POLITECNICO DI TORINO

Master degree course in Software Engineering

Master Degree Thesis

Supporting Predictive Maintenance through a semi-supervised labelling of robot cycles

Supervisors
prof. Tania Cerquitelli

Candidates
Luca Mazzucco

Accademic Year 2018-2019

This work is subject to the Creative Commons Licence

Summary

Predictive maintenance has always been a difficult problem in real word modern industries. With the new Industry 4.0 the manufacturing environments are becoming digital factories and this context produce vast volumes of raw data.

Enhancing manufacturing intelligence brings a wide range of benefits, predictive diagnostics is one of the most important. To sustain this well known issue, this work present the design and development of a semi-supervised data-driven methodology, to characterize multi-cycle processes and support robot cycle labelling.

The latter exploits the best developed clustering algorithms, discovering automatically clusters of production cycles through time-independent common properties.

Later, a self-tuning strategy has been integrated to help the selection of the best approach, input data and parameter.

Finally, each cluster is locally characterized from a set of most relevant features.

Acknowledgements

More than six months ago I started to work on this thesis and many things happened since then. I would like to thank a lot of people who supported me, technically and morally on developing this thesis in the last months.

First and foremost, I want to express my gratitude to my supervisor professor Tania Cerquitelli, who gave me the opportunity to work on this challenging and interesting project, wich was for me a completely unknown subject before that moment.

I'm really thankful to her for her patience and her helpful comments and observations.

My gratitude goes also to Francesco, who helped me when I got stuck during this period in many task with his deep knowledge and important advices and of course, for his friendship too. The same goes to the welcoming working environment I found with all the other people of Lab5.

Moreover, a huge, huge thanks goes to my family that supported me in all those years that brought me at this point.

And last, I would like to thanks all my friends with wich i shared this long journey, that without them wouldn't have been the same, enriching an important part of my life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the new Industry 4.0, the manufacturing environments are evolving in digital factories and this context produce enormous volumes of raw data. Leading the data will help managers on making better-informed business decision, improving production processes and bringing advantages to who will lead this fundamental knowledge.

Predicting and avoiding equipment and devices malfunctioning or potential failure, bring several advantages in terms of efficiency and production line maintenance costs.

This work introduce a supporting approach for predictive maintenance challenges, which represent a big task still mainly to be explored and exploited for the incoming Industrial 4.0 revolution, applying semi-supervised data-drive methodologies.

The objective of the presented study is to identify the proper configuration, in terms of clustering algorithms and parameters, and data pre-processing criterias, that better represent the original dataset and better support the predictive maintenance analysis, through the evaluation of cluster quality indices. Moreover, from the produced clusters with the selected configuration will be extracted the most relevant properties that characterize them.

The presented analysis are based on a robot (for industrial production needs) manufacturing dataset. Several robot cycles production processes has been monitored, which are often repeated periodically and characterized by an individual duration.

All the experiments has been produced by developing semi-supervised algorithms in Python language, over a dataset imported as a Json file, partially processed with the support of Google Collaboratory notebooks.

The developed semi-supervised approach consists of various subsequent

steps, in particular: data pre-processing, data aggregation, clusters validation and clusters characterization.

In chapter 2 we will see the current state of art in Industry 4.0, having a first preview of the industrial context under analysis in this work. In particular, each monitored cycle, after having collected the correspondents electrical signals value, has been splitted over the time domain, to simpler extract the variability independently for each sub-cycle.

This features collection is later subjected to a correlation selection through the Pearson index. The obtained features for each cycle compose the so called Smart Data, used by the subsequent steps from the analytical process for predicting each cycle outcome.

Later, in chapter 3, the various implementation steps are exposed. The data pre-processing phase consists on producing various starting Smart Data through different splittings criteria of the original electrical signals. Since this phase stays on top of the analysis procedure, is fundamental to understand how the initial parameters influence the produced Smart Data and consequently the obtained further analysis, with the aim of maximize the clustering quality and possibly, reduce the features amount. The subsequent data-aggregation phase consist on applying various unsupervised machine learning algorithms, like K-Means, DbScan, Agglomerative and Gaussian Mixture. Each of them has been applied with several different input parameters, useful for a later model comparison.

These evaluations are performed by the clustering validation blocks, consisting on comparing the previously generated models through a wide set of validation indices, both supervised and unsupervised. The aim of this step is therefore to identify with which algorithms and parameters is possible to maximize the clustering quality.

The last phase is the so called Clustering characterization. Once selected the best clustering configuration from the previous steps, this block is demanded to extract the main relevant features and segments that more influence the production cycles, helping the domain experts to better understand a specific meaning for every group.

Chapter 4 list the platforms, procedures and programming instruments adopted for developing the aforementioned work.

Finally, chapter 5 shows experimental results got following the previously proposed methodology, followed by a chapter listing the possible future development.

Chapter 2

State of the art

2.1 Industry 4.0

The previous three industrial revolutions were all triggered by mechanical innovations: the introduction of water powered and steam powered mechanical manufacturing at the end of the 18th century, the division of labor at the beginning of the 20th century and the introduction of PLC (programmable logic controllers) for automation purposes in manufacturing in the 1970s [1].

According with industry and research experts, the upcoming industrial revolution will be triggered by the Internet (and IoT), which allows communication between humans as well as machines in Cyber-Physical-Systems (CPS) throughout large networks.

Industry 4.0 put focus on the creation of intelligent products and production processes. Manufacturing, factories have to cope with the need of really rapid product development, flexible industrial chain of production as well as complex environment [2].

The introduction of sensor in the manufacturing environment leads companies to a completely new scenario, where new approaches are needed to cope with a new huge amount of informations, that need to be analyzed to get new knowledge.

Step by step, more and more production equipment and industrial machineries (with the real-time sensorization) are continuously increasing the industrial data collecting: as a consequence, a methodology need to be developed to extract useful information from this amount of data to better support the decision-making [3].

Cyber-Physical Systems are pushing modern industries in transforming data into valuable knowledge. However, this new scenario is far away from being well explored, instead, represents a difficult challenge. Leading the data will help managers on making better-informed business decision, improving production processes and bringing advantages to who will lead this important knowledge.

Predictive maintenance has always been a difficult problem in real word modern industries. With the new Industry 4.0 the manufacturing environments are becoming digital factories and this context produce vast volumes of raw data.

Different solutions [4], [5], [6], [7] exploit Big Data frameworks to cope with this modern industrial scenario. In particular, in [4], authors show a scalable solution exploiting free licensed technologies (i.e. Apache Kafka and Spark) for offline and online processing. In [5] is instead presented a Big Data analytics framework to improve health monitoring services, useful also for different application on an aerospace and aviation industrial. An integrated Self-Tuning Engine for Predictive maintenance for Industry 4.0 is instead presented in [6], getting advantages from Big Data technologies and systems (i.e. Cassandra, Spark, Kafka) and running on top contenerized Docker environment. Moreover, in [7] a further solution for predictive maintenance is proposed, tailored to wind turbines monitoring, presenting a data-drive solution deployed for predictive model generation.

Enhancing manufacturing intelligence brings a wide range of benefits, predictive diagnostics is one of the most important. To sustain this well known issue, this work present the design and development of a semi-supervised data-driven methodology, to characterize multi-cycle processes and support robot cycle labelling.

The majority of the previously cited articles doesn't consider the performance degradation of their model that the upcoming of new unknown data distributions can cause over a prediction process.

2.1.1 A Smart Data

As mentioned before, this work is based on a robot (for industrial production needs) manufacturing dataset, to support robot cycle labelling, evaluation and as a consequence, to support predictive maintenance.

Nowadays Industrial sensor, monitor several production processes, that are often repeated periodically and characterized by a individual duration. This new scenario gives the opportunity to produce a Smart Data, which is in

charge of processing and transforming raw data for extracting main features describing the whole signal. Each monitored cycle, after having collected the electrical signals values, is splitted over the time domain, to simpler extract the variability indipendently for each sub-cycle. Every split is therefore represented by several statistical features (i.e. standard deviation, mean, Kurtosis, quartiles, skewness, sum of absolute values, root mean squared error, absolute energy, number of elements over the mean, mean absolute change, and much more..).

This features collection is the so called Smart Data, used by subsequent steps from the analytical process for predicting each cycle outcome.

2.1.2 Experimental context

The real experimental context use case on which this work has been tested on consists in predicting the suitable tensioning level (in electric signal terms) of the installed belts in a robot axis, evaluating the consumed electricity by the engines and actuators.

To discretize this belt tensioning levels, in this use case washers has been used. Each different number of washers correspond to a different label, knowing this means having the ground truth knowledge. A lower belt tensioning consumption correspond to an higher number of washers.

The needed washers number represent the assigned label at each measured electricity cycle. The objective is to predict the right number of washer, corresponding to the most suitable belt tensioning for each incoming electricity cycle.

NumWashers	# of cycles	Dataset %
0	6395	20,206 %
1	19915	62,926 %
2	5338	16.868 %
Total cycles	31648	100 %

Table 2.1: Washers dataset distribution

Table 2.1 shows robot cycles distribution related to washers number, divided so by class. The correct precision and functioning of the robot is assured by choosing the best tensioning of the belt.

An uncorrect configuration will lead to different malfunctioning: low tension values cause premature belt's and pulley's wear and slippage, while, in

the other hand, an over tensioning lead to excessive strain on belts, which correspond into components overheating.

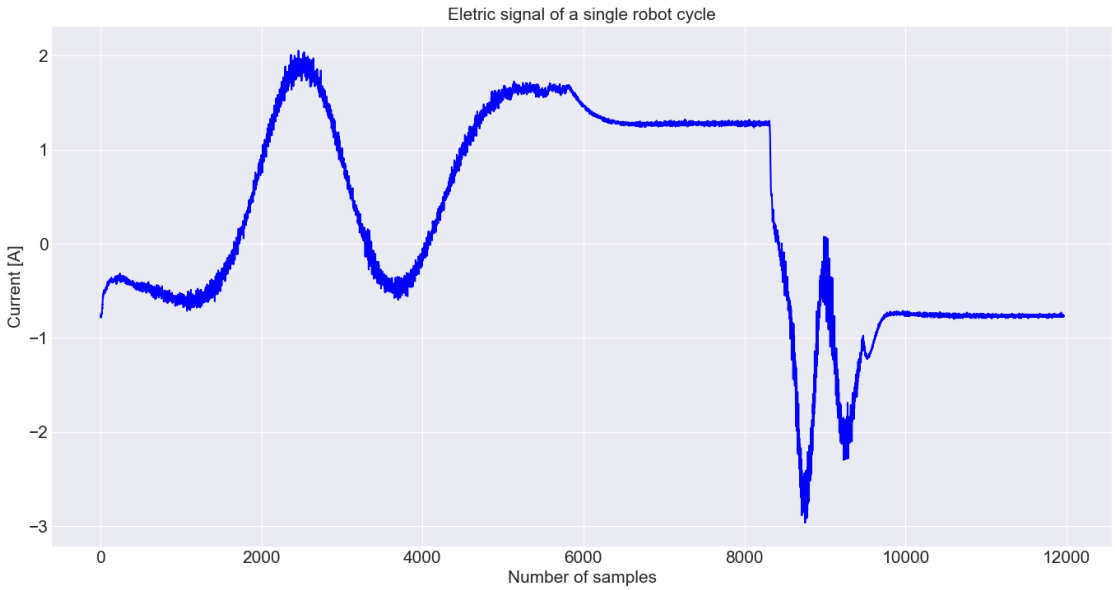


Figure 2.1: Electric signal of a single robot cycle

In the above Figure 2.1 is shown instead an example of the electricity consumption value trend for different washer numbers required to tensioning the belts.

2.1.3 Data preparation

Later, all incoming electrical signals has been splitted, and each single split has been characterized by several statistical features. More splitting configurations have been applied for better evaluating and comparing this phase.

Since for every split an high number of features has been generated (i.e. std, mean, Kurtosis, quartiles, skewness) up to almost 350, a feature selection approach has been applied. Through the Pearson index correlation, the number of features is reduced according to a choosen threshold.

From this step, quality and performance of the next model will be affected, from here the need to try different configuration to compare. Also, since the amount of features generated depend on the number of splits (being the total amount equal to the sum of features of each split), reducing this parameter lead to a lower total features amount and lower computing complexity as a natural consequence.

In [8], a solution taking into account 24 splits is proposed. In this work, different splitting configuration will be proposed and compared, again, with the objective of having lighter computation with not considerable precision looses.

Number of Split	# of features	[8] Features %
4	30	12,82 %
6	50	21,35 %
8	63	26,92 %
10	87	37,17 %
12	110	47,00 %
18	172	73,50 %
24	234	100 %

Table 2.2: Dataset features distribution over splitting configurations

In Table 2.2 is shown the amount of generated features with different splitting configuration, compared, in the last column with the amount used in the solution [8].

Choosing a lower number of split would lead to a comparable computation complexity reduction to the shown percentage.

Later in this work, in chapter 5, splitting configurations will be analyzed, compared and presented.

Next sections are dedicated to understand a fast overview of the main machine learning techniques: supervised, unsupervised and semi-supervised learning.

Other techniques like reinforcement learning will not be discussed in this document.

The main difference that characterize them is the a priori knowledge of the ground truth.

2.2 Supervised data analytics

Supervised learning is usually done in the classification context, when the objective is to map input to output labels, or regression, when is need to connect input to a continuous output. Most used algorithms in supervised learning contain naive bayes, logistic regression, artificial neural networks, support vector machines, and random forests. In both classification and

regression, the objective is to find specific structure or connections in the input data which allow us to properly produce correct output data.

Considering that "correct" output is determined completely by the training data, having a ground truth considered true from our model, it isn't direct to conclude that data labels in real-world situations are always correct.

Incorrect or noisy data labels will obviously decrease the effectiveness of the built model.

When applying supervised learning, the model complexity and bias-variance trade-off are the main considerations. Moreover, both of these are interrelated.

Model complexity relate to the function complexity to learn — similar to the polynomial degree. A proper model complexity level is generally determined by the training data nature. A low-complexity model is more suitable to a small amount of data, or not uniformly spread data among different possible scenarios. An high-complexity model will overfit if applied on a small dataset. Overfitting refers to a function that fits very well the training data, but doesn't generalize with other data points.

Bias-variance trade-off also relate to model generalization. In every model, there is a trade-off between bias, which correspond to the constant error term, and variance. The latter is the variance amount that the error may get between different training sets.

Therefore, high bias and low variance correlate to a model mainly wrong at 20% of the time, while a low bias and high variance bring to a model that may be wrong from 5%-50% of the time, depending on the used data for training. Bias and variance usually move in opposite directions of each other; decreasing bias will usually lead to higher variance, and vice versa.

When developing a model, the nature of your data and the specific problem should support on making an informed choose on where to fall on the bias-variance spectrum. In general, decreasing bias (and increasing variance) corresponds in models with relatively guaranteed baseline performance levels, which can be critical in specific tasks.

Moreover, in order to create well generalized models, the model variance should scale with the dimension and complexity of the training data — large, complex data-sets will usually request higher-variance models to completely learn the data structure, and simple, small data-sets should generally be learned with low-variance models.

2.3 Unsupervised data analytics

As already mentioned before, in unsupervised data analytics there isn't a ground truth knowledge.

The most frequent unsupervised learning tasks are clustering, density estimation, and representation learning. In each of these cases, the objective is to learn the correspondent structure of the dataset without beforehand provided labels. Since labels are not provided, comparing models performance in unsupervised learning methods is a difficult task.

Two frequent unsupervised learning use cases are dimensionality reduction and exploratory analysis.

Unsupervised learning is really helpful in exploratory analysis, since it can directly recognize data structures. If an analyst try to segment consumers, unsupervised clustering methods represent a good starting point for analysis. In conditions where it is hard or impractical for humans to guess data trends, unsupervised learning provides initial insights that can then be exploited to test single hypotheses.

Dimensionality reduction, which relate to represent data exploiting less columns or features, can be conducted through unsupervised approaches. In representation learning is need to find relationships between individual features, permitting to represent data with the latent features that interrelate initial features. Having far fewer features than the starting set, it can makes possible to do further data processing much less intensive, and eliminating redundant features.

	supervised learning	Unsupervised learning
Discrete	Classification	Clustering
Continous	Regression	Dimensionality reduction

2.4 Semi-supervised data analytics

Semi-supervised learning is mainly just what it sounds like: a training dataset with both labeled and unlabeled data. Think to it as happy medium between the previously illustrated solutions.

In this work, this will be the main applied method, comparing with and without using labels the obtained solutions. This approach is particularly helpful when recognizing relevant features from the data is hard, and labeling examples is a time-intensive task for experts.

Chapter 3

Proposed methodology

3.1 Overview

The main blocks of the industrial context on which this work is based and tested is presented in Figure 3.1. An Event Hub gather the sensor values from the production line, being in our case an international robotics industry, from industrial robot arms. As explained in 2.1.3, the electricity consumption needed from the engines in each single robot cycle represent the main information on which to apply data analysis and produce valuable knowledge.

The Event Hub, after having collected the data from the various sensors, submit the values to the factory monitoring block and to the semi-supervised analysis block.

The real-time factory monitoring provides through KPI (Key Performance Indicators) informative dashboards computed on data collected from production lines. The outcome of this generated statistics could be of interest for two roles in production floors: shift supervisors on production lines and production managers. From the moment that production managers are mainly interested on evaluating and guarantee an optimal production level, the OEE (Overall Equipment Effectiveness) index is evaluated. From the other side, the production shift supervisors need the OEE for unforeseen problematic issues and suddenly react to solve the causes.

Production managers are interested also to asses the conceptual and real context aspects that can contribute to factory inefficiency like equipment and device conditions. Unfortunately, the majority of predictive maintenance methods relates on supervised algorithms, needing a-priori ground

truth knowledge (i.e. labels) on the scenario under analysis that is commonly not available.

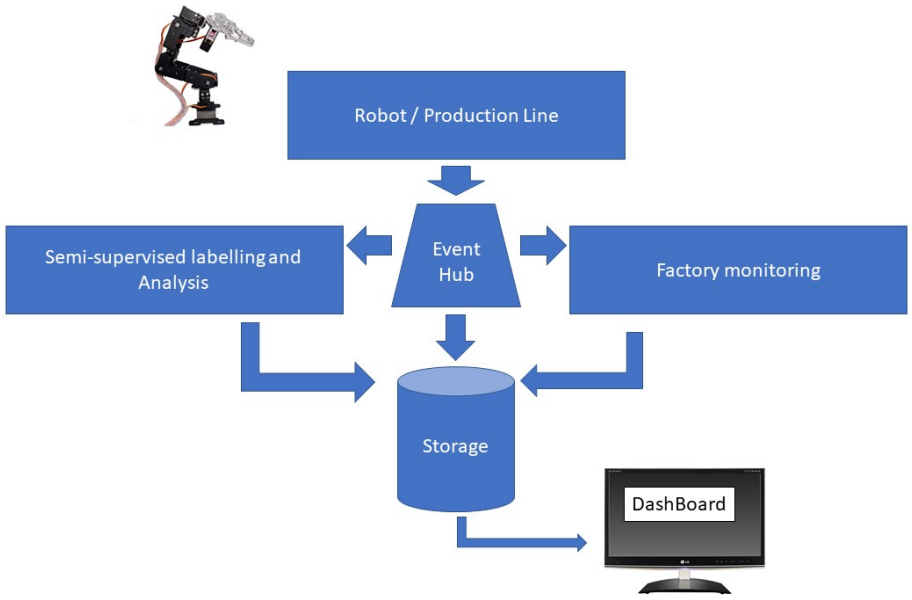


Figure 3.1: Production and Data collection Architecture

For addressing this issue, the proposed approach include semi-supervised labelling methods, aimed to infer labels automatically.

This work has been developed with the objective to develop specifically this semi-supervised labelling and analysis block shown in Figure 3.1.

Experts of the domain will manually control only a small subset of representative samples of every group of data.

The overall process followed when developing this semi-supervised learning model can be summarized in the following chart 3.2.

The aim of this study is therefore focused on the subsequent processes, in particular:

- Data pre-processing
- Smart Data computation
- Clustering
- Cluster validation
- Cluster characterization

- Result interpretation and valuable Knowledge extraction

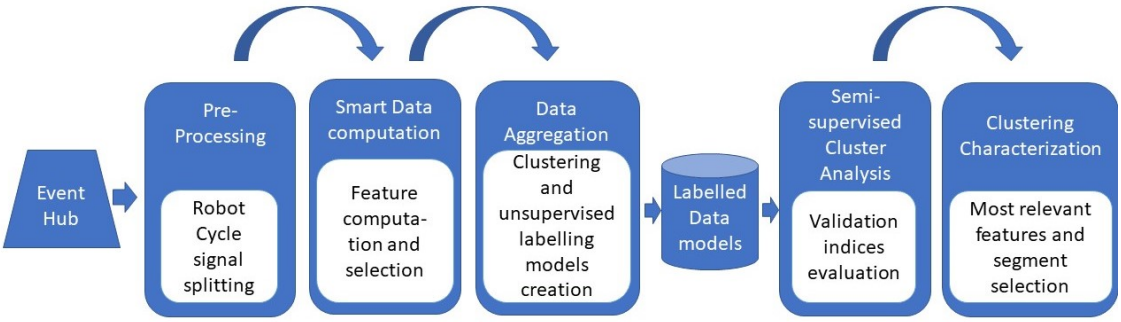


Figure 3.2: Semi-supervised learning architecture

For what concern data pre-processing and smart data computation, the work developed in [8] will be exploited and re-adapted to this use case analysed context. In particular, this two first blocks are demanded to detect outliers and noise, align cycles, compute features time series and select feature.

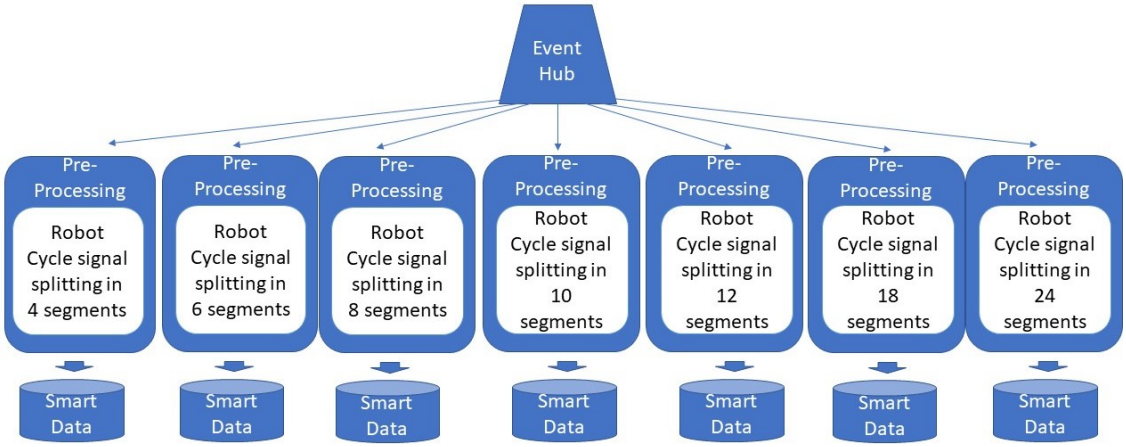


Figure 3.3: Pre-Processing architecture

Data pre-processing and Smart Data computation block are demanded to develop the task explained in section 2.1.3, they will be adapted from [8] for this specific approach: different starting splitting configuration will be evaluated and compared between each other.

For making things clear, diagram 3.3 illustrate the different configurations applied to pre-procesing block after having received the row data from the

event hub. For each pre-processed dataset, the Smart Data is consequently produced.

3.2 Clustering

In basic terms, the objective of clustering is to find different groups within the elements aggregating the data. To do so, clustering algorithms find the structure in the data so that elements of the same cluster (or group) are more similar to each other than to those from different clusters. In this work, many different algorithm has been applied, with the aim to evaluate the various properties of the dataset and detect the one that best fit the starting dataset.

For each previously Smart Data produced, clustering algorithms has been applied producing the correspondent labelled data models.

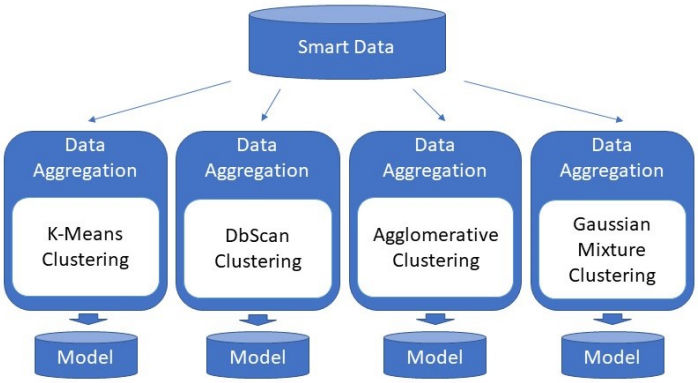


Figure 3.4: Clustering architecture

Since clustering algorithms taken into consideration are various, as will be possible to see in chapter 5 first of all a selection of a good algorithms and paramenters configuration will be executed, applying therefore less data aggregations to each Smart Data.

3.3 Clustering algorithms

In this section, a full list of the applied clustering algorithms is presented and explained, heightening their advantages and disadvantages.

3.3.1 K-means

K-Means [9] algorithms are extremely easy to implement and very efficient computationally speaking.

Those are the main reasons that explain why they are so popular. But they are not very good to identify classes when dealing with in groups that do not have a spherical distribution shape.

The K-Means algorithms aims to find and group in classes the data points that have high similarity between them. In the terms of the algorithm, this similarity is understood as the opposite of the distance between data points.

The closer the data points are, the more similar and more likely to belong to the same cluster they will be.

Key Concepts

- Squared Euclidean Distance

The most commonly used distance in K-Means is the squared Euclidean distance. An example of this distance between two points x and y in m-dimensional space is:

$$d(x, y)^2 = \sum_{j=1}^{m} (x_j - y_j)^2 = \|x - y\|_2^2$$

Here, j is the jth dimension (or feature column) of the sample points x and y.

- Cluster inertia or SSE

Cluster inertia is the name given to the Sum of Squared Errors within the clustering context, and is represented as follows:

$$SSE = \sum_{i=1}^{n} \sum_{j=1}^{k} w^{(i,j)} \left\| x^{(i)} - \mu^{(j)} \right\|_2^2 \tag{3.1}$$

Where μ(j) is the centroid for cluster j, and w(i,j) is equal to 1 if the sample x(i) is in cluster j, 0 otherwise. K-Means can be understood as an algorithm that will try to minimize the cluster inertia factor.

Algorithm Steps

1. First, we need to choose k, the number of clusters that we want to be generated
2. Then, the algorithm will select randomly the centroids of each cluster
3. It will be assigned each datapoint to the closest centroid (using euclidean distance)
4. It will be computed the cluster inertia
5. The new centroids will be calculated as the mean of the points that belong to the centroid of the previous step. In other words, by calculating the minimum quadratic error of the datapoints to the center of each cluster, moving the center towards that point
6. Back to step 3

K-Means parameters

- Number of clusters: The number of clusters and centroids to generate
- Maximum iterations: Of the algorithm for a single run
- Number initial: The number of times the algorithm will be run with different centroid seeds. The final result will be the best output of the number defined of consecutive runs, in terms of inertia

How to choose the right number of clusters

Selecting the right number of K (number of clusters) is one of the key points of the K-Means. To find it there are some methods:

- field knowledge
- business decision
- Elbow method

As being aligned with the Data Science methodology, the Elbow method is the preferred one, as it relies on a mathematical method backed with data, to choose a decision.

Elbow Method

The Elbow method is applied for choosing the best number of clusters in a data set. It works by plotting the ascending values of K versus the total error obtained when using that K.

$$\%Variance = \frac{Variance\ between\ groups}{Total\ Variance} \tag{3.2}$$

The objective is to select the K that for each cluster will not significantly increase the variance, that correspond to the obtained curve's elbow.

K-Means limitations

Although K-Means is a great clustering algorithm, it's most useful when the exact number of clusters is beforehand known and when dealing with spherical-shaped distributions.

3.3.2 DbScan

Density-Based Spatial Clustering of Applications with Noise, or DBSCAN [10], [11] , is another clustering algorithm specially useful to correctly identify noise in data.

DbScan assigning criteria

It is based on a number of points with a specified radius ϵ and there is a special label assigned to each data point. The process of assigning this label is the following:

- There is a specified number MinPts (minimum points) of neighbour points. A core point will be assigned if there is this MinPts number of points that fall in the ϵ radius
- A border point will fall in the ϵ radius of a core point, but will have less neighbors than the MinPts number
- Every other point will be noise points, the so called outlier

Algorithm steps

The algorithm follows this logic:

1. Identify a core point and create a group for each one, or for each group of connected core points (if they satisfy the criteria to be core point)

2. Identify and assign border points to their correspondent core points

3.3.3 Hierarchical Clustering

Hierarchical clustering [12] is a wide group of clustering algorithms that generate nested clusters by merging or splitting them. This clusters hierarchy is shown as a tree (or dendrogram).

The root of this tree is the unique cluster that contains all the samples, at the bottom, the leaves gather only a single sample.

Kinds of Hierarchical Clustering

There are two different approaches to this type of clustering:

- Divisive: this solution starts by collecting all data points in one single cluster. Then, the latter will be splitted iteratively into smaller ones until each one contains just a single sample

- Agglomerative: this other solution starts with each sample belonging to a different cluster and then merging them by the ones that are closer from each other until there is only one cluster, using therefore a bottom-up approach

In this study we will use only agglomerative ones. The linkage criteria determines the metric used for the merge strategy, in particular it can be single, complete, ward or average.

- Single Linkage

Single linkage starts by considering that each data point is a different cluster. After, it evaluate the distances between each pair of clusters and merge the cluster pair for which the computed distance is the smallest, putting together the most similar members.

- Complete Linkage

Although being really similar to the single linkage, its metric is exactly the opposite, comparing the most dissimilar data points of a cluster pair and performing the merge with the highest distance computed.

- Ward Linkage

Minimizes the sum of squared differences within all clusters. It is a variance-minimizing approach and in this sense is similar to the k-means but tackled with an agglomerative hierarchical approach.

- Average Linkage

It merge pair of cluster minimizing the average distance between points of the clusters

Advantages of Hierarchical Clustering

- The resulting hierarchical representations can be very informative
- Dendrograms provide an interesting and informative way of visualization
- They are specially powerful when the dataset contains real hierarchical relationships

Disadvantages of Hierarchical Clustering

- They are really sensitive to outliers and, with their presence, the obtained model performance decreases significantly
- They are really expensive, computationally speaking

3.3.4 Gaussian Mixture

A Gaussian mixture [12] model is a probabilistic model that consider all data points generated from a mixture of a finite number of Gaussian distributions with unknown parameters.

One can suppose mixture models as generalizing K-Means clustering to collect information about the covariance structure of the data as well as the centers of the latent Gaussians.

It implements the expectation-maximization (EM) algorithm, fitting mixture of Gaussian models. It can even draw confidence ellipsoids for multivariate models, and compute the Bayesian Information Criterion to evaluate the number of clusters in the data.

Advantages of Gaussian Mixture

- It is the lightest algorithm for generating mixture models
- As this algorithm only maximizes the likelihood, it will not bias the means towards zero, or the cluster sizes to have specific structures that may or may not apply

Disadvantages of Gaussian Mixture

- When not enough points per mixture are provided, evaluating the covariance matrices becomes difficult, and the Gaussian Mixture is known to diverge, finding solutions with infinite likelihood unless one regularize the covariances manually
- This algorithm will always apply all the components it has access to, needing information theoretical criteria to decide how many components to use in the absence of external cues

3.4 Clustering validation

Clustering validation is the process of evaluating the result of a cluster objectively and quantitatively. We will do this validation by applying cluster validation indices. There are two main categories, supervised (or external) indices, who use labeled data and unsupervised (or internal) indices, who use instead unlabeled data.

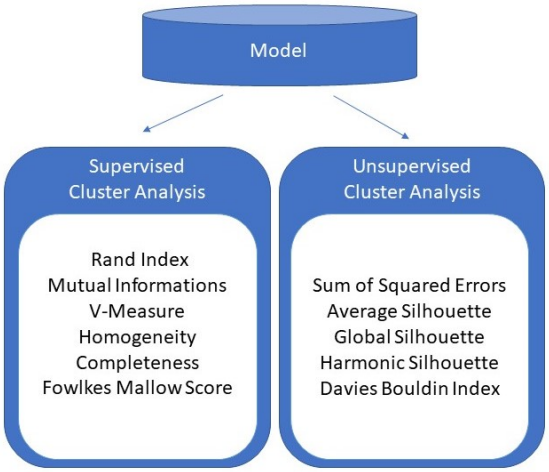


Figure 3.5: Clustering Validation architecture

Evaluating the quality of a clustering algorithm is not as simple as counting the amount of errors or the precision of a supervised classification algorithm.

Moreover, any evaluation metric shouldn't take the absolute values of the cluster labels into account but rather, if this clustering define a good separations of the data according to a ground truth set of classes or satisfying some internal assumption according to some similarity metric, like belonging to the same class means having more similar members than in a different classes.

As mentioned in the previous section, the clustering block produce a labelled data model, according to Figure 3.5, Clustering Validation is evaluated in that upcoming model from the preceding data aggregation block.

In the following subsections a complete list of supervised and unsupervised indices that has been evaluated on the data models, highlighting their positive and negative aspects.

3.4.1 Supervised indices

As mentioned before, this class of indices makes use of the ground truth (the original labelling), which not often is known. The clustering structure is so matched with before hands known informations. In some application is also possible to match clustering structure between them, making the following indices working in an unsupervised context. In this work are by the way applied in supervised mode only.

Adjusted Rand Index

The adjusted Rand index [13] is a metric that measures the similarity of two assignments, ignoring permutations and with chance normalization.

To understand it we should first define its components:

$$RandIndex = \frac{a+b}{\binom{n}{2}} \tag{3.3}$$

where:

- a : is the number of points that are in the same cluster both in C and in K
- b: is the number of points that are in the different cluster both in C and in K

- n: is the total number of samples

The adjusted Rand Index is instead defined as:

$$ARI = \frac{RI - Expected\ Index}{Max(RI) - Expected\ Index} \tag{3.4}$$

The ARI can get values ranging from -1 to 1. The higher the value, the better it matches the original data. A completely uniform random label assignment will score an ARI close to zero.

This index is really good to compare dissimilar clustering algorithm, since no assumption is made on the clustering structure.

Mutual Info

Like the Rand Index score, Mutual Information [14] is a function that evaluate the agreement of the two assignments, ignoring permutations. Two different versions are available, Normalized Mutual Information (NMI) and Adjusted Mutual Information (AMI). NMI is more used in the literature, while AMI was proposed more recently and is normalized against chance.

A good advantage of this metric is again being bounded between 0 and 1, a uniform random labelling with lead the mutual info to zero.

Assume two label assignments (of the same N objects), U and V. Their entropy is the amount of uncertainty for a partition set, defined by:

$$H(U) = -\sum_{i=1}^{|U|} P(i)log(P(i)) \tag{3.5}$$

$$H(V) = -\sum_{j=1}^{|V|} P'(j)log(P'(j)) \tag{3.6}$$

where P(i) and P(j) is the probability that an object picked at random from U and V falls into class U_i and V_i.

The mutual information (MI) between U and V is calculated by:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)log\left(\frac{P(i,j)}{P(i)P'(j)}\right) \tag{3.7}$$

Normalized Mutual Info

The Normalized version is expressed as:

$$NMI(U,V) = \frac{MI(U,V)}{mean(H(U),H(V))} \tag{3.8}$$

Since, like the normal Mutual Info metric is not adjusted for chance, its value will tend to increase as the K (number of cluster) increase.

Adjusted Mutual Info

To solve this latter issue, an expected value E is introduced [15]. Using the expected value, the adjusted mutual information can then be calculated using a really similar form to the adjusted Rand index previously mentioned:

$$AMI = \frac{MI - E[MI]}{mean(H(U),H(V)) - E[MI]} \tag{3.9}$$

Homogeneity, Completeness and V-mesaure

Those metrics are again bounded between 0 and 1 and no assumption are made on the cluster structure, this will come really helpful when comparing different metrics on different algorithm.

In particular they are defined as desirable objectives, using conditional entropy analysis:

- Homogeneity: each cluster contains only members of a single class

$$h = 1 - \frac{H(C|K)}{H(C)} \tag{3.10}$$

- Completeness: all members of a given class are assigned to the same cluster

$$c = 1 - \frac{H(K|C)}{H(K)} \tag{3.11}$$

where $H(C|K)$ is the conditional classes entropy given the cluster assignments, defined as:

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n}\cdot log\left(\frac{n_{c,k}}{n_k}\right) \tag{3.12}$$

and $H(C)$ is the classes entropy:

$$H(C) = -\sum_{c=1}^{|C|} \frac{n_c}{n} \cdot log\left(\frac{n_c}{n}\right) \tag{3.13}$$

in the end the V-measure [16] is simply define as the harmonic mean of the two above metrics:

$$v = 2 \cdot \frac{h \cdot c}{h + c} \tag{3.14}$$

Fowlkes-Mallows Score

The Fowlkes-Mallows [17] score FMI is defined as the geometric mean of the pairwise precision and recall:

$$FMI = \frac{TP}{\sqrt{(TP + FP)(TP + FN)}} \tag{3.15}$$

where:

- TP = true positive: the pair belonging to the same cluster before and after clustering
- FP = false positive: the pair belonging to the same cluster before clustering and in different cluster after
- FN = false negative: the pair belonging to the same cluster after clustering but in different ones before clustering

Again the advantages of this metric is being bounded between 0 and 1, with no assumption of the cluster structure.

With this metric finish the section of the used supervised indices in this work.

3.4.2 Unsupervised indices

In unsupervised learning, we will work with unlabeled data and this is when internal indices are more useful.

If the ground truth labels are not known, and this is the most frequent case in industrial environments, evaluation must be performed using the model itself.

Silhouette

One of the most used and common indices is the Silhouette Coefficient [18] and its derivations that are later exposed.

The Silhouette value for a single sample is defined as:

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)} \tag{3.16}$$

where:

- a: is the mean distance between a point and all other points in the same cluster
- b: is the mean distance between a point and all other points in the nearest cluster

Again also this coefficient is bounded, between -1 and 1, good for comparing clustering with different K, not being affected to number of cluster increasing.

In poor words it represent the good cohesion between points of the same cluster and the good separation between different cluster.

ASI - Average Silhouette Index

The ASI index is simply the average silhouette value of a specific label assignment between the silhouette values of all sample.

$$ASI = \frac{\sum_{i=1}^{|N|} s_i}{N} \tag{3.17}$$

where N is the data set cardinality

GSI - Global Silhouette Index

The GSI is an index that tend to give equal wight between all cluster evaluating the silhouette.

This, for giving relevance to local dataset behaviour that wouldn't be noticeable with only the ASI index.

For each generated cluster, the ASI is separately computed, and then, the average between each cluster ASI is finally calculated, getting in this way the GSI.

Simply:

$$GSI = \frac{\sum_{i=1}^{K} ASI_i}{K} \tag{3.18}$$

were K is the number of clusters.

HSI - Harmonic Silhouette Index

In the end, the HSI represent the average between the ASI and the GSI indices. As will be possible to see later, this will be in this work the most used index for comparing clustering techniques.

The advantages of this index are multiples:

- no need of ground truth knowledge
- being the mean between GSI and ASI put in evidence different aspect of the dataset
- bounded between 1 and -1
- value is higher when clusters are dense and well separated, which correspond to a standard concept of a cluster

One of its disadvantages is surely the heavy computational load needed for evaluating it.

$$HSI = \frac{ASI + GSI}{2} \tag{3.19}$$

In Figure 3.6 a comparison between HSI, GSI and ASI is shown, HSI obviously laid between the ASI and GSI, being their average value, both ASI and GSI highlight different properties and aspects, making the harmonic silhouette a more overall complete indicator despite the simple silhouette. GSI gives the same weight to all clusters independently from their cardinality, this is importance since small cluster may contains important informations that could be not visible in the normal silhouette..

Figure 3.6: HSI, GSI, ASI Comparison

Davies-Bouldin Index

This long but useful list of indices terminate here with this last unsupervised index. Like the previous one, it always evaluate the goodness of a labelling assignment, showing up proprieties such density and separation among clusters.

The Davies-Bouldin Index [19], [20], is defined as the mean similarity between clusters. A value close to zero represent so a better clustering.

$$DB = \frac{1}{K}\sum_{i=1}^{K} max_{i\neq j} R_{ij} \tag{3.20}$$

where:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \tag{3.21}$$

- s_i: is the mean distance between cluster points i and their centroid of that cluster – also know as cluster diameter

- d_{ij}: the distance between the cluster i and j centroids

The main advantage of this last index is surely its computing efficiency, much more lighter than silhouette.

3.5 Cluster characterization

Cluster characterization helps domain expert on simpler identifying data labelling to robot cycles, and in general to production cycles. How will be possible to see later in this work, each group will be locally characterized from *the 10 most relevant data features*, to support the domain attention through the most relevant features that more influence each group.

To this aim, a Decision Tree Classifier (better explained in 3.5.1) has been used, the top 10 used features in the tree classifier correspond to the most relevant features characterising the root-leaves path of the tree.

After selecting these most relevant features, their distribution has been shown through the use of *boxplots*, which will help to better recognise the most characterizing features of a cluster in terms of relevant properties and content.

That additional knowledge may help domain expert to better understand a specific meaning for every group. For humans would be impossible to manually inspect all samples, having so, a small representative group of features is an important support.

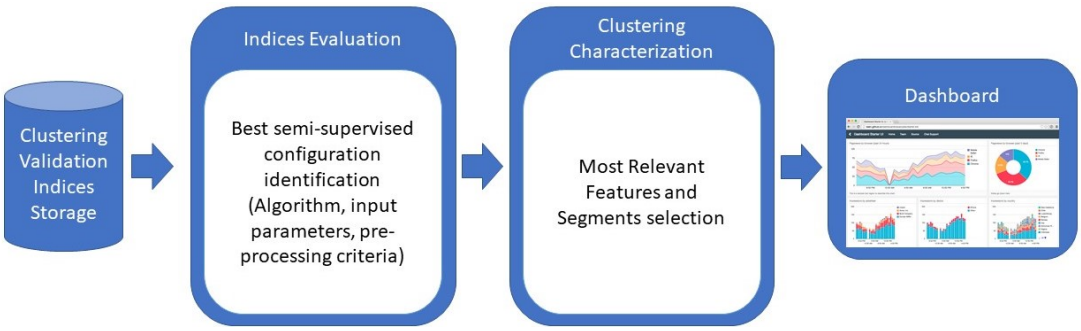


Figure 3.7: Clustering Characterization architecture

According to Figure 3.7, the characterization stage operate over the previously elaborated indices, with the final goal of generate useful output for a clear and representative dashboard.

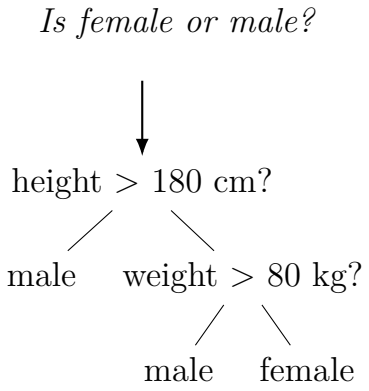
3.5.1 Decision Tree Classifier

A Decision Tree Classifier is a simple representation for classifying samples. It is a Supervised Machine Learning technique where the data is continuously splitted according to certain parameters.

Decision Tree consists of:

- Nodes: check the value of certain attribute by some condition;
- Edges: connection between outcome of a node to the next node;
- Leaf nodes: terminal nodes that predict the outcome

The objective of a classification tree is so to classify the input data. Given an input, it will be assigned to a specific class/label.



A Decision Tree Classifier example

Such a tree is created through a process known as binary recursive partitioning. This is an iterative process of splitting the data into partitions, and then again splitting it up further on each branch.

Chapter 4

Implementation details

In this chapter a fast recap of the main instruments and approaches I used to develop this work.

A first Hardware section followed from the Software section where I report the main code features developed.

4.1 Hardware

Since the clustering elaboration has been a heavy computational task, has been splitted and pipelined between different elaboration system. The heavier part has been processed with the support of Google Colab, leaving so the lighter computational part to a less powerful working station.

4.1.1 Google Colab

Colaboratory is a free service provided by Google, consist in a Jupyter notebook environment in the cloud that requires no setup.

With Colaboratory you can write and execute code, save and share your analyses for free from your browser. It provided high computational resources, in particular:

- 12 GB GDDR5 VRam Memory
- 360 GB Hard Disk
- GPU: 1xTesla K80 with 2496 CUDA cores
- CPU: Intel(R) Xeon(R) CPU @ 2.20GHz, 56MB cache memory, 1xsingle core hyper threaded

4.2 Software

Because of the long processing time needed to apply clustering algorithm and because of the large dimension of the dataset, especially in terms of feature, I decided to apply some improvement at the code with the aim to optimize it.

For what concern the structure of the code I have modulized it, bringing it at a really high number of splitted modules, where each of them produce a job and dump it, in a chain way with the other modules.

Thanks to this the whole process has been processed in separeted portions in parallel, launching each portion in a separated Google Colab session, with the possibility to restore the processing at the last successfully elaborated dump, and being able to continue the processing in local terminals later.

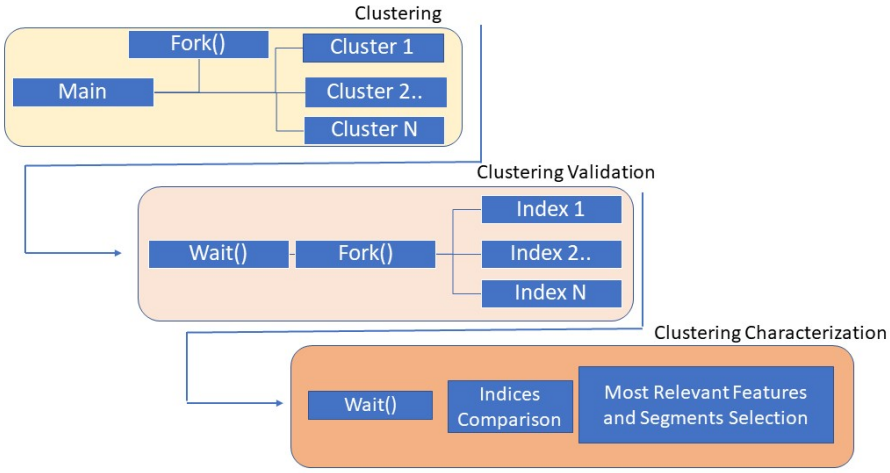


Figure 4.1: Software structure

I considered as a dump each single cluster configuration. After this step, each previous dump is evaluated with all cluster validation indices previously reported, and each index validation is an other dump, being possible to restart from each step without needing to reprocess all the previous steps.

Thanks to this, it has been possible to process the heavy part on Google Colab, and let the lighter processing part possible on a laptop.

In the end, a third block is able to identify for each algorithm and for each

validation index the first max local pick value, and later the best local pick value among all the algorithm, selecting in the end the best algorithm and input parameter.

Figure 4.1 show an illustration of the above discussed structure of the developed code.

4.2.1 PyCharm

PyCharm has been choosed as IDE (Integrated Development Environment) since provide high support for Python, the programming language choosed for this work. Moreover, is cross platform and the community edition is under the Apache Licence, therefore free to use.

4.2.2 Python 3.0

Python [21]is an extremely high level programming language that is easily available to the most disparate programming paradigms, from the object-oriented to the functional one.

It is a language that is simple to use and understand, since the code blocks are delimited by indentation, the increase or reduction of the indention lead to entering or leaving a given code block respectively.

Python, in a similar way to C #, is a managed language, which means that it also enjoys a greater intrinsic code robustness than other programming languages and an automatic memory management.

Python is also a language with dynamic typing of variables, which means that variables in a python script can be considered simply as names assigned to objects that reside in memory. The same object can be associated with multiple names and a name can be associated with different objects, changing type depending on the associated object.

Although Python is an interpreted language, therefore slightly slower than compiled languages, such as C or C ++, or semi-compiled, like C # and Java, it is possible to extend it with compiled code calls, where greater efficiency is needed, which makes Python an extremely versatile language.

Moreover Python has a very active community and a large number of additional modules to the language, making it an extremely powerful language, which I chose to use during this work due to the ease with which you can manage json files, csv files, datasets and much more.

4.2.3 Libraries

Since an high number of libraries has been used for developing this work, just a few selection are here shown since their help on speeding up the processes was fundamental.

Pandas

pandas[22] is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

- A fast and efficient DataFrame object for data manipulation with integrated indexing;
- Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
- Intelligent data alignment and integrated handling of missing data: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible reshaping and pivoting of data sets;
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets;
- Columns can be inserted and deleted from data structures for size mutability;
- Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets;
- High performance merging and joining of data sets;
- Hierarchical axis indexing provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;
- Time series-functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data;

- Highly optimized for performance, with critical code paths written in Python or C.

- Python with pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

Json

A usefull library for converting, reading, writing, loading, econding, decoding and formatting json [23] file in easy way. Since the original provided dataset was a json file I decided to use this helpfull library.

MatPlotLib

All the graphics shown on this work are generated with this efficient and easy to use library. MatPlotLib [24] provide several 2D plotting solution for all the scientific needs.

NumPy

NumPy [25] is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object

- sophisticated (broadcasting) functions

NumPy can be used also as a container of generic data-type. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

SkLearn

This is one of the most used libraries for this work. SkLearn [26], [12], provides all the algorithm used for clustering, cluster validation, cluster characterization, feature selections and much more. It is completely compatible with other libraries such NumPy and MatPlotLib and provide many useful API [27].

JobLib

Joblib [28]is a set of tools to provide lightweight pipelining in Python. In particular:

- transparent disk-caching of functions and lazy re-evaluation (memoize pattern)
- easy simple parallel computing
- Avoid computing the same thing twice
- Persist to disk transparently

Joblib is optimized to be fast and robust on large data in particular and has specific optimizations for numpy arrays. It is BSD-licensed.

Chapter 5

Experimental results

The experimental results chapter is organized in the following manner. A first section evaluates the benefits and goodness of the proposed semi-supervised data labelling methodologies in section 3.2, comparing them with the different metrics previously illustrated (section 3.4), in respect of the state-of-the-art approach. Therefore, the objective of this section will be to identify the best configuration, in terms of algorithm and algorithm inputs that better suite the starting dataset.

Clustering algorithms has been applied, as already mentioned with different parameters, in particular:
for K-Means, Agglomerative ones and Gaussian mixture:

- K = [2, 30]: the clusters number has been processed for all values between 2 and 30;
- n_init: each configuration has been tested for 10 iterations, and the best run in terms of inertia has been selected and saved;
- max_iter: the maximum number of algorithm iterations for a single run;

for DbScan:

- ϵ: a clustering solution has been processed of each eps radius value in range [1,30];
- MinPts: for each ϵ value, values (50, 100, 150, 200, 250) has been computed;

Later, in the next section, different pre-processing of the initial Smart Data will be analysed according to previously discussed section 2.1.3, with the aim to recognize a possibly lighter starting data set with no relevant negative effect in validation terms.

Finally, a last section, once identified the best configurations, is demanded to present the characterization of the previously obtained clusters, again according to already discussed principle in section 3.5. A most relevant feature selection will be presented, and the same will be done with most relevant segments too.

For formatting and layout reason, some indices will be referred using initials, in particular:

- AMI : Adjusted Mutual Info
- MI : Mutual Info
- NMI : Normalized Mutual Info
- FMS : Fowlkes Mallows Score
- ASI : Average Silhouette Index
- GSI : Global Silhouette Index
- HSI : Harmonic Silhouette Index

5.1 Clustering evaluations

5.1.1 The Elbow Method

One of the first evaluation is the Elbow method, useful to identify the most appropriate K value or K range on which compute clustering. As already discussed in section 3.2, the goal is to find the proper k that for each cluster will not rise significantly the variance.

$$\%Variance = \frac{Variance\ between\ groups}{Total\ Variance} \tag{5.1}$$

To this aim, the sum of squared error has been plotted using the K-means algorithm on the original dataset, for all K values in range [0, 50].

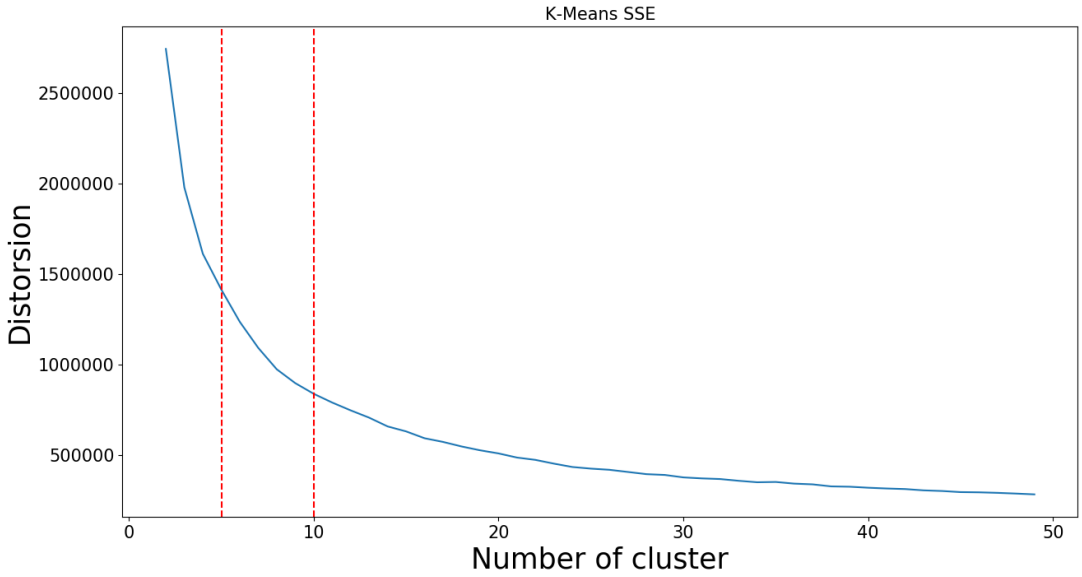


Figure 5.1: Sum of squared error on K-Means over K values

In this case, as is possible to see from Figure 5.1, values in interval [5,10] contains the curve's elbow and represent therefore a good choose for K, by the way, clustering will be computed up to greater values for further analysis. Instead, from higher k value than 30, no interesting information can be obtained.

From these observation we can consider, as beforehand mentioned, [2,30] a good range for computing next clustering analysis.

5.1.2 Supervised and Unsupervised indices comparison

Therefore, after having computed all the validation indices previously explained in section 3.4, since a sighting comparison was difficult due to the high amount of graphs and data, a validation block has been developed.

The validation block, is able for each computed index to extract the first local max value and its corresponding configuration input value (k or ϵ) shown in brackets close to it.

In the next tables, all the first local max values are shown, separately for algorithm that use k and ϵ and separated by supervised and unsupervised indices.

Index	k-Means	Aggl-wrd	Aggl-sgl	Aggl-cpt	Aggl-avg	Gauss-mix
Rand Index	0.169(5)	0.231(5)	0.001(7)	0.032(7)	0.088(7)	0.364(5)
AMI	0.233(5)	0.319(5)	0.01(7)	0.128(6)	0.198(7)	0.46(5)
MI	0.407(7)	0.528(7)	0.002(7)	0.170(7)	0.197(7)	0.704(6)
NMI	0.308(5)	0.422(5)	0.004(7)	0.154(6)	0.207(7)	0.592(5)
V-measure	0.297(5)	0.406(5)	0.005(7)	0.152(6)	0.207(7)	0.574(5)
Completeness	0.233(5)	0.319(5)	0.098(3)	0.128(6)	0.199(7)	0.460(5)
FMS	0.449(3)	0.489(5)	0.682(3)	0.491(4)	0.671(4)	0.595(5)
Homogeneity	0.445(7)	0.578(7)	0.002(7)	0.186(7)	0.215(7)	0.770(6)

Table 5.1: First local max pick for each algorithm in k for the various supervised indices

Index	k-Means	Aggl-wrd	Aggl-sgl	Aggl-cpt	Aggl-avg	Gauss-mix
ASI	0.302(3)	0.249(5)	0.601(3)	0.208(6)	0.266(3)	0.213(3)
GSI	0.305(3)	0.279(3)	0.2(3)	0.264(3)	0.292(3)	0.219(3)
HSI	0.303(3)	0.263(3)	0.401(3)	0.215(3)	0.279(3)	0.216(3)
Davies	1.107(5)	1.092(3)	0.247(3)	1.075(4)	0.603(4)	1.755(3)

Table 5.2: First local max pick for each algorithm in k for the various unsupervised indices

Index	DbScan-50	DbScan-100	DbScan-150	DbScan-200	DbScan-250
Rand Index	0.276(11)	0.258(12)	0.259(13)	0.264(14)	0.248(5)
AMI	0.367(11)	0.349(12)	0.348(13)	0.353(14)	0.336(5)
MI	0.497(11)	0.48(12)	0.476(13)	0.478(14)	0.445(7)
NMI	0.447(11)	0.428(12)	0.426(13)	0.429(14)	0.405(5)
V-measure	0.439(11)	0.419(12)	0.417(13)	0.421(14)	0.398(5)
Completeness	0.368(11)	0.349(12)	0.348(13)	0.353(14)	0.337(5)
FMS	0.682(29)	0.682(29)	0.682(29)	0.682(9)	0.682(3)
Homogeneity	0.544(11)	0.525(12)	0.521(13)	0.522(14)	0.487(7)

Table 5.3: First local max pick for each algorithm in eps for the various supervised indices

Index	DbScan-50	DbScan-100	DbScan-150	DbScan-200	DbScan-250
ASI	0.135(11)	0.192(13)	0.199(14)	0.21(15)	0.187(3)
GSI	0.58(8)	0.666(9)	0.648(10)	0.771(10)	0.67(3)
HSI	0.222(11)	0.259(13)	0.266(14)	0.272(15)	0.27(3)
Davies	1.739(8)	1.983(9)	1.962(10)	1.836(10)	1.949(5)

Table 5.4: First local max pick for each algorithm in eps for the various unsupervised indices

For what concern DbScan, an important properties is the presence and amount of outliers in the clusters produced. As explained in section 3.2, outliers are the data points that doesn't satisfy the density paradigms of the DbScan algorithm, representing potential noise or anomalous samples.

MinPts	Epsilon	Number of Clusters	Rand Index	Outliers
50	11	3	0.2759	9374
100	12	5	0.2580	11031
150	13	4	0.2587	10583
200	14	3	0.2639	9585
250	14	3	0.2483	12226

Table 5.5: Outliers with various DbScan configurations

Table 5.5 report the outliers amount for DbScan clusters produced with the correspondent parameters. In this table are reported just the best solutions found with DbScan (evaluated on Rand Index, which is the best performed index from the latter), but even in this case, outliers represent a significant percentage of the original dataset, about 30%. This can be an input parameters problem or DbScan can not be suitable to issue this task.

As a further investigation, a comparison between DbScan SSE values and a non DbScan algorithm is shown in Figure 5.2.

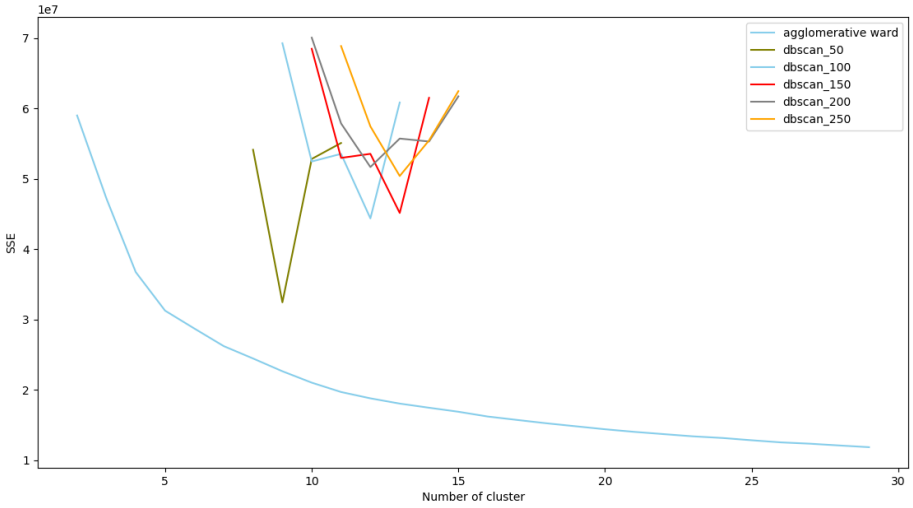


Figure 5.2: SSE Comparison on DbScan

Comparing algorithms K-driven with algorithms ϵ-driven is not simple. For giving an idea of the results, in 5.2 the SSE values of all clusters obtained with DbScan is compared with clusters obtained with an Agglomerative algorithms and is evident how them all laid above the Agglomerative curve, corresponding so to an higher SSE and a worst solution.

The next step that the validation block is demanded to compute, is to select from the previously generated tables, the best algorithm (in terms of first local max pick value) for each proposed index and of course, its value and configuration inputs. The first table is dedicated to supervised indices and the second to unsupervised ones.

Index	Value	Algorithm	k/eps
Rand Index	0.3643	Gaussian Mixture	5
AMI	0.4604	Gaussian Mixture	5
MI	0.7045	Gaussian Mxture	6
NMI	0.5927	Gaussian Mixture	5
V-mesaure	0.5743	Gaussian Mixture	5
completeness	0.4605	Gaussian Mixture	5
Fowlkes Mallows score	0.6821	DbScan (250MinPts)	10
Homogeneity	0.7701	Gaussian Mixture	6

Table 5.6: Supervised indices algorithm comparison

Index	Value	Algorithm	k/eps
ASI	0.6014	Agglomerative single	3
GSI	0.7713	DbScan (200MinPts)	10
HSI	0.4009	Agglomerative single	3
Davies Bouldin Index	0.2469	Agglomerative single	3

Table 5.7: Unsupervised indices algorithm comparison

From a first look at Table 5.6 and Table 5.7, two winners emerge, in particular, Gaussian Mixture seems to lead the supervised indices selection and Agglomerative (with "single" linkage) leads on the unsupervised ones.

For a better inspection the next figures show a comparison between the various clustering algorithms for the most representative index in both learning categories.

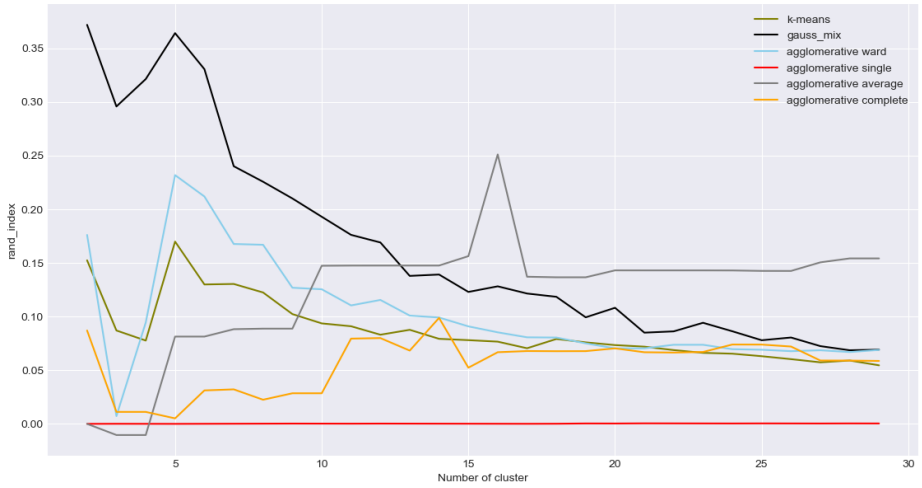


Figure 5.3: Rand Index Comparison

From the above Figure 5.3, where is shown a comparison between the Rand Index evaluated among the different clustering, is highly noticeable how the Gaussian Mixture is predominant in the K range of interest: [5-10].

This clustering adopted, present the same behaviour among whole supervised indices, seeming the most suitable for the specific used original dataset. But, having a look at next Figure 5.4, where a comparison between harmonic silhouette among all clustering techniques is shown, the Gaussian Mixture

generated clusters produce the worst values, seeming the less appropriate algorithm taking unsupervised validations into account.

Again, considering the Agglomerative (with single linkage), which is the best selected algorithm between unsupervised indices from Table 5.7 by the validation block, shows the same behaviour of the Gaussian Mixture, but in the opposite sense. Good results among all unsupervised indices, while, in this case, extremely negative results in supervised context.

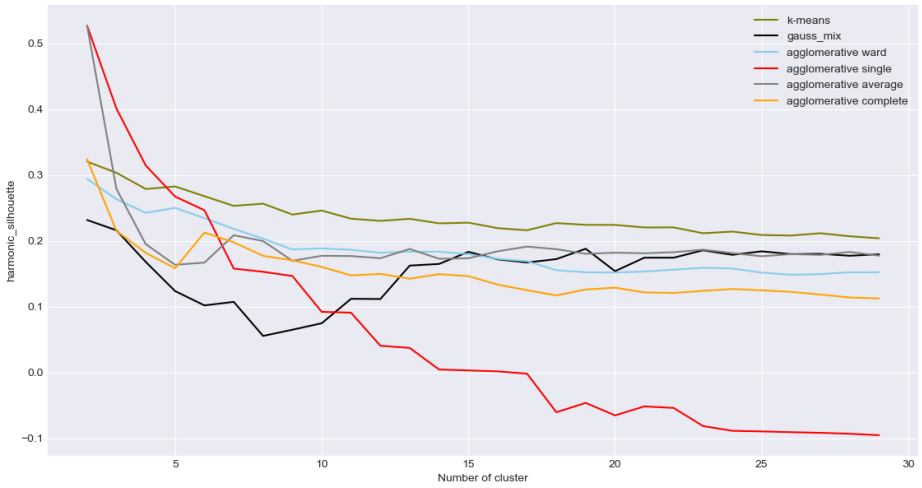


Figure 5.4: Harmonic Silhouette Comparison

Since the objective of this work is to support predictive maintenance models, with the aim to find solutions that apply in as much as possible industrial cases, it's important to take into account the behaviour in different conditions and not simply finding a single local context were a specific configuration produce the best solution but with not appreciable results overall.

In generic predictive maintenance context, the model need to auto detect degeneration, exclude anomalous inputs, and auto-trigger model updates. This means that parameters and configurations of the model can change over the time in a certain range, where the model need to produce stable and consistent evaluations.

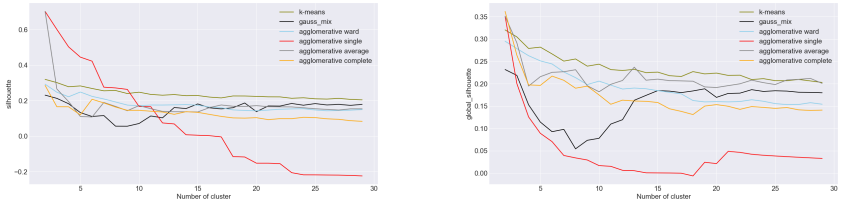
For instance, the Agglomerative (single linkage) clustering algorithm has been selected by the validation block as the best one based on unsupervised indices, since his first local pick is the highest between all the proposed solutions. According by the selection this value is in correspondence to K =

3.

Supposing that at a certain point in the time, new unlabeled data income in the dataset highly degrading the model in short time, the new production samples, if not recognized as anomalous cases, will be assigned to new properly created labels, and, as a consequence, the model will be re-trained.

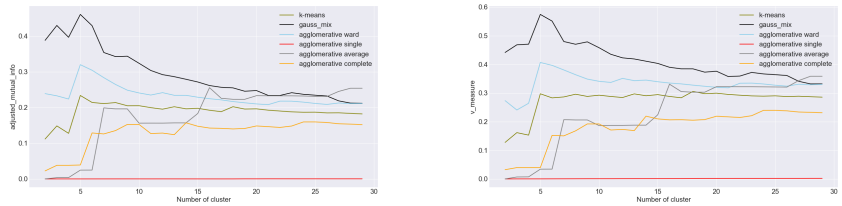
Considering for example K = 5 as new parameter of the model after the auto-triggered retraining, using the Agglomerative solution will lead to highly decreased validation values as it is possible to see from Figure 5.4. The Agglomerative with single linkage is inherently affected from number of cluster increasing.

Each solution and configuration brings its own properties, that need to be balanced and taken into account for the various conditions and contexts. Instead of considering just the best local maximum pick value, a better inspection for configuration selection would be to properly balance the overall behaviour.



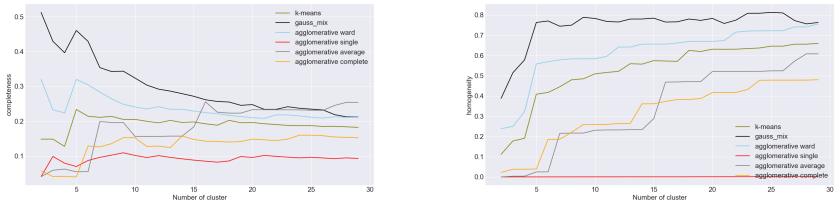
(a) ASI comparison (b) GSI comparison

Figure 5.5: ASI and GSI comparison



(a) AMI comparison (b) V-Measure comparison

Figure 5.6: AMI and V-Measure comparison

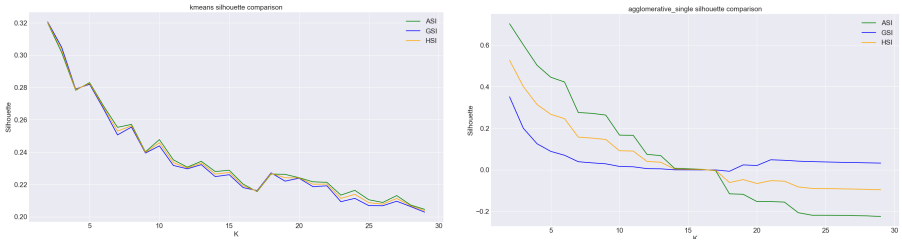


(a) Completeness comparison (b) Homogeneity comparison

Figure 5.7: Completeness and Homogeneity comparison

In the Figures 5.5,5.6,5.7, a fast overview of other different plotted indices is shown.

Since in a real-industry use-case labeled data is rare and not a common situation, unsupervised indices must be taken into account with more weight for selecting the best configuration. Comparing algorithms among all indices in the figures above, is evident how K-Means produce stable results (over parameters range of interest) and high values in the average, in both supervised and unsupervised indices, and in the latter ones, the best average values.



(a) K-Means Silhouette comparison (b) Agglomerative-single comparison

Figure 5.8: ASI, GSI and HSI comparisons between K-Means and Agglomerative-single

As a further inspection, Figure 5.8 shows a comparison of the different silhouette trends, evidencing how the two different clustering solutions produce different group formations. In particular, is clear how the Agglomerative solution is highly affected from number of cluster increasing, Instead of K-Means which offer a stable solution over different parameters, and groups really well distributed.

In the next analysis section, different dataset pre-processing will be evaluated, therefore, the amount of computation proportionally increase at the

square of number of dataset configurations. From this point just the K-Means and the Agglomerative-single (for further analysis) will be considered.

For each different dataset pre-processing configuration, clustering has been re-executed and all the indices re-computed and plotted.

5.1.3 Different dataset pre-processing evaluation

As already partially discussed in section 2.1.3 , all new incoming production cycles signals has been splitted into partitions, and each single split has been characterized by several statistical features. Therefore, an higher splitting configurations has been applied for better comparing and validating this phase.

Since for every split an big number of features has been generated (i.e. std, mean, Kurtosis, quartiles, skewness) up to almost 350, a feature selection approach has been applied. Exploiting the Pearson index correlation, the features amount is decreased according to a certain threshold.

From this step, quality and performance of the next model is highly affected, hence the need to try different configuration to compare. Also, since the features amount generated depend on the number of splits (being the total amount equal to the sum of features of each split), reducing this parameter lead to a lower total features amount and lower computing complexity as an obviously natural consequence.

In [8], a solution taking into account 24 splits is proposed. In this work, different splitting configuration will be proposed and compared, again, with the objective of having lighter computation workload with not considerable quality looses.

Number of Split	# of features	[8] Features %
4	30	12,82 %
6	50	21,35 %
8	63	26,92 %
10	87	37,17 %
12	110	47,00 %
18	172	73,50 %
24	234	100 %

Table 5.8: Features distribution over different splittings criteria

In the upper Table 5.8, the correspondence between splits number, amount of features produced and percentage despite [8] solution is shown.

Being clustering algorithm computation highly related to features amount, is evident how choosing a lower number of splits can benefit the overall computation workload.

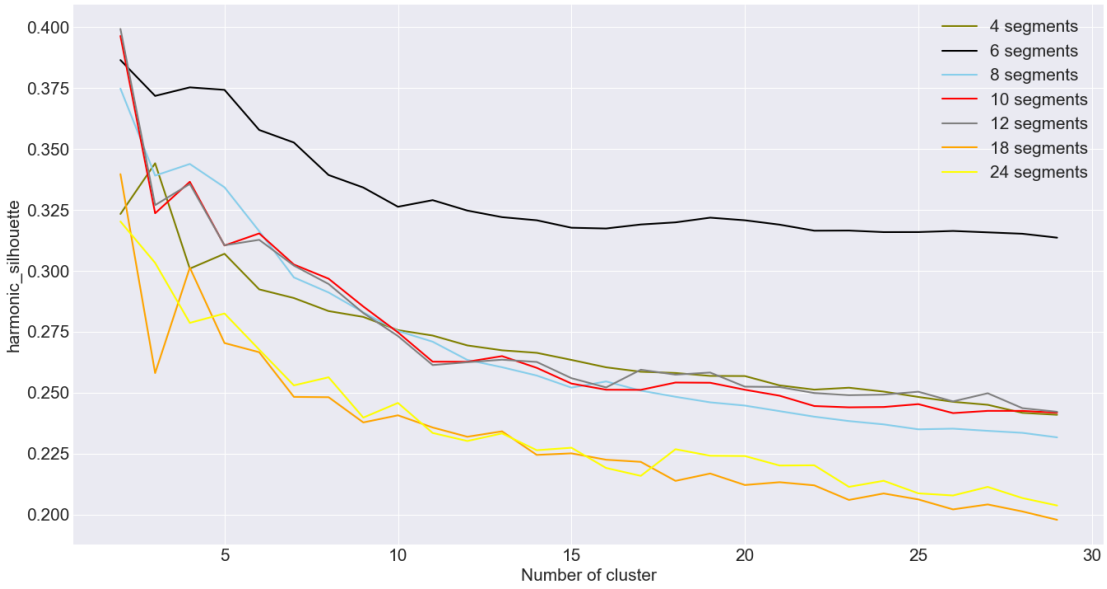
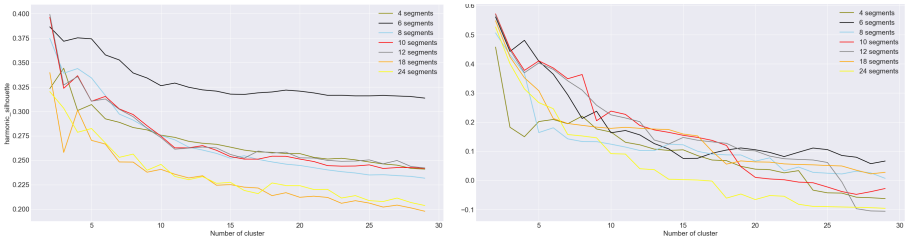


Figure 5.9: HSI over K-Means comparison between different splitting

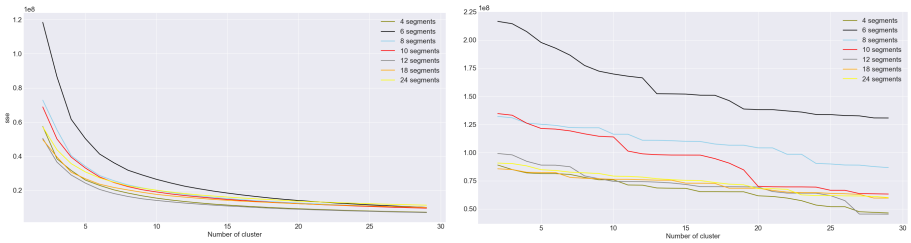
After having computed clustering (K-Means and Agglomerative-single) and validation indices over all the different pre-processed dataset, graphics like the one presented in the above Figure 5.9 has been plotted.



(a) K-Means HSI comparison over splits

(b) Aggl-single HSI comparison over splits

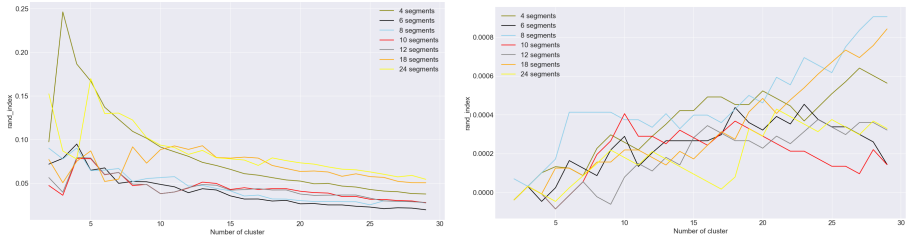
Figure 5.10: HSI over K-Means and Agglomerative for the different splitting



(a) K-Means SSE comparison over splits

(b) Aggl-single SSE comparison over splits

Figure 5.11: SSE over K-Means and Agglomerative for the different splitting



(a) K-Means Rand Index comparison over splits

(b) Aggl-single Rand Index comparison over splits

Figure 5.12: Rand Index over K-Means and Agglomerative for the different splitting

From the above Figures 5.10, 5.11, 5.12, a first comparison between K-Means and Agglomerative-single put in evidence the better results, in terms of stability over different parameters and absolute values, obtained by K-Means solution.

Moreover, is noticeable how the different splittings criteria doesn't affect negatively the results obtained. In particular, for what concern a number of splits equal to 6, in the HSI comparison seems to give even better results, a bit lower results in SSE comparison and an average results in the Rand Index comparison.

In the end, summing up all the results previously taken into consideration, a possible good configuration, suitable for different parameters, and with a good trade-off between computation workload and indices result quality can be:

- Algorithm: K-Means
- K: 5

- Number of splits = 6

5.1.4 Clustering characterization evaluations

As already partially explained in section 3.5 cluster characterization helps domain expert on simpler identifying data labelling to robot cycles, and in general to production cycles. How will be possible to see later in this work, each group will be locally characterized from *the 10 most relevant data features*, to support the domain attention through the most relevant features that more influence each group.

To this aim, a Decision Tree Classifier (better explained in 3.5.1) has been used, the top 10 used features in the tree classifier correspond to the most relevant features characterising the root-leaves path of the tree.

After selecting these most relevant features, their distribution has been shown through the use of *boxplots*, wich will help to better recognise the most characterizing features of a cluster in terms of relevant properties and content.

That additional knowledge may help domain expert to better understand a specific meaning for every group. For humans would be impossible to manually inspect all samples, having so, a small representative group of features is an important support.

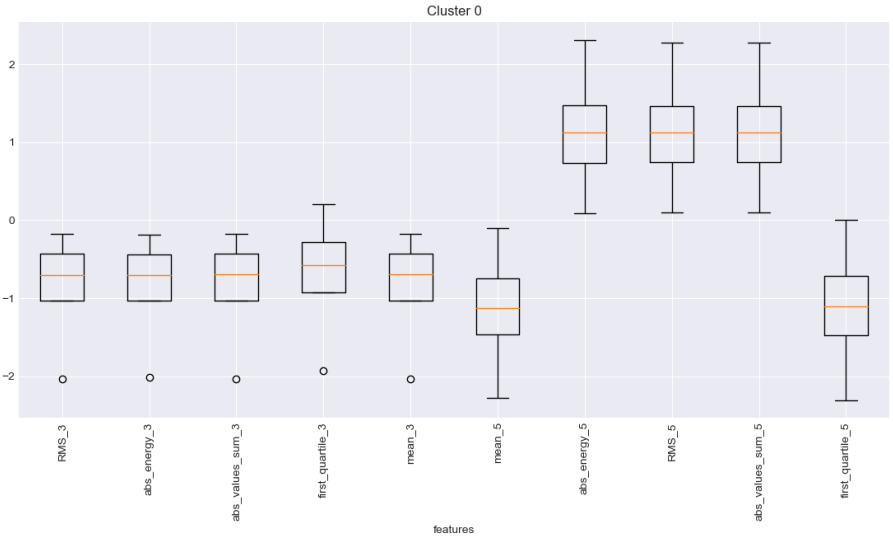
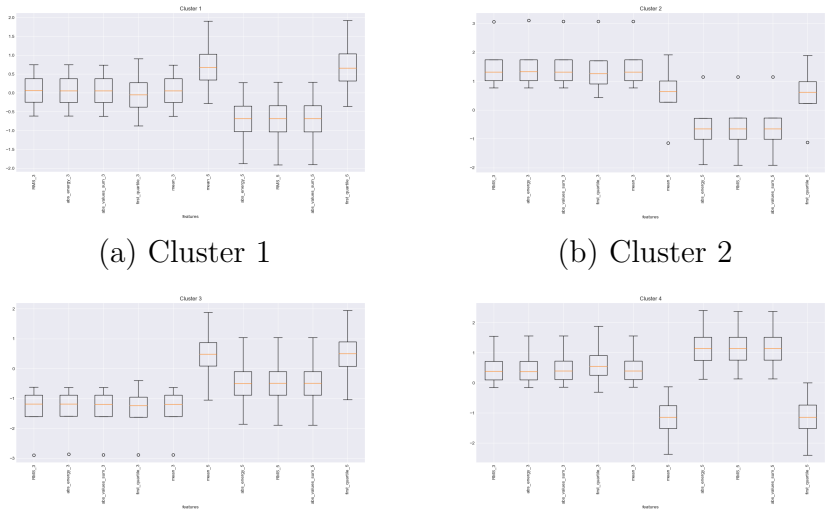


Figure 5.13: Cluster 0



(a) Cluster 1

(b) Cluster 2

(c) Cluster 3

(d) Cluster 4

Figure 5.14: Top-10 relevant features data distribution separately for each cluster. The description of the feature reported on the x axis is in the format: <feature_name>_<segment_id>

Figure 5.13, 5.14 shows boxplot of the most peculiar selected features (listed for each plot in the x axis) for each cluster. From the above boxplots is possible to observe how clusters are well cohesive and well separated.

In the feature name is also signed the correspondent belonging segment, hence, the selection of the most relevant segments. In poor words, these segments are mainly affected, in terms of belt tensioning, by the top 10 features.

Looking at figure 5.14, the most relevant segments correspond to n = 3,5.

Finally, figure 5.15 shows the original electric signal of a single production cycle splitted into 6 segments with segments 3 and 5 highlighted in red.

The engine axis is positioned parallel to the floor, and the production cycle is set as follow. In the start, the initial angle engine position is -500 degrees, after, at 20 % of max speed reaches +90 degrees and it maintains that position for 5 seconds. Then, it return back to -500 degrees at maximum speed. In the last phase it keep the position for other 5 seconds.

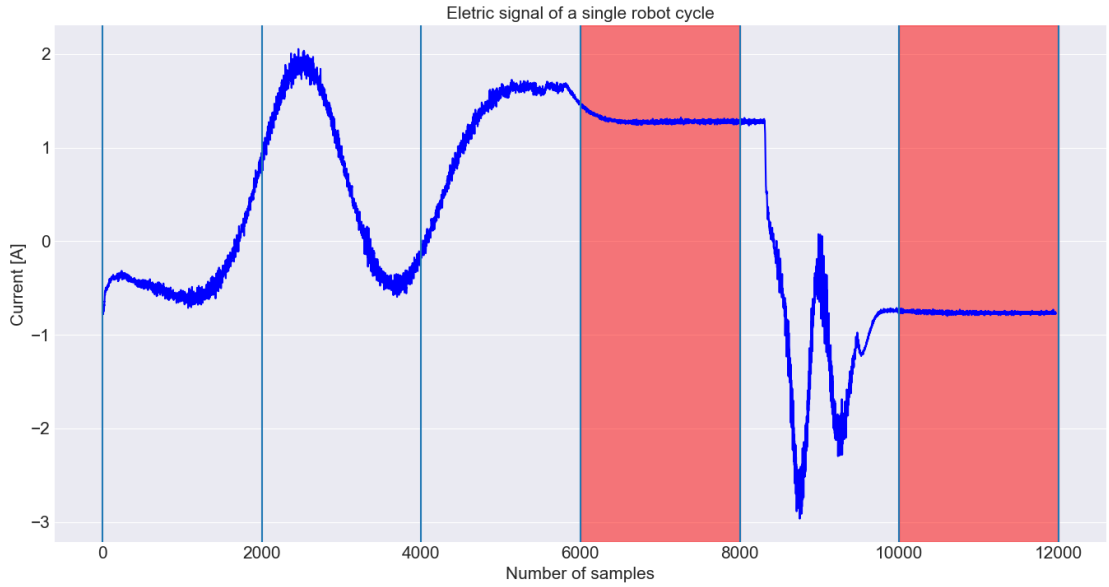


Figure 5.15: Electric signal with relevant segments highlighted

Figure 5.15 represent the current consumption during a single engine cycle, highlighting the most relevant production cycles segments in the semi-supervised data labelling block. The selected segments 3 and 5 by the block correspond to the electricity consumed when the motor reaches and maintains +90 degrees before and -500 degrees after a fast return back to start position.

Chapter 6

Conclusion and future work

This presented work introduce a supporting solution to predictive maintenance challenges, which, as already deeply discussed in previous chapters, represent a big task still mainly to be discovered and exploited for the Industrial 4.0 benefits.

The objective here was to help the parameters and configurations choose in the semi-supervised context, in both clustering and data pre-processing task, since represent an important block that is able to influence the rest of the entire predictive-model block.

Experiments have highlighted the most suitable algorithms and parameters for this industrial context, pointing out how some solution were giving local better solution but with high variance in other and not predicted conditions (e.g. Agglomerative Single) and other were giving slightly lower local picks values, but a more stable trend in possibly different conditions (e.g. K-Means).

Moreover, the comparison between different data pre-processing shown how a different splitting criteria from [8], in particular with a lower splits amount, brings to a way lower computational workload related to lower features amount and not considerable negative affects in term of validations.

Future works of this research surely include the predictive model itself, being able to identify when new incoming unlabeled data doesn't fit anymore the trained model with the initial distribution, triggering in this way a model retraining with different paramenters. Also, being able to recognize whether

new input data causing important model degradation can be assigned to a new on purpose created label, or they correspond to anomalous production cycle that need to be excluded.

Moreover, degradation of model performance is also a task that need to be implemented, being able to trigger model updates in different scenarios and contexts, with the aim to generalize the model in as much as possible Industry 4.0 cases.

As already deeply and widely discussed in section 2.1, predictive maintenance has always been a difficult problem in real word modern industries. With the new Industry 4.0, the manufacturing environments are evolving in digital factories and this context produce enormous volumes of raw data.

Leading the data will help managers on making better-informed business decision, improving production processes and bringing advantages to who will lead this important knowledge.

Bibliography

[1] Industrie 4.0 Working Group. "Recommendations for implementing the strategic initiative Industrie 4.0". In: (2013).

[2] V.Vyatkin Z.Salcic P.S.Roop and J.Fitzgerald. "Now that's smart!" In: 1.4 (2007), 17–29.

[3] S.Yang J.Lee H. D. Ardakani and B.Bagheri. "Industrial big data analytics and cyber-physical systems for future maintenance service innovation". In: *Procedia CIRP* 38 (2015), pp. 3–7.

[4] Gaurav and S.Bari G. M.D Silva A.Khan. "Real-time processing of iot events with historic data using apache kafka and apache spark with dashing framework". In: *2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)* (2017), pp. 1804–1809.

[5] B.Xu and S.A.Kumar. "Big data analytics framework for system health monitoring". In: *IEEE International Congress on Big Data* (2015), pp. 401 –408.

[6] T.Cerquitelli A.Macii E.Macii M.Poncino D.Apiletti C.Barberis and F.Ventura. "iSTEP: an integrated Self-Tuning Engine for Predictive maintenance in industry 4.0". In: *16th IEEE International Symposium on Parallel and Distribuited Processin with Applications, ISPA-18 Melbourne, Australia* (2018), p. 8.

[7] A.Conde S. Charramendita M.Canizo E.Onieva and S.Trujillo. "Real-time predictive maintenance for wind turbines using big data frameworks". In: *IEEE International Conference on Data Mining* (2017), pp. 70–77.

[8] D. Apiletti T. Cerquitelli F. Ventura S. Proto. "A new unsupervised predictive-model self-assessment approach that SCALEs". In: (2018).

[9] David Arthur and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics". In: (2007).

[10] H. P. Kriegel J. Sander Ester M. and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press* (1996), pp. 226–231.

[11] Sander J. Ester M. Kriegel H. P. Xu X. Schubert E. "DBSCAN revisited, revisited: why and how you should (still) use DBSCAN". In: *In ACM Transactions on Database Systems (TODS), 42(3), 19* (2017).

[12] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[13] L. Hubert and P. Arabie. "Comparing partitions". In: *Journal of classification 1985* (1985).

[14] Alexander Sthrel and Joydeep Gosh. "Cluster ensembles - a knowledge reuse framework for combining multiple partitions". In: (2002), pp. 583–617.

[15] Nguyen Xuan Vinh, Julien Epps, and James Bailey. "Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary?" In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: ACM, 2009, pp. 1073–1080. ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553511. URL: http://doi.acm.org/10.1145/1553374.1553511.

[16] Anrew Rosenberg and Julia Hirshberg. "V-Measure: A conditional entropy-based external cluster evaluation measure". In: (2007).

[17] E. B. Fowkles and C. L. Mallows. "A method for comparing two hierarchical clusterings". In: *Journal of the American Statistical Association* (1983). DOI: http://wildfire.stat.ucla.edu/pdflibrary/fowlkes.pdf.

[18] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53 –65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. URL: http://www.sciencedirect.com/science/article/pii/0377042787901257.

[19] D. L. Davies and D. W. Bouldin. "A Cluster Separation Measure". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2 (1979), pp. 224–227. DOI: 10.1109/TPAMI.1979.4766909.

[20] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. "On Clustering Validation Techniques". In: *Journal of Intelligent Information Systems* 17.2 (2001), pp. 107–145. ISSN: 1573-7675. DOI: 10.1023/A:1012801612483. URL: https://doi.org/10.1023/A:1012801612483.

[21] Python Software Foundation. *Python Language Reference, version 3.5*. URL: https://docs.python.org/3.5/.

[22] *Python Data Analysis Library*. URL: https://pandas.pydata.org/.

[23] *JSON encoder and decoder*. URL: https://docs.python.org/3/library/json.html.

[24] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[25] *NumPy.org*. URL: https://numpy.org/.

[26] *SciKit-learn, Machine Learning in Python*. URL: https://scikit-learn.org/stable/.

[27] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[28] JobLib Developers. *Joblib: running Python functions as pipeline jobs*. URL: https://joblib.readthedocs.io/en/latest/.