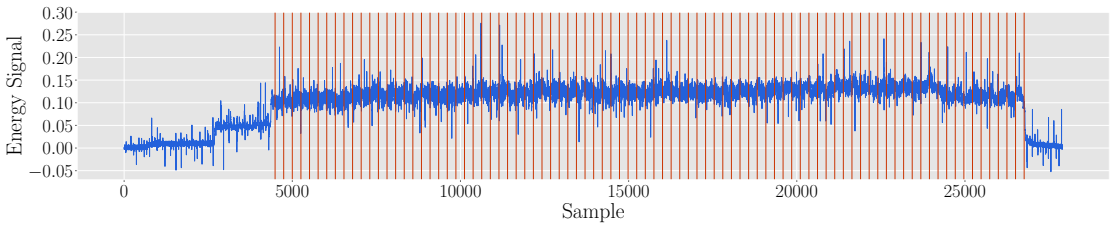


Figure 3.9: *reference* signal divided into windows of 256 samples each]

Looking at the plot in Figure 3.9, it is evident how the signal has a high dynamic and a noticeable noise. These factors could affect the quality of the analyses, especially when signals' trends have to be compared with their *reference*s. In order to diminish the signals' dynamic and their noise, a low-pass filter was applied. A low-pass filter (LPF) attenuates all signals' frequencies above the cutoff frequency and includes those that are below it [28]. Usually, the cutoff frequency for an LPF is chosen as the frequency where the signal's magnitude response is 3 dB lower than the value at 0 Hz [27]. In our case, it was chosen manually, without evaluating the 3 dB drop, according to how much signal dynamic it was required for the analyses. In Figure 3.10 is shown an example of a *reference* signal filtered and divided into windows of 256 samples each.

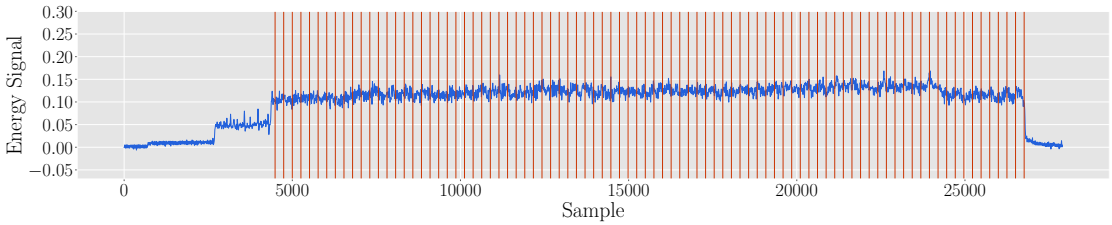


Figure 3.10: *reference* low-pass filtered signal divided into windows of 256 samples each]

In the following are explained what type of features are extracted from each signal's window and how these were compared to *reference*s.

Count of windows in band of quality

This first analysis on *window feature extraction* consisted of retrieving each signal's window's mean, evaluating if that result stays inside a range called *quality band*, and counting how many windows are out of that range. The *quality band* is a concept used to effectively compare signal's windows with its *reference*. Once we select a signal and retrieve its *reference*, the *quality band* is obtained as follows:

$$\mu = \frac{\sum_{n=start}^{end} x[n]}{N_{intermediate}} \tag{3.2}$$

$$QualityBand = \mu \pm \alpha(STD(X_{intermediate})) \tag{3.3}$$

The equation 3.2 shows how the mean for the intermediate section of the *reference* signal was computed. The value $x[n]$ represents the *reference* signal while *start* and *end* symbolize where its intermediate part begins and finishes. $N_{intermediate}$ is the value that represents how many samples are inside the intermediate part. The equation 3.3 is the one that represents how the quality band was finally obtained. Once the mean of the *reference* signal is computed (equation 3.2) the result is used to evaluate how large the band is going to be. $X_{intermediate}$ is the middle part of the *reference* signal and its standard deviation (STD) represents the upper and lower bound of the quality band. The result of the STD is reduced of a value equal to α (usually set equal to 50 - 60% of the signal's STD). As you can see from Figure 3.11 a signal labeled as good is divided in windows of 128 samples each, and for every window it was computed the mean of the signal inside of it. The idea is to compute in how many windows the mean signal (the plot in black) lies in the quality band (the blue bar). Whenever the mean is inside a window, it is green, otherwise it is red.

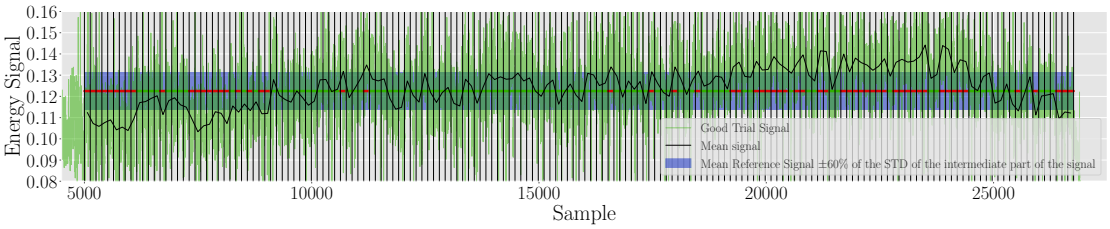


Figure 3.11: *good* signal divided into windows of 128 samples along with windows' analysis

Distance computation from adaptive quality band

The previous analysis was limited to counting how many times the window's mean was inside or not the quality band. This methodology is different because instead of counting how many times the mean is in the quality band, it computes how far it is. Another difference from the previous methodology, is that now the quality band is no more constant. The way the quality band is computed consists of:

1. Dividing the intermediate section of *reference*'s signal into windows (for example each window contains 128 samples);
2. Selected the kth window, computing the mean that window:

$$\mu_k = \sum_{n=0}^{128} window_k[n]$$

3. For the given window, compute the upper and lower bound in this way:

$$UP_k = \mu_k + \alpha(STD(window_k))$$
$$DOWN_k = \mu_k - \alpha(STD(window_k))$$

The value α is a scalar number to fraction the amount of STD and $window_k$ includes all the values of the signal in a given window k. In Figure 3.12 is reported an example of a *reference* signal and of the quality band linked to it.

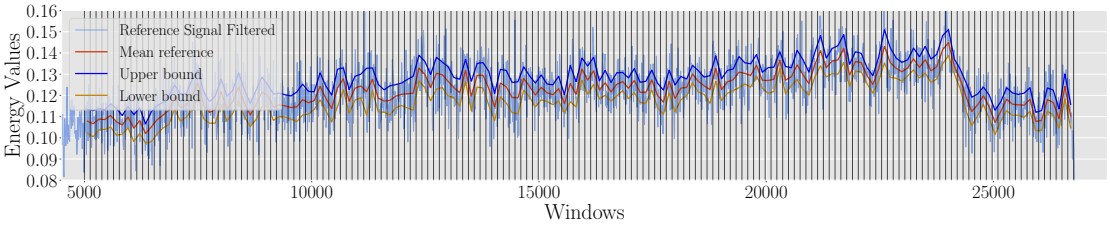


Figure 3.12: Visualization of *reference*'s adapted quality band in windows of 128 samples

Figure 3.13 shows the final application of the adaptive band to a referred signal. The signal is still divided into windows (in this example each window is of 128 samples) and the black plot is the mean for each signal's window.

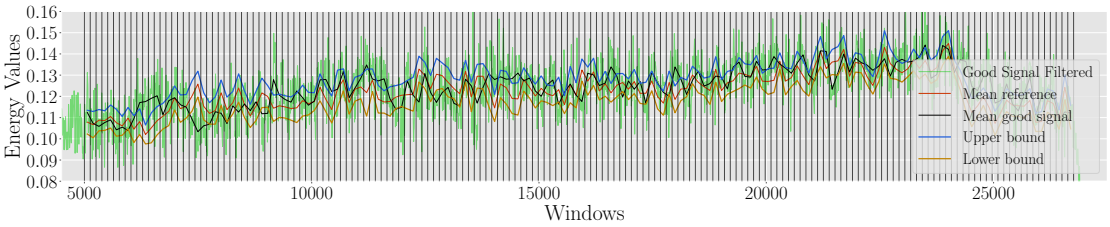


Figure 3.13: Example of a *reference*'s mean adapted to windows of 128 samples

This methodology, once the adaptive quality band is created and the signal's windows' mean is retrieved, calculates the distance between the mean of the signal and the quality band, for each one of the signal's windows. The distance, in order to keep the values normalized between 1.5 and -1.5, is computed as follows:

- If the mean of the signal is over the adaptive quality band:

$$distanceWindow_k = 1 + \left(\frac{\mu_k - UP_k}{meansignal_k} \right)$$

- If the mean of the signal is below the adaptive quality band:

$$distanceWindow_k = 1 + \left(\frac{\mu_k + DOWN_k}{meansignal_k} \right)$$

Computing the distances for all the signals of a production day is a lot of information, and their representation with a heat-map (see Figure 3.14) allows a global overview of how signals behave and differ from their *reference* over time. In the heat-map of Figure 3.14 is shown, on the X-axis, all the signals' names from a picked production day, while on the Y-axis, are shown all the windows in which the signals are split. The heat-map's color bar shows, distinguishing *good* and *bad* signals with different colors, how distant is the kth signal's window from the kth quality band's section. The gradient of the colors represents if the kth signal's window is above/under/near or in the quality band.

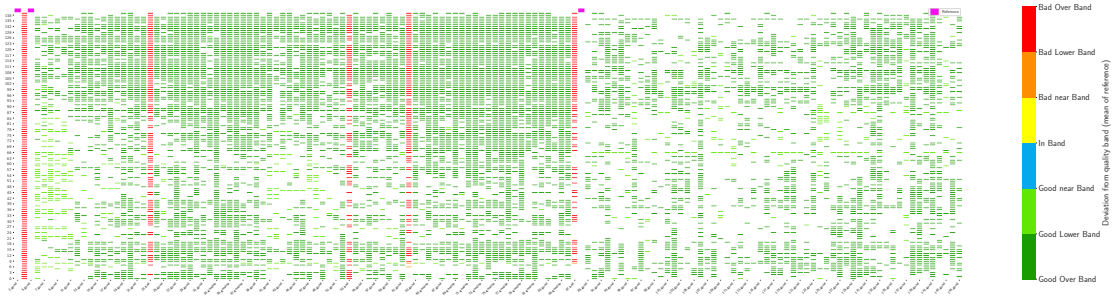


Figure 3.14: Heatmap of the distances

Euclidean distance between statistics arrays

The windowing approach is employed again in this analysis. Here are computed different statistics for all signals' windows, then the euclidean distance from their *reference*s is determined. By considering all the signals on a production day I computed, per each signal's window an array of these statistics:

1. 99 Percentile of the signal's window;
2. 1 Percentile of the signal's window;
3. Mean of the signal's window;
4. Standard Deviation of the signal's window;
5. Max Value of the signal's window;
6. Min Value of the signal's window.

As a preliminary evaluation, each statistic's values were visualized to determine if they needed any normalization. In figure 3.15 are shown all the values of a signal's windows. On the x-axis, there are the windows' indexes, and on the y-axis are present the results for each statistic. The standard deviation is close to zero for all windows, and it was excluded. This choice is because its contribution would be null in the distance's computation, and including it would have added complexity to the algorithm.

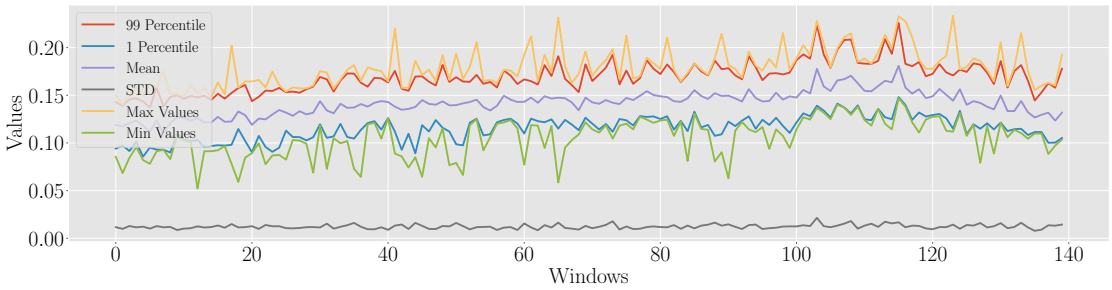


Figure 3.15: Results of euclidean distance of statistics' arrays for one signal

Below are reported two array of statistics, one for a *reference* (the one on the left) and one for the signal referred (the one on the right).

$$\begin{bmatrix} \text{99 Percentile} \\ \text{1 Percentile} \\ \text{Min value} \\ \text{Max value} \\ \text{Mean value} \end{bmatrix}_{Signal_s Win_w} \xrightarrow{E.Distance} \begin{bmatrix} \text{99 Percentile} \\ \text{1 Percentile} \\ \text{Min value} \\ \text{Max value} \\ \text{Mean value} \end{bmatrix}_{Reference_r Win_w}$$

$$s = 0,1 \dots S \qquad w = 0,1 \dots W \qquad r = 0,1 \dots R$$

These are the description of all the variables:

- The variable s is used to specify which signal of the production day we are considering;
- The variable w is used to specify which signal's window of the production day we are considering;
- The variable r is used to specify which *reference* of the production day we are considering;

When two arrays are obtained, one for the sTh signal and one for its rTh *reference* and both for the wTh window, the euclidean distance is computed.

Linear Regression

This methodology is no more based neither on *reference* signals or the *windowing* technique. The idea is to compute a **linear regression** of each one of the signal's intermediate part and compare signals' results with each other. Linear regression returns three parameters (slope, intercept and R^2 see next paragraph) which are examined to evaluate if they are strong enough to distinguish a *bad* signal from a *good* one. In Data Science, Linear Regression (LR) is one of the most used techniques to generate a linear approach for modeling the relationship between a scalar response and one or more explanatory variables [29]. Considering a dataset made of scalar response variables (dependent variables $\mathbf{y}$) and explanatory variables (independent variables $\mathbf{x}$), the aim is to find a linear function (a non-vertical straight line) that, as accurately as possible, predicts the dependent variable values as a function of the independent variable[30]. In Figure 3.16 is pictured an example of Linear Regression. The linear model (the blue line) fits two-dimensional data points scattered onto the Cartesian plane.

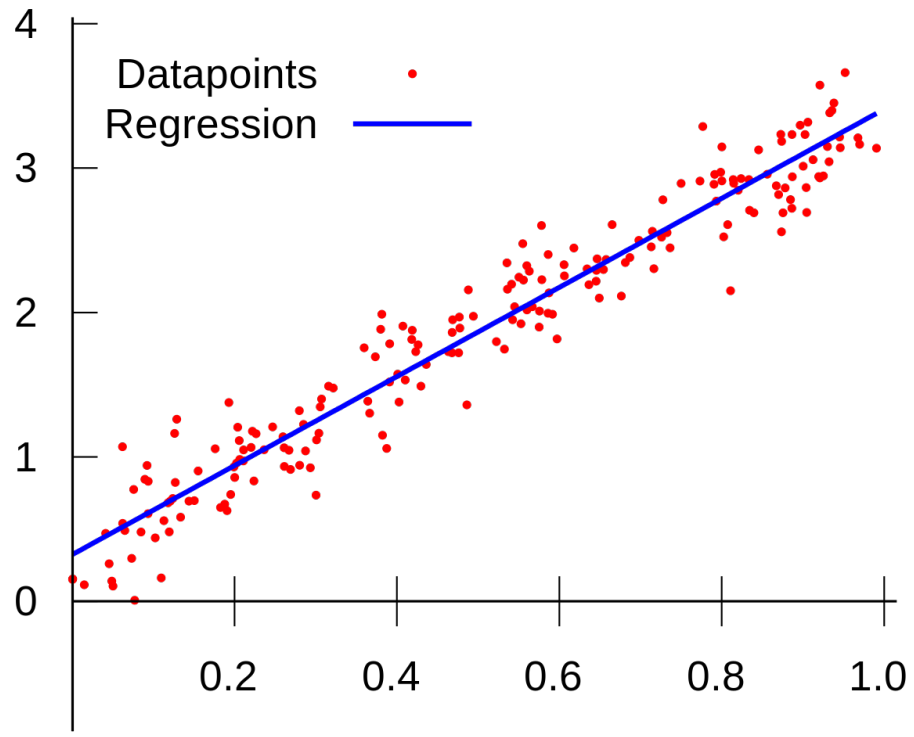


Figure 3.16: Example of Regression [31]

When performing LR, two parameters are weighed to fit the model to the observed data (green points in Figure 3.17): Given a line equation $f(x) = b_0 + b_1 x$ (black line in Figure 3.17) the objective is to find optimal values of b_0 and b_1 to minimize the the residual distances and obtain the optimal regression line. These are the line's coefficients:

- b_0 - **intercept**: represents the point where the line intersects the **y** axis, it;
- b_1 - **slope**: represents the inclination of the regression line.

Residual distance between observed data (y_i) and the prediction of the linear model ($f(x_i)$) is represented in the figure below as the gray dashed lines and mathematically defined as:

$$y_i - f(\mathbf{x}_1) = y_i - b_0 - b_1 x_1 \text{ for } i = 1, \ldots, N$$

An optimal linear regression implies minimizing the overall gap between the green points and the red squares, and it is denoted as Sum of Squared Residuals (SSR):

$$SSR = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

The smaller is the SSR result, the better is the model's fitting.

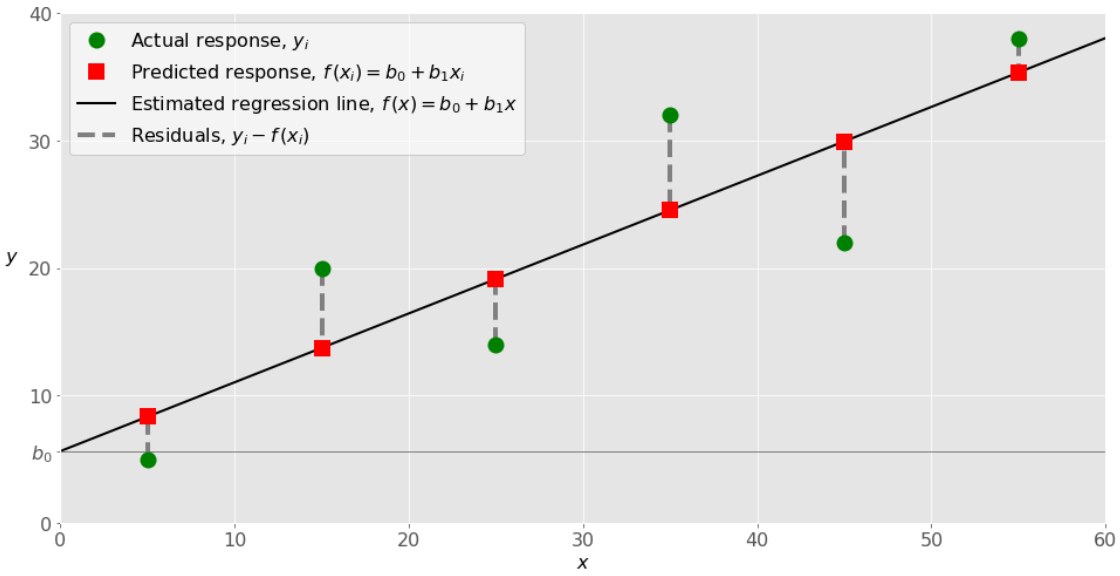


Figure 3.17: Example of Regression [32]

The performance of a linear model is explained by the indicator R^2, called the **coefficient of determination**. It is the square of the correlation between

predicted values of the linear model $f(x_i)$ and observed data (y_i); it ranges from 0 to 1. $R^2 = 0$ means that the dependent variable cannot be predicted from the independent variable, while $R^2 = 1$ means that the dependent variable can be predicted without error from the independent variable [33].

$$R^2 = \left[\left(\frac{1}{N} \right) \sum_{i=1}^{N} \frac{(x_i - \hat{x})(y_i - \hat{y})}{\sigma_x * \sigma_y} \right]^2 \tag{3.4}$$

N is the number of observations used to fit the model, x_i is the x value for the i-th observation, $\hat{x}$ is the mean x value, y_i is the y value for the i-th observation, $\hat{y}$ is the mean y value, σ_x is the standard deviation of x, and σ_y is the standard deviation of y [33].

Chapter 4

Results

4.1 Metadata analysis results

Here I show the results of the analyses performed on *metadata*, more precisely the outcomes of linear and non-linear classification using only signals' features. For both classification methodologies, only `Velocity C` signals have been employed.

4.1.1 Classification with Decision Tree

Decision Tree is utilized as a linear classifier for *metadata*, which include the following entries:

- Constant Penetration;
- Percentage Porosity;
- Energy of the signal;
- Mean of the signal;
- Standard deviation of the signal.

The model's training has been done using only *bad* and *reference*'s metadata, while the testing phase was performed on the whole dataset of `Velocity C` signals. As anticipated in the previous chapter 3.1.1, I opted for a 10-Fold Cross Validation, to reduce over-fitting, and different scoring techniques. In the following I present the confusion matrices and results of accuracy and precision for each type of scoring technique.

DT classification - recall score - *bad* label

Table 4.1 shows that 85% of *bad* signals have been correctly identified by the trained DT model, while 40% of *good* signals have been incorrectly classified as belonging to the counterpart label. The reason why the result is better for the *bad* label is due to the *recall* scoring technique that tries to classify as much as possible to a pre-defined class, which in this case is the *bad* one. The model's accuracy is 0.6, which is a good result (it can range between 0 and 1). This is due to how unbalanced data quantities are (73 *bad* signals versus 5770 *good* signals versus). Obtaining a high accuracy with an unbalanced dataset is a phenomenon, in the field of data science, known as "*Accuracy paradox*". The majority label represents "usual" and the minority label represents "unusual," such as a fault or a fraud. Having a good performance on the minority class will be preferred over a good performance on both classes, leading to an increased accuracy [34]. Precision for the *bad* label is very low, meaning that the model is unable to classify *bad* signals.

		ASSIGNED LABEL	
		Bad	**Good**
TRUE LABEL	**Bad**	61	12
	Good	2292	3408
ACCURACY	0.60 (Model)		
PRECISION	**Bad**	0.03	
	Good	1.00	

Table 4.1: Confusion matrix with maximization of the recall score on *bad* class

DT classification - F1 score - *bad* label

By using the F1 scoring technique on the *bad* label, results do not improve since the accuracy drops below 0.5 and the *bad* label's precision remains low, which means that this classifier is not able to correctly distinguish *bad* signals when tested.

		ASSIGNED LABEL	
		Bad	Good
TRUE LABEL	Bad	68	5
	Good	3013	2687
ACCURACY	0.48 (Model)		
PRECISION	Bad	0.02	
	Good	1.00	

Table 4.2: Confusion matrix with maximization of the F1 scoring *bad* class

DT classification - accuracy score - both labels

When using the *accuracy* scoring technique to increase both labels' accuracy, the results are exactly the same of the previous classifier with *F1* scoring technique.

		ASSIGNED LABEL	
		Bad	Good
TRUE LABEL	Bad	68	5
	Good	3013	2687
ACCURACY	0.48 (Model)		
PRECISION	Bad	0.02	
	Good	1.00	

Table 4.3: Confusion matrix with maximization accuracy of both classes

4.1.2 Classification with Support Vector Machine

SVM is a non-linear classifier employed to classify signals considering the following *metadata* entries:

- Constant Penetration;
- Percentage Porosity;
- Energy of the signal;
- Mean of the signal;

The model's training has been done using only *bad* and *reference*'s metadata, while the testing phase was performed on the whole dataset of `Velocity C` signals. For

building this classifier a mapping function is chosen among those listed in the previous chpater 3.1.2 and, moreover, I still opted for a 10-Fold Cross Validation, to reduce over-fitting. The choice of ideal parameters, including the selection of the best kernel, has been performed with an additional tool to manage such demanding operation: Apache Spark. As described in [35], Spark is an open source platform for large-scale data analysis processing, needed to speed up operations. It does not include a data management system and is therefore usually deployed on other storage platforms.Its most prominent feature is its in-memory cluster computing which is responsible for increasing the data processing speed. A cluster simply represents a group of computers connected (called nodes) and coordinated with each other to process and analyze data. Utilizing Spark allowed to retrieve faster the parameters of the SVM model, which turned out to be:

- Kernel: RBF
- C: 26.366508987303554 (Regularization parameter for the chosen kernel)
- Random State: 2 (Pseudo random number to shuffle data during probability estimates)
- Gamma: 1.0 (Kernel coefficient)
- Class Weight: None (Both labels have the same importance) [36]

The model, using these parameters in the training phase, returned a precision equal to 0.88. This means that the model is able to precisely distinguish *bad* signals from *references*. Now is time to test this this model with all the other *Velocity C* signals and see if the classifier's performance are the same.

In table 4.4 there are the results of the testing phase. The accuracy of the model is very high (due to *Accuracy Paradox*) while the precision for the *bad* label is low. This means that is impossible to precisely distinguish and classify *bad* signals among those of `Velocity C`.

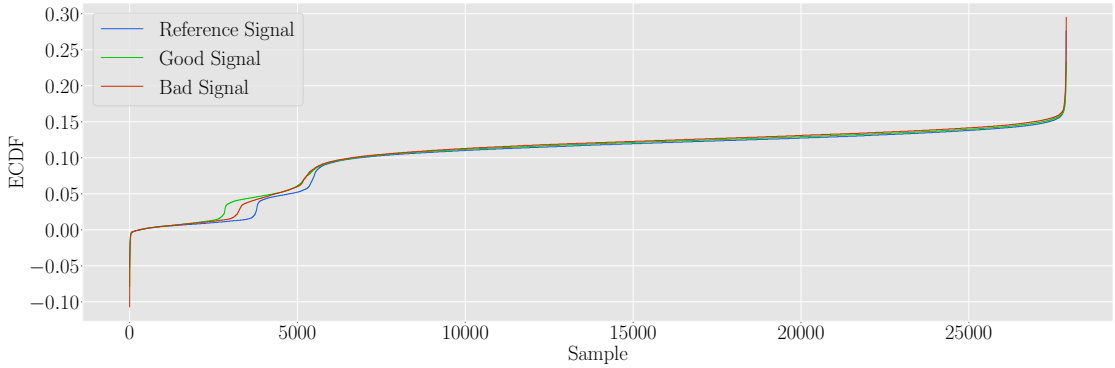
		ASSIGNED LABEL	
		Bad	**Good**
TRUE LABEL	**Bad**	49	24
	Good	1403	4297
ACCURACY	0.75 (Model)		
PRECISION	**Bad**	0.03	
	Good	0.99	

Table 4.4: Confusion matrix of the test results with SVM model

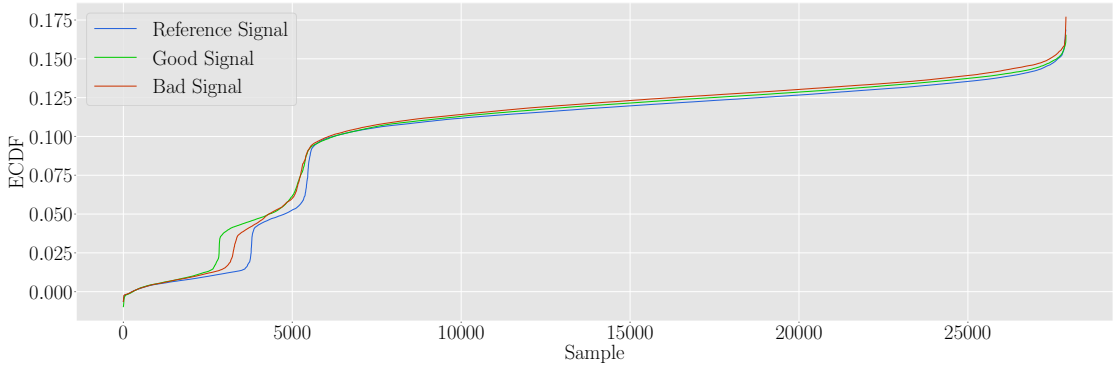
These classification results show that using both linear and non-linear classifier with the chosen metadata, is not enough to label signals with precision.

4.2 Signal analysis results

In this section I present all the results obtained from the analyses of the signals' timeseries. In section 4.1 the focus was on testing classification using signal's features; now the study is on the time series' trends and the relationship between signals and their *references*. An initial test consisted of plotting the Empirical Cumulative Distribution of signals and study if their distribution differed based on the label assigned. Figure 4.1 depicts the ECDF of a *reference* signal, along with the a *good* and *bad* signal referred to it. The first plot 4.1a shows that signals have almost the same distribution with no substantial difference between the *good* signal and the *bad* signal. The second plot 4.1b represents the distributions of the same signals but with a low-pass filtered applied. As mentioned in 3.2.2, the filtering was applied to reduce the signal's noise and dynamic. From what is shown in the plot 4.1b, signals have similar distributions. In this case filtering doesn't help in deriving potential differences among signal classes.



(a) ECDFs of signals not filtered



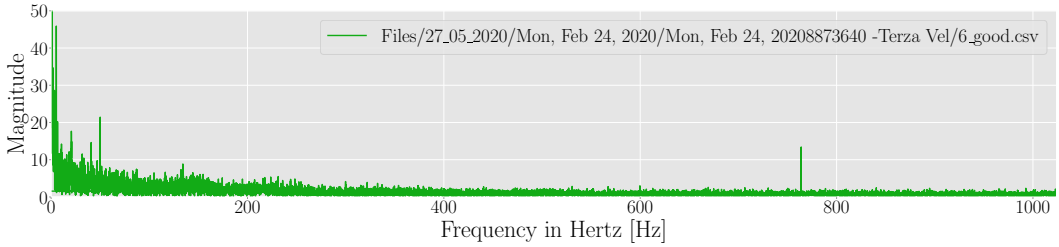
(b) ECDFs of filtered signals with low-pass filter with the cutoff frequency at 101Hz

Figure 4.1: ECDFs of test signals

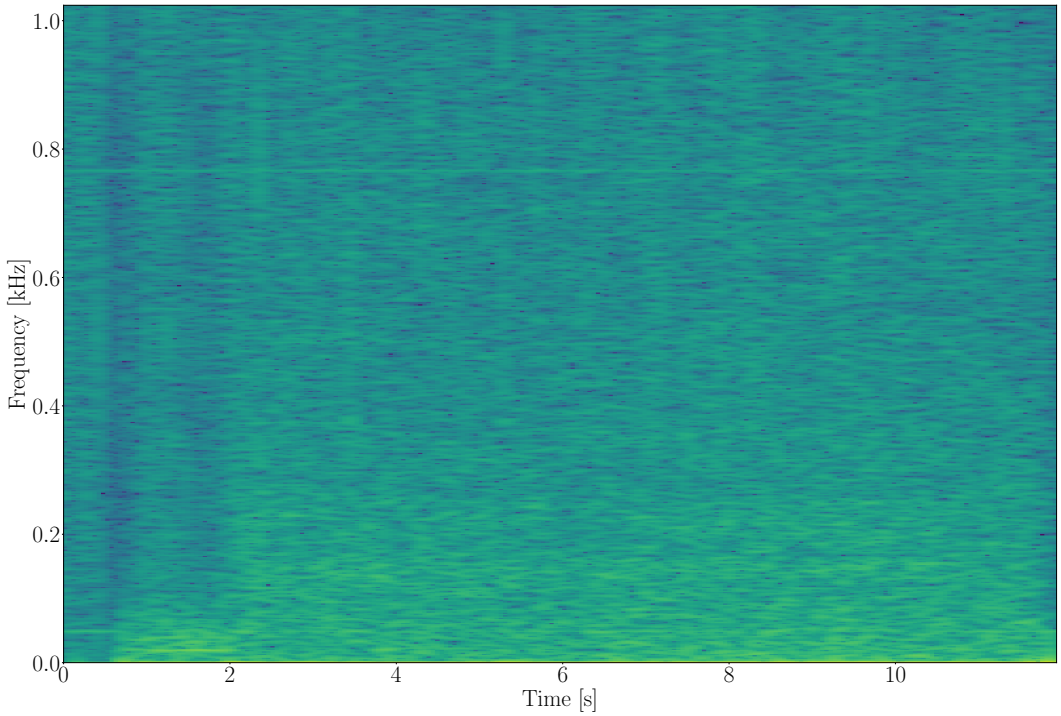
4.2.1 Frequency-Domain analysis

In figure 4.2 are depicted the analysis in the frequency domain for a signal labeled as *good*. The first plot 4.2a shows the frequencies on the x-axis and their magnitude on the y-axis. The frequencies that stand out from the others are 50[Hz] and 780[Hz]. The first one, rather than being descriptive of the signal, is more likely linked to the recording machine's electric current, which works at 50[Hz] frequency.

In the spectrogram of the same signal 4.2b is shown that over time these two frequencies have a more considerable intensity to the others; in fact, on the y-axis at 50 and 780 [Hz] is visible a yellow line that over time remains visible.



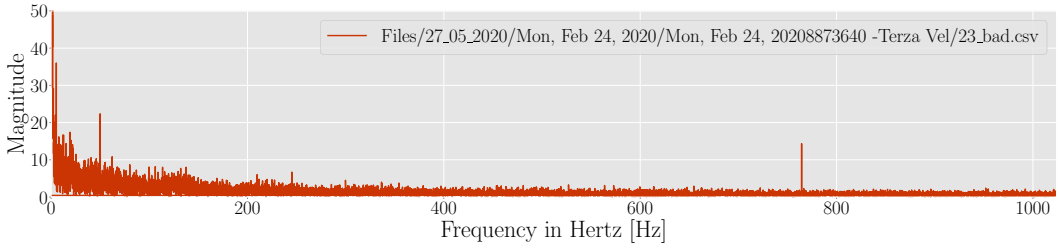
(a) Frequencies of a *Good* signal



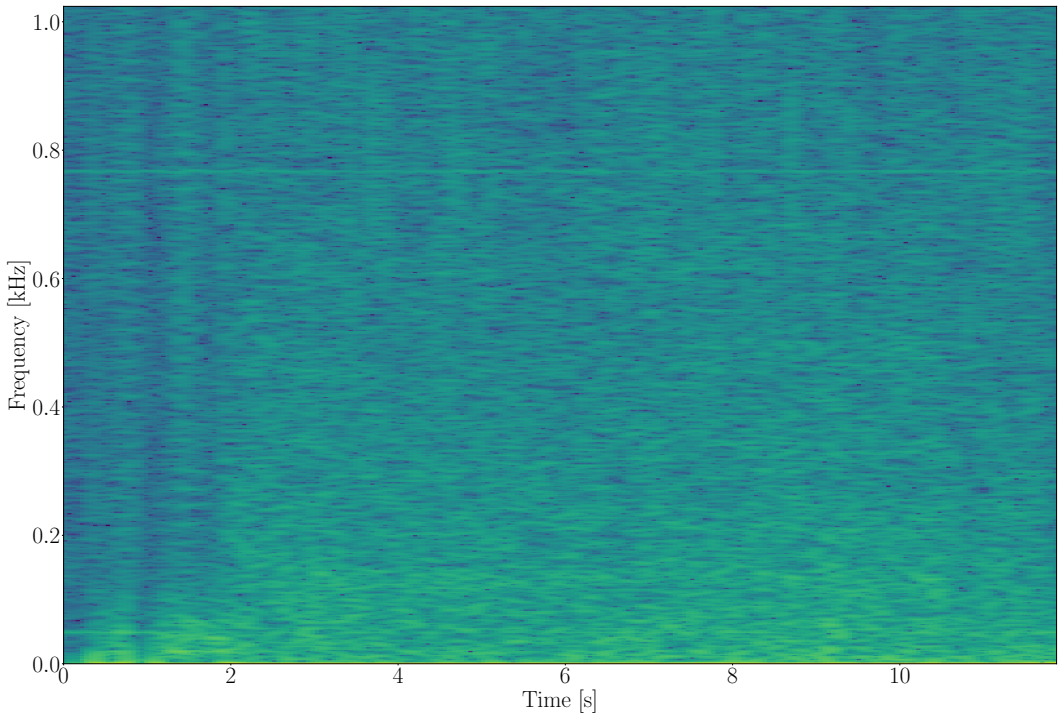
(b) Spectrogram of a *Good* signal

Figure 4.2: Frequency analysis of a signal labeled as *good*

The same analysis is performed on a signal labeled *bad* on the same production session of the previous signal. In figure 4.3a is visible that the most intensive frequencies are always 50[Hz] and 780[Hz]. The spectrogram 4.3b once again confirms this result. From these analyses in the frequency domain, no apparent correlation results between the signal's frequencies and its labels. *Good* and *bad* signals share the most prominent frequencies, which does not underline any potential difference given the belonging class.



(a) Frequencies of a *bad* signal



(b) Spectrogram of a *bad* signal

Figure 4.3: Frequency analysis of a signal labeled as *bad*

4.2.2 Time-Domain analysis

Count of windows in band of quality

One of the first analyses performed in the time domain, as explained in section 3.2.2, was determining how different a signal is from its *reference*. For this examination, I selected two production days of the `Velocity C`:

1. 2020/01/17: on this day, the average of the signals' means was around 0.11. According to CRF, this average value is low, and it is a potential sign that the welding machine requires maintenance. Given this scenario, having a low average would raise the chances that a signal will be labeled as *bad*.
2. 2020/02/24: on this day, the average of the signals' means was around 0.15. According to CRF, this average value is high, and the chances that a signal will be labeled as *bad* are lower than the scenario at point 1.

In figure 4.4 are shown the results of this experiment for the production day 2020/01/17 and the percentages of *bad* signals do not distinguish from the *good* ones. The same situation happens in figure 4.5 where the *bad* signals have some percentages of windows out of band even lower than *good* signals.

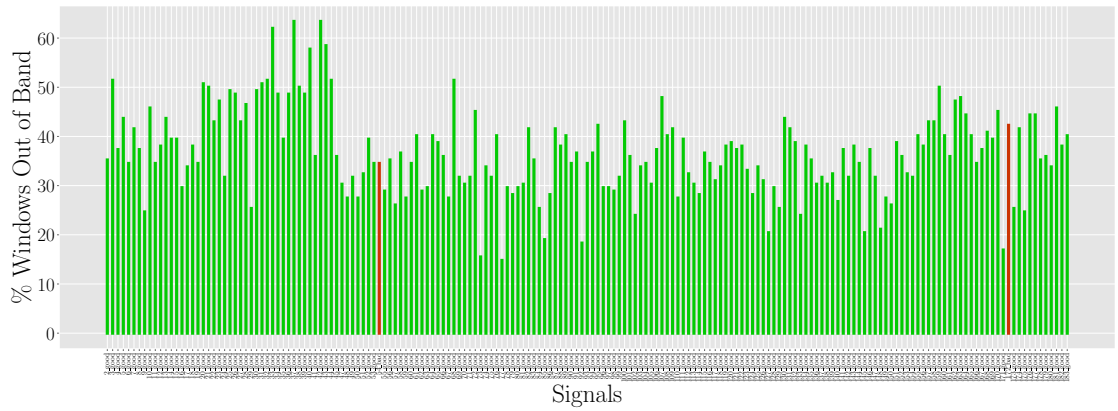


Figure 4.4: Percentages of windows out of band of quality per each signal of production day 2020/01/17 (low average)

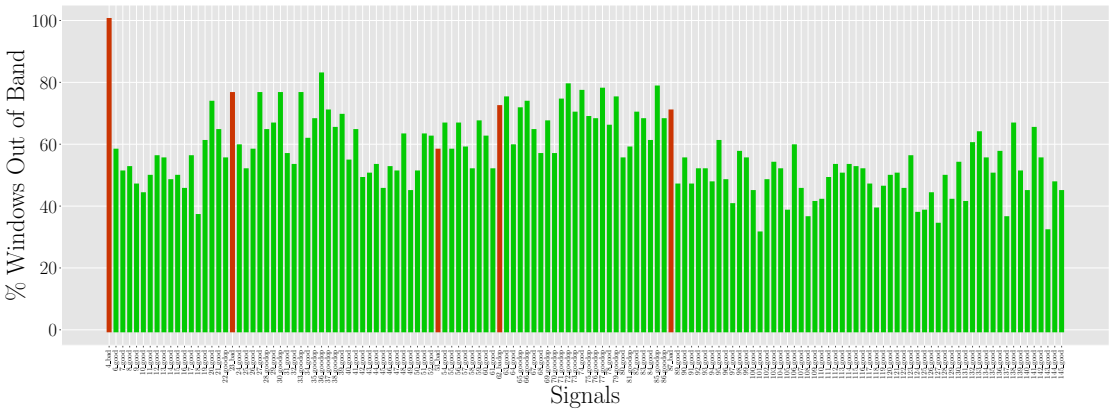


Figure 4.5: Percentages of windows out of band of quality per each signal of production day 2020/02/24 (high average)

Distance from *quality band*

The previous analysis shows the percentage of signal windows that are not inside the *quality band*, and now we compute how distant they are from an adaptive *quality band*. This time are considered all velocities, a window size of 128 samples, and a fraction equal to $\alpha = 0.2$ (the methodology is explained in section 3.2.2). As previously mentioned in the previous chapter 3.2.2, an adaptive *quality band* means that instead of a monotone trend, it follows the mean values of the *reference*'s windows. Figure 4.6 shows the result from computing the distance between the signal's windows and the *quality band*. The amount of information derived from this analysis is large, so I opted for a heatmap to represent it. It reports the distances for all the windows of all the `Velocity A` signals recorded on the 10Th February 2020. On the x-axis, there is the signal's filename, and on the y-axis, there is the window's number. The color bar on the right reports the distances of *bad* signals' windows with warm colors, while for *good* signals is used a cold color palette. The gradient of colors expresses how distant a window is from the *quality band*. In the case of null distances, when the signal's mean is similar to the *reference*'s, the color is white. Signal *references* are indicated with a pink mark on top of the heatmap, but they are also noticeable for the straight white lines under each marker. The straight white line in the heatmap represents the *reference*'s distance from itself, which is obviously null. This plot can be studied vertically (bottom to top) or horizontally (left to right). A vertical analysis shows how the distance varies from the first window to the last one. The horizontal analysis describes how the distance varies in all signals for the same window. By examining the plot 4.6 vertically, it is easy to see that the colors' pattern repeats almost periodically, and this is linked to the periodicity of `Velocity A` signals shown in figure 2.1a. The horizontal analysis underlines how signals differ from their *reference* almost identically, given how

colors have the same shade for almost the same windows. In this analysis, there are no `bad` signals, and in the selected portion of the dataset are absent `Velocity A` signals with that label. It was interesting to visualize that the distance between signals and *references*, for a specific window, is almost identical for all signals in a production session.

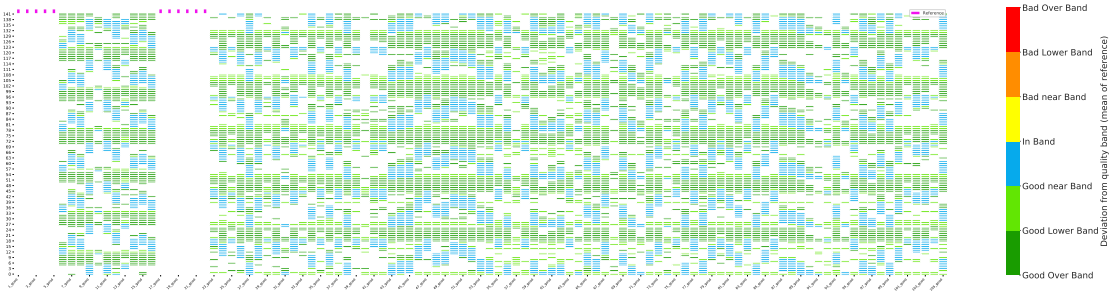
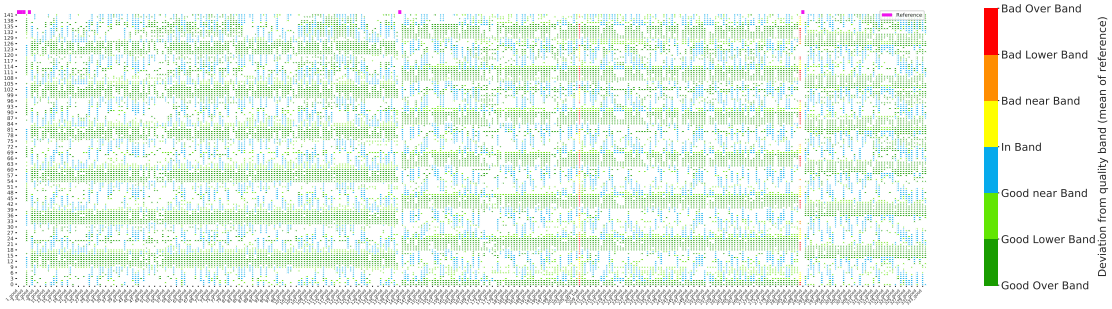


Figure 4.6: Heatmap of the distances for `Velocity A` on production day 2020/02/10

Figure 4.7a reports the results of the same methodology for `Velocity B` signals produced on the 28Th January 2020. Similarly to what is shown in figure 4.6, colors from bottom to top have a periodical trend due to `Velocity B` signals' periodicity (see figure 2.1a). In this experiment there are only two *bad* signal but their distance's color seem to follow the patter of all the others, showing no apparent correlation between a signal's label and the distance from its *reference*. Figure 4.7b shows the ECDF of the signals' windows that are not inside the *quality band*, divided per label. Since the number of out-of-band windows is larger for the *good* signals, the distribution is longer and more defined than the other. Also in this case was interesting to visualize how the signals' periodicity is perceivable when computing their distance from their *references*.

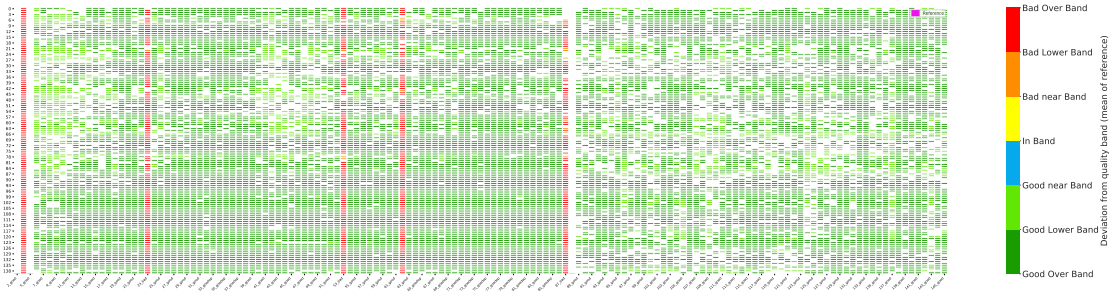


(a) Heatmap of the distances for `Velocity B` on production day 2020/01/28

(b) ECDF of windows out of *quality band* for production day 2020/01/28 of `VelocityB`

Figure 4.7: Distance Analysis of `Velocity B`

Figure 4.8a shows the distances per window of `Velocity C` signals' from the *quality band*. In this scenario we have no periodical color patterns since trends do not repeat like for `Velocity A` and `Velocity B`. The production day under analysis contains **5** *bad* signals which are all over the *quality band*, like most of *good* signals. This means that a *bad* label is not correlated to the distance from the *quality band*. An interesting result is that after *reference* `86_good` has been assigned, results improve: looking at the colors of the windows after that reference the amount of color white increases, meaning that signals' trend became more similar to their *reference*. Another interesting evaluation is that signals produced before *reference* `86_good` are generally more distant from their *references*. The first 60 windows of these signals are all in dark green/dark red, meaning that the distance from the *reference* is high and that their values are on top the *quality band*. Figure 4.8b shows the ECDFs of the windows out of the *quality band*, for all the signals divided per label. As we can see the ECDF of *bad* signals is more defined than the one in figure 4.7b because there more signals with that label. Even if the two distribution are different and separated, is not possible to retrive a significant information to motivate a correlation between the number of windows out of band and the label assigned to a signal.



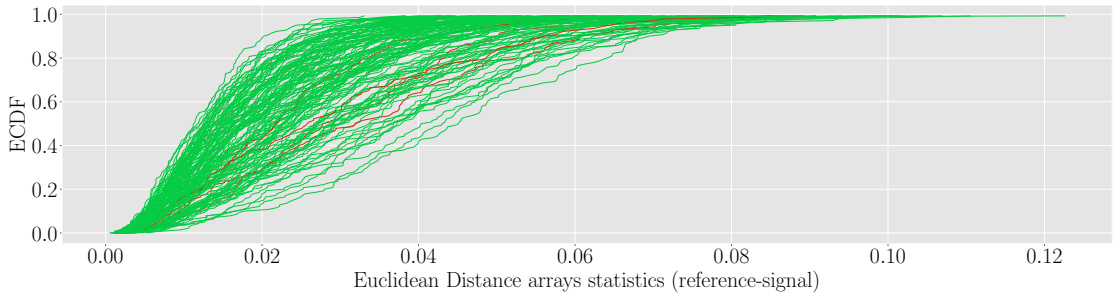
(a) Heatmap of the distances for `Velocity C` on production day 2020/02/24

(b) ECDF of windows out of *quality band* for production day 2020/02/24 of `Velocity C`

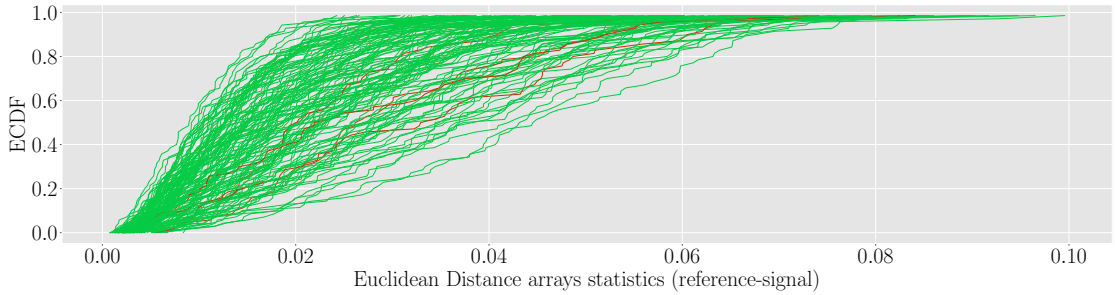
Figure 4.8: Distance Analysis of `Velocity C`

Euclidean distance between statistics arrays

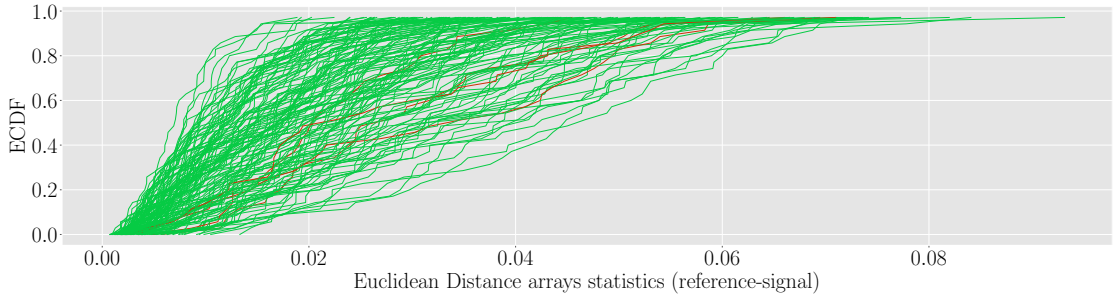
In figure 4.9 are shown the results of statistical arrays' euclidean distance. As explained in the previous chapter 3.2.2, signals produced on a determined production day are divided into windows (windows' sizes 128, 256, and 512). In figure 4.9a are shown the results of `Velocity C` signals, divided into windows of 128 samples. The ECDFs in green represent the Euclidean distances of *good* signals, while the few red ones represent the *bad* signals. Distributions do not show a distinct separation between the two labels, proving that the extracted features are not correlated to the assigned labels. In figure 4.9b and 4.9c are depicted the euclidean distances' distribution of signals split into windows of 256 and 512 samples, respectively. The results do not change but enlarging the windows' sizes, as we can see from the plots, reduces the resolutions of the ECDFs.



(a) ECDF of the euclidean distances between array of statistics per window of 128 samples



(b) ECDF of the euclidean distances between array of statistics per window of 256 samples



(c) ECDF of the euclidean distances between array of statistics per window of 512 samples

Figure 4.9: ECDF's of the distance of statistics' arrays for `Velocity C`

Linear Regression

The following figures 4.10 4.11 4.12 show the results of linear regression. The algorithm was performed on different versions of signals (raw, low-pass filtered and mean) and for all velocities. Each Plot is dedicated to one of the three parameters listed in section 3.2.2 and to one of the three velocities. All plots have on the x-axis the signals' names, and on the y-axis the magnitude of the parameter for all signals' versions. Given the different scales of parameters' results for each version of the signal, results have been normalized between 0 and 1 using the *MinMax* methodology. Parameters are represented with a scatter plot that changes color

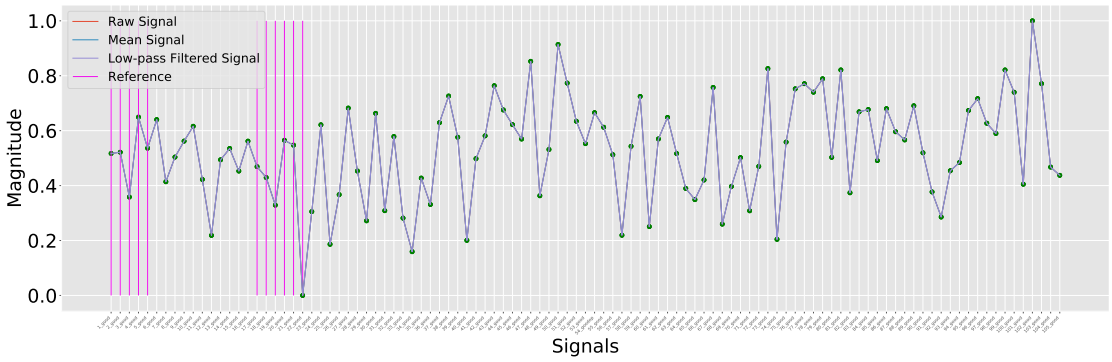
according to the label: green for *good* and red for *bad*. *References* are marked with straight pink lines.

Figure 4.10 illustrates the result of `Velocity A` signals. The first plot 4.10a represents the slope parameter for each signal of the production day (10/02/2020),figure 4.10b represents the values for the intercept parameter, and 4.10c represents the R^2 parameter. After normalization, all results perfectly overlap, showing no difference among signals' versions. On this production day, no *bad* signals were created. Therefore it is not possible to evaluate a potential difference in the results according to the assigned label.

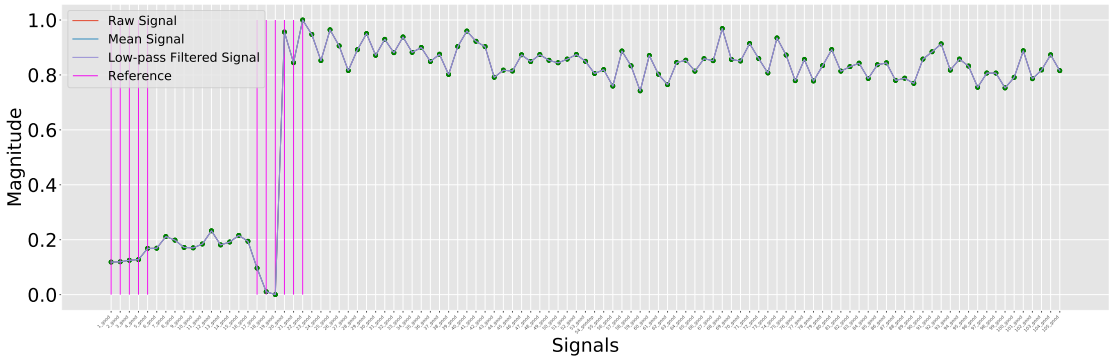
Figure 4.11 illustrates the results for `Velocity B` signals. On this production day (28/01/2020), are present two bad signals. Also in this case when normalizing all the values, the different signals' plots overlap showing no difference. The analysis of these parameters related to assigned labels, shows an anomaly with signal *207_bad* which actually differentiates from *good* signals. In figures 4.11a and 4.11c it has a higher results than the average, while in 4.11b it is lower. Even if it seems that this *bad* signal is different from the others, it cannot be considered as an evidence of a potential difference correlating LR parameters to labels. This is due to the fact that the other *bad* signal (*288_bad*) has similar results as the *good* label for all the three parameters.

Lastly all the results for `Velocity C` signals are reported in figure 4.11. Equally as in figure 4.11, *bad* signals do not distinguish from their counterpart for all of three parameters and signals' plots overlap in figure 4.12a and 4.12b. Unlike all the other results for the R^2 parameter for the other velocities, `Velocity C` signals do not overlap for the R^2 parameter, as shown in figure 4.12c.

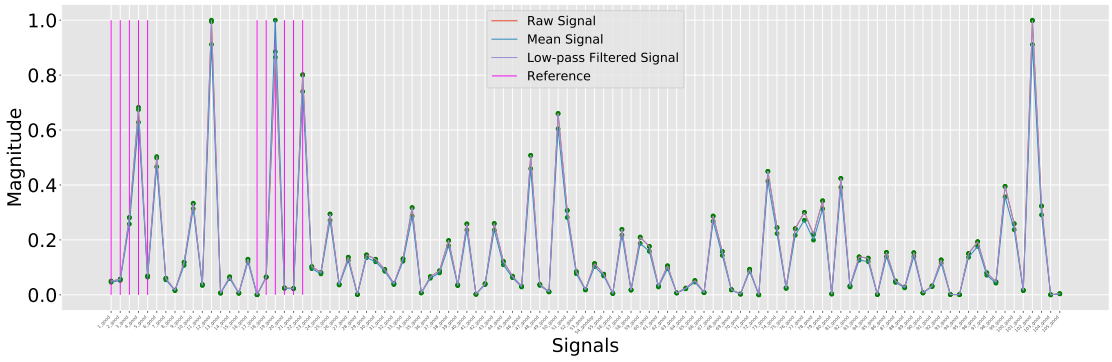
To conclude, it can be said that LR parameters seem to not be correlated to signals' assigned label.



(a) Slope parameter of `Velocity A` signals

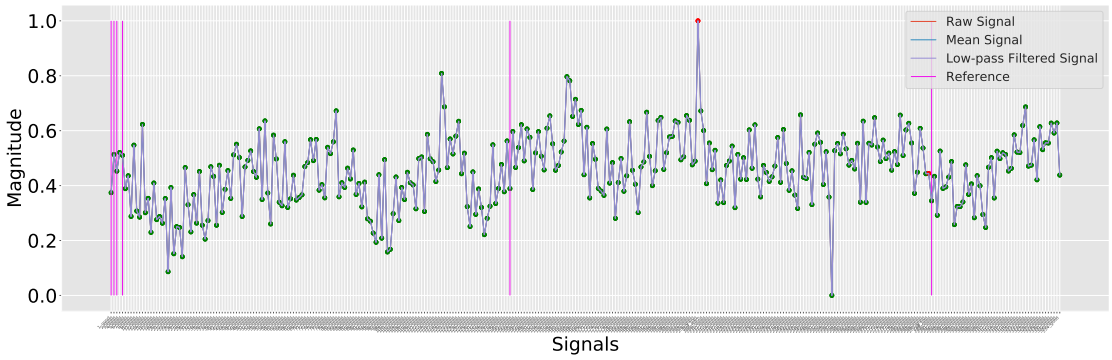


(b) Intercept parameter of `Velocity A` signals

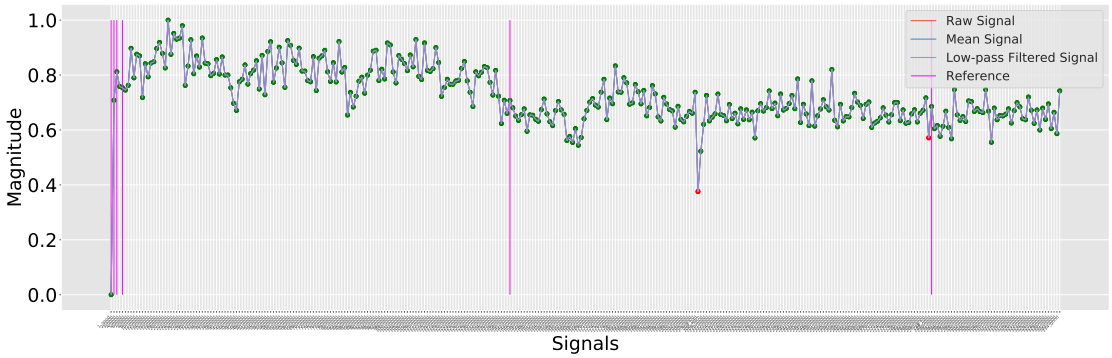


(c) R^2 parameter of `Velocity A` signals

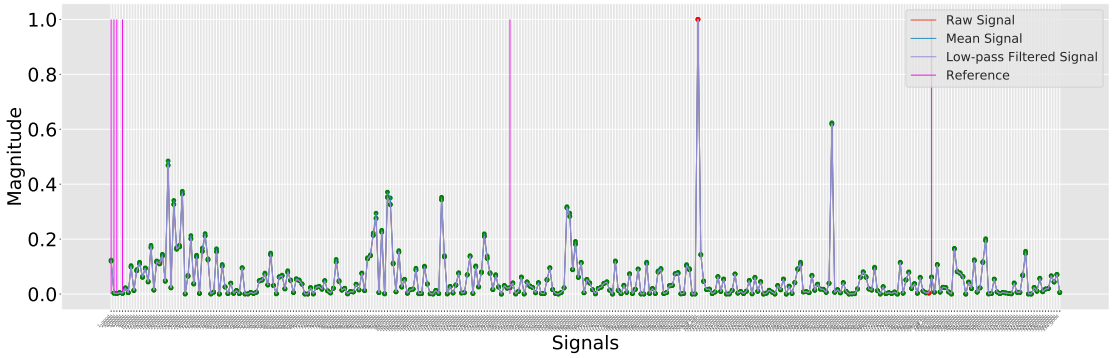
Figure 4.10: Parameters of Linear Regression for `Velocity A` on production day 2020/02/10



(a) Slope parameter of `Velocity B` signals

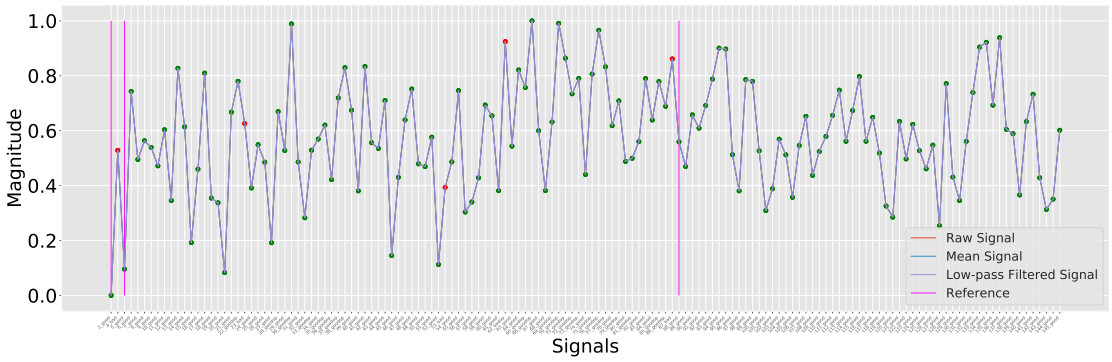


(b) Intercept parameter of `Velocity B` signals

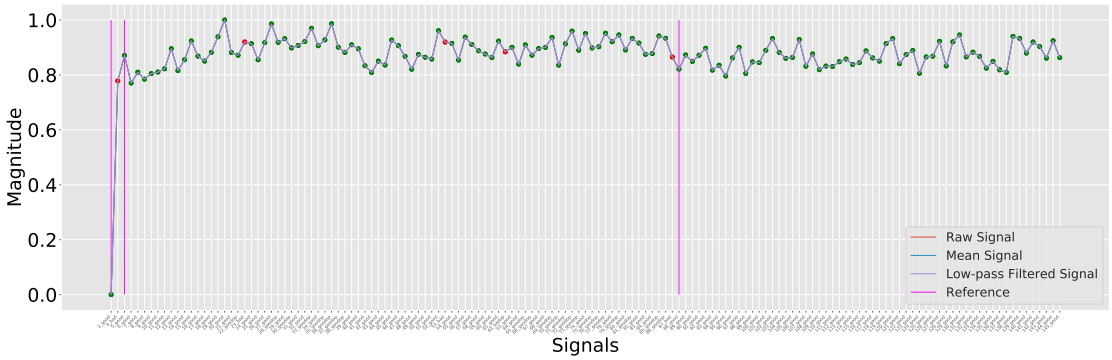


(c) R^2 parameter of `Velocity B` signals

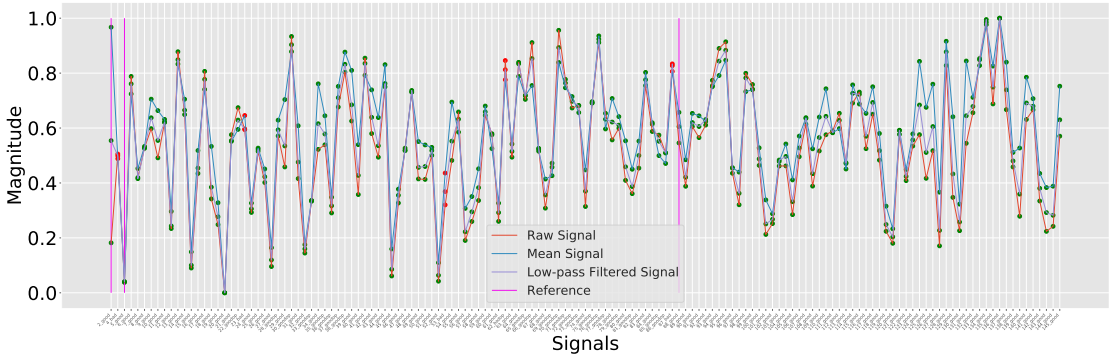
Figure 4.11: Parameters of Linear Regression for `Velocity B` on production day 2020/01/28



(a) Slope parameter of `Velocity C` signals



(b) Intercept parameter of `Velocity C` signals



(c) R^2 parameter of `Velocity C` signals

Figure 4.12: Parameters of Linear Regression for `Velocity C` on production day 2020/02/24

Chapter 5

Conclusion

To draw the conclusion of this work, let us review the previous chapters' main points.

The analyzed dataset comprises two complementary subsets: time-series signals and their relative *metadata* containing pre-computed information such as the signal's label, mean, standard deviation, and energy.

The analyses on *metadata* showed that signal labels are not correlated to the pre-computed features. Training both a linear and non-linear classifier demonstrated that it is impossible to assign labels considering only the signal's features precisely. The learning model could be improved by including additional information in the training set, such as environmental temperature, which could affect the weld's output. The analyses for the time series were carried out with two methodologies. The first consisted of computing how distant the signals' windows were from those of their *references* and developing a tailored heat map to visualize the results for all the windows of all the signals recorded on a specific production day. The results showed no correlation between the signal's labels and their distance with the *reference*. However, the adopted visualization method resulted in being an effective tool to describe signals' behavior for a whole production day.

The approaches of this work could be developed in future projects, for example *metadata*'s classification could be improved including descriptive measurements of the laser machine's environment. Another development could be to use the already-developed methodologies with datasets produced in different years.

To conclude this work, it is important to underline that predictive maintenance is the key to improve production processes and industrial machines' reliability over time. Nowadays industrial productions are more demanding both in terms of output quantities and quality. PdM could help in reducing time and product wastes by forecasting when machines could require assistance and subsequently scheduling maintenance between production sessions.

Bibliography

[1] Zeki Murat Çınar, Abubakar Abdussalam Nuhu, Qasim Zeeshan, Orhan Korhan, Mohammed Asmael, and Babak Safaei. «Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0». In: *Sustainability* 12.19 (2020). ISSN: 2071-1050. DOI: 10.3390/su12198211. URL: https://www.mdpi.com/2071-1050/12/19/8211 (cit. on p. i).

[2] Wikipedia. *Industrie 4.0 — Wikipedia, The Free Encyclopedia*. http://de.wikipedia.org/w/index.php?title=Industrie%204.0&oldid=208387613. [Online; accessed 07-March-2021]. 2021 (cit. on p. 1).

[3] Henning Kagermann, Wolf-Dieter Lukas, and Wolfgang Wahlster. *Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution*. 2011. URL: https://www.ingenieur.de/technik/fachbereiche/produktion/industrie-40-mit-internet-dinge-weg-4-industriellen-revolution/ (cit. on p. 1).

[4] Tiago Zonta, Cristiano André da Costa, Rodrigo da Rosa Righi, Miromar José de Lima, Eduardo Silveira da Trindade, and Guann Pyng Li. «Predictive maintenance in the Industry 4.0: A systematic literature review». In: *Computers & Industrial Engineering* (2020), p. 106889 (cit. on p. 1).

[5] Wikipedia. *Data mining — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Data%20mining&oldid=1010751186. [Online; accessed 10-March-2021]. 2021 (cit. on p. 2).

[6] Terry Wireman. *World class maintenance management*. Tech. rep. INDUSTRIA PRESS, 1990 (cit. on p. 2).

[7] Maurizio Bevilacqua and Marcello Braglia. «The analytic hierarchy process applied to maintenance strategy selection». In: *Reliability Engineering & System Safety* 70.1 (2000), pp. 71–83 (cit. on p. 2).

[8] Hongzhou Wang and Hoang Pham. «A quasi renewal process and its applications in imperfect maintenance». In: *International journal of systems science* 27.10 (1996), pp. 1055–1062 (cit. on p. 2).

[9] Andrew KS Jardine, Daming Lin, and Dragan Banjevic. «A review on machinery diagnostics and prognostics implementing condition-based maintenance». In: *Mechanical systems and signal processing* 20.7 (2006), pp. 1483–1510 (cit. on p. 2).

[10] M. You, F. Liu, W. Wang, and G. Meng. «Statistically Planned and Individually Improved Predictive Maintenance Management for Continuously Monitored Degrading Systems». In: *IEEE Transactions on Reliability* 59.4 (2010), pp. 744–753. DOI: 10.1109/TR.2010.2085572 (cit. on p. 3).

[11] Hongfei Li, Dhaivat Parikh, Qing He, Buyue Qian, Zhiguo Li, Dongping Fang, and Arun Hampapur. «Improving rail network velocity: A machine learning approach to predictive maintenance». In: *Transportation Research Part C: Emerging Technologies* 45 (2014). Advances in Computing and Communications and their Impact on Transportation Science and Technologies, pp. 17–26. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2014.04.013. URL: https://www.sciencedirect.com/science/article/pii/S0968090X14001107 (cit. on p. 3).

[12] Mikel Canizo, Enrique Onieva, Angel Conde, Santiago Charramendieta, and Salvador Trujillo. «Real-time predictive maintenance for wind turbines using Big Data frameworks». In: *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*. IEEE. 2017, pp. 70–77 (cit. on p. 3).

[13] In: (). URL: http://minicon.iccs.ntua.gr/ (cit. on p. 4).

[14] G. Makridis, D. Kyriazis, and S. Plitsos. «Predictive maintenance leveraging machine learning for time-series forecasting in the maritime industry». In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–8. DOI: 10.1109/ITSC45102.2020.9294450 (cit. on p. 4).

[15] C. Lin, Y. Hsieh, F. Cheng, H. Huang, and M. Adnan. «Time Series Prediction Algorithm for Intelligent Predictive Maintenance». In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 2807–2814. DOI: 10.1109/LRA.2019.2918684 (cit. on p. 4).

[16] *Holdouts and Cross Validation: Why the Data Used to Evaluate your Model Matters*. 2021. URL: https://community.alteryx.com/t5/Data-Science/Holdouts-and-Cross-Validation-Why-the-Data-Used-to-Evaluate-your/ba-p/448982?lightbox-message-images-448982=71553i43D85DE352069CB9 (cit. on p. 18).

[17] Jason Brownlee. *What is the Difference Between Test and Validation Datasets?* 2020. URL: https://machinelearningmastery.com/difference-test-validation-datasets/ (cit. on p. 18).

[18] Satish Gunjal. *K Fold Cross Validation*. 2020. URL: https://satishgunjal.com/kfold/ (cit. on p. 18).

[19] Simplilearn. *Classification in Machine Learning: The Best Classification Models*. 2021. URL: https://www.simplilearn.com/tutorials/machine-learning-tutorial/classification-in-machine-learning (cit. on p. 18).

[20] *Decision tree*. 2021. URL: https://en.wikipedia.org/wiki/Decision_tree (cit. on p. 19).

[21] Pratik says: and EdurekaSupport says: *Decision Tree: Decision Tree Introduction With Examples*. 2020. URL: https://www.edureka.co/blog/decision-trees/ (cit. on p. 19).

[22] Lujing Chen. *Support Vector Machine - Simply Explained*. 2019. URL: https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496 (cit. on p. 21).

[23] *Support-vector machine*. 2021. URL: https://en.wikipedia.org/wiki/Support-vector_machine#Linear_SVM (cit. on pp. 21–23).

[24] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009 (cit. on p. 21).

[25] Drew Wilimitis. *The Kernel Trick*. 2019. URL: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f (cit. on p. 22).

[26] Raoof Naushad. *Fourier Transform for Image Processing in Python from scratch.* 2020. URL: https://medium.datadriveninvestor.com/fourier-transform-for-image-processing-in-python-from-scratch-b96f68a6c30d (cit. on p. 24).

[27] Vidya Muthukrishnan. *Cutoff Frequency: What is it? Equation & How To Find it*. 2020. URL: https://www.electrical4u.com/cutoff-frequency/#Cutoff-Frequency-of-a-Low-Pass-Filter (cit. on pp. 24, 25).

[28] *Low-pass filter*. 2021. URL: https://en.wikipedia.org/wiki/Low-pass_filter (cit. on p. 25).

[29] *Linear regression*. 2021. URL: https://en.wikipedia.org/wiki/Linear_regression#Intercept (cit. on p. 31).

[30] *Simple linear regression*. 2021. URL: https://en.wikipedia.org/wiki/Simple_linear_regression (cit. on p. 31).

[31] Berland. *Linear Regression Example*. URL: https://it.m.wikipedia.org/wiki/File:LinearRegression.svg (cit. on p. 31).

[32] Real Python. *Linear Regression in Python*. 2021. URL: https://realpython.com/linear-regression-in-python/ (cit. on p. 32).

[33] *Stat Trek*. URL: https://stattrek.com/statistics/dictionary.aspx?definition=coefficient_of_determination (cit. on p. 33).

[34] Jason Brownlee. *Failure of Classification Accuracy for Imbalanced Class Distributions*. 2021. URL: https://machinelearningmastery.com/failure-of-accuracy-for-imbalanced-class-distributions/ (cit. on p. 35).

[35] Lorenzo Govoni. *Apache Spark: la piattaforma per elaborare i big data*. 2020. URL: https://www.lorenzogovoni.com/apache-spark-lo-strumento-per-elaborare-i-big-data/ (cit. on p. 37).

[36] *sklearn.svm.SVC*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC (cit. on p. 37).