University of Padua

Department of Information Engineering

Master Thesis

in

Control System's Engineering

Machine Learning approaches for Anomaly Detection in Industrial IoT scenarios

---

Author:
Enrico Convento

Supervisor:
Prof. Gian Antonio Susto

Co-Supervisor:
PhD. Chiara Masiero

Academic Year: 2022/2023

Oct 3, 2022

Abstract of thesis entitled

Machine Learning approaches for Anomaly Detection in Industrial IoT scenarios

Submitted by

Enrico CONVENTO

for the degree of Master in Control System Engineering

at The University of Padua

in September, 2022

Machine learning has become a part of our daily life and is commonly used across a wide range of industries. These methodologies have been applied in countless areas of application and their use is in continuous expansion. In particular, these approaches play a key role in enabling Industry 4.0 and IoT scenarios. However, many of the algorithm results cannot be understood and explained in terms of how and why a specific decision was made. With the advancement of machine learning research, several techniques and approaches have emerged in recent years, but only a few studies have been produced regarding the end-user perspective. Therefore, the lack of interpretability in this technology is the biggest obstacle to the spread of these applications. Anomaly detection is a large field of machine learning technology that has enormous applicability in industrial scenarios. In fact, it is extremely relevant for the purposes of quality monitoring, predictive prevention and much more. Furthermore, the strength of this type of approach is that it can be implemented without the need for tagged data and obviously in this type of framework, where the data is often "dirty", it is very peculiar not to have labelled data. Obviously, this last application is also affected by the interpretability problem that the whole family suffers from. This thesis describes the development of an anomaly detection system that is interpretable, which therefore aims at alleviating the problems introduced above by trying to focus as much as possible on the perspective of the end-user. The two main topics are anomaly detection on the one side and the interpretability of the models on the other.

Machine Learning approaches for Anomaly Detection in Industrial IoT scenarios

by

Enrico CONVENTO

September, 2022

Contents

1 Introduction **1**
1.1 Main Goals . . . 1
1.2 Real-world industrial case . . . 5

2 Data exploration and features engineering **7**
2.1 Data acquisition and pre-processing . . . 7
2.1.1 Machine System Health (MaSH) platform . . . 7
2.1.2 Data store and management . . . 10
2.1.3 Raw data structure . . . 10
2.1.4 Data preparation . . . 12
2.2 Feature Engineering . . . 12
2.2.1 Alarms information section . . . 12
2.2.2 Machine states . . . 16
2.2.3 Machine states transitions . . . 18
2.2.4 Recipe/Format/Speed . . . 22
2.2.5 Visualization techniques . . . 23
2.3 Representations: featurized data . . . 27
2.3.1 Representations 1,2,3 . . . 28
2.3.2 Representation 4 . . . 29
2.3.3 Representation 5 . . . 31
2.3.4 Some considerations . . . 33

3 Anomaly Detection **35**
3.1 Anomaly Detection: an overview . . . 35
Types of anomalies . . . 35
AD algorithms . . . 37
3.2 Computational Aspects . . . 40
3.3 The isolation paradigm . . . 41
3.4 Isolation Forest . . . 41

3.4.1 How it works? . . . 42
3.4.2 Background: the Isolation Forest formally . . . 43
3.4.3 Results . . . 45
3.4.4 AD results and considerations . . . 47
Different time resolutions . . . 50
Test on other machines . . . 52
Data quality lack . . . 54
3.5 HDBSCAN . . . 56
3.6 HDBSCAN: interpretability . . . 58

4 Interpretability **61**
4.1 Interpretability in Machine Learning: a brief introduction . . . 61
4.2 Methods . . . 63
4.2.1 AcME . . . 63
How it works ? . . . 64
AcME: Anomaly Detection interpretability tool . . . 66
4.2.2 DIFFI . . . 70
4.3 Comparison: normalized counts and TF-IDF . . . 72

5 Conclusion **77**
5.1 Proposed interpretable anomaly detection system . . . 77
5.2 Future research direction . . . 79

A Appendix **81**
A.1 Anomaly detection results . . . 81
A.1.1 Normal points . . . 81
A.1.2 Anomaly points . . . 83
A.2 HDBSCAN interpretability results . . . 86
A.2.1 Medoids . . . 86
A.2.2 Curse of dimensionality handling . . . 91

Chapter 1

Introduction

1.1 Main Goals

Data-driven techniques are gaining more and more ground becoming more powerful and complex, capable of performing incredible tasks. On the other hand the techniques that aim to interpreting and explaining the answers provided by these models are scarce and act as a bottleneck to the increasingly vast adoption in the industrial world of these techniques. The trend that is being observed is that of a strong interest in what is defined as interpretable machine learning. This shifts the focus on models that humans understand and trust and all those methodologies that make a model verifiable and understandable.

The main topic of the thesis is related to the *development of an anomaly detection system based on alarm sequences that is simple, interpretable and also capable of providing the reasons why a sequence is considered anomalous*.

Anomaly detection is a subfield of machine learning that deals with the concepts of *normal* and *anomalous*. Given a set of data-objects or patterns, two families can be observed, the first, the majority one, takes into account the most frequent and therefore more "normal" patterns, on the other hand, the minority ensemble, composed by the data that deviate significantly from the rest of the objects is defined as the set of outliers which leads to the concept of anomalous. This separation can be directly perceivable at times or may require a model to find out the same. Regardless of the means to understand the deviant behavior of the data objects, their presence in the data has strong practical implications. Therefore, anomalous extraction/detection aims to discover such objects in the data. It is not possible to impose a sharp mathematical definition on outliers, as detecting them is strongly linked to the context and to the application itself. Because of the foregoing, the accurate and efficient detection of outliers in a data set is always challenging. Accordingly, considering an adequate definition and developing an acceptable method that meets the requirements are one of the main part that compose an outlier detection problem.

Given the above understanding, the outlier detection problem can be fundamentally defined as: given a set of data objects, finds a specific subset of objects that are remarkably dissimilar, exceptional and inconsistent with regard to the rest data.

Outlier detection is interesting to many researchers as a data cleaning task, prior research in this regard talks about the practical significance of the outlier phenomenon in real-life scenarios. Similarly, the growing number of research works being published on this research topic is a candid indicator of the importance associated with this task in the fields of data science and data mining. In this regard, a few of the recent applications involving outlier detection methods are the identification of misinformation and misbehavior on the web (Kumar et al., 2018), anomaly detection from system logs in complex computer systems (Ranga Suri, Murty M, and Athithan, 2019), modeling antisocial behavior in online discussion communities (Cheng, Danescu-Niculescu-Mizil, and Leskovec, 2021) , characterizing a malignant tumor in medical images for diagnosis (Wei et al., 2018) and many others.

Over the years, outlier detection has emerged as a significant data mining task due to the general perception that outliers represent evolving novel patterns in data. With the widespread use of data mining techniques in various application domains, many methods exist for detecting outliers that meet various application specifications requirements. Therefore, picking an adequate method becomes a challenge in

its own right as it decides the course of the detection process as well as the ultimate outcome.

Aligning with the case of interest, it must be said that AD is mainly an unsupervised learning task aimed at detecting anomalous behaviors with respect to data. In particular, it has an important role in many applications thanks to the capability of summarizing the status of a complex system or observed phenomenon with a single indicator called **anomaly score** and thanks to the unsupervised nature of the task that does not require human tagging.

Typically, traditional AD techniques as LOF (Breunig et al., 2000), ABOD (Kriegel, Schubert, and Zimek, 2008), Isolation Forest (Liu, Ting, and Zhou, 2009) are being preferred over solutions based on DNNs, this is empathized if a IoT scenario is considered.

In order to explain a bit about this last comparison DNNs in general suffer of one main problems that is linked to the fact that those models and in particular the training process associated depends on the complexity of the task, the dimensionality and the available computational power. While "classical" machine learning algorithms typically avoid these issues thanks to their structure and internal logic which make them more efficient models. The Isolation Forest is one of the most commonly adopted algorithms in the field of Anomaly Detection, due to its proven effectiveness and low computational complexity. Some example can be observed in Togbe et al., 2020, Chun-Hui et al., 2018 or Karev, McCubbin, and Vaulin, 2017.

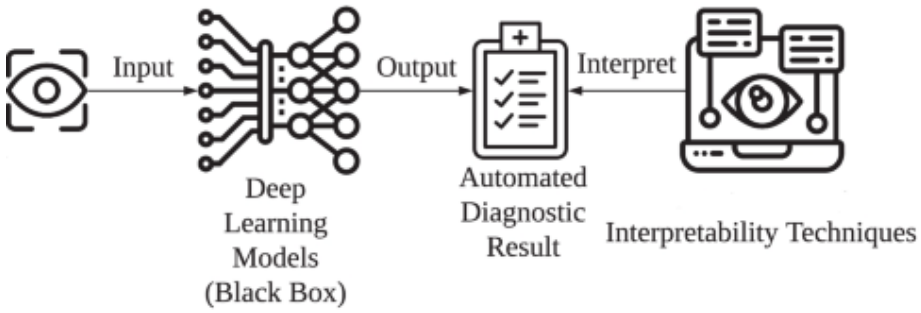


Figure 1.1

As said previously, one major drawback affecting Isolation Forest and other models used to perform anomaly detection is represented by the lack of interpretability.

So if a macro theme is related to AD, the other one is linked to the interpretability. This term can sound a little bit hazy, a non-mathematical definition of interpretability can define it as "the ability to explain or to present in understandable terms to a human" (Doshi-Velez and Kim, 2017). But why is it required to have this property of interpretability? The problem is linked to the fact that while AD algorithms have proven to be extremely useful and effective, the adoption of the latter is very far from being an everyday thing in the industry.

This is actually a problem that goes to the root of every technology associated with Machine Learning for decades. Choosing a model that was transparent to human practitioners or consumers often meant choosing straightforward data sources and simpler model forms such as linear models, single decision trees, or business rule systems. Although these simpler approaches were often adequate choices, and still are today, they can fail in real-world scenarios when the underlying modelled phenomena are nonlinear, rare or faint. On the other hand, highly effective models are often black-box, i.e., it is hard to explain their predictions. However, the lack of interpretability often hurdles the adoption of Machine Learning in practice.

The interpretability of the model attempts to fill this void by providing methods, even simple ones, that attempt to provide a machine-human connection (Ranga Suri, Murty M, and Athithan, 2019; Hall and Gill, 2019).

1.2 Real-world industrial case

The ultimate goal of the thesis, as previously anticipated, is interspersed with a real-world case study. The interpretable AD system was developed for filling machines produced by Galdi[1]. Galdi is a company located near Treviso that deals with the design and development of automatic filling machines, paying maximum attention to food safety and the repeatability of the performance of their machines over time.

The company offers different solutions and different products that go to cover a large area of the market, from the packaging of milk, eggs, fresh products and much more. The two main characteristics of Galdi are the customer-centered approach and continuous improvement due to continuous experimentation that allows Galdi to constantly expand the resources made available to the customer, in order to accompany him towards solutions capable of generating greater value, both in terms of before and after sales. These solid foundations make the company not only a partner of the world's largest producers of liquid foodstuffs, but also a strong support for small farmers who want to package their products or discover new market niches.

Motivated by the real-world industrial case, not only does the thesis deal AD and ML interpretability, but also with issues related to the productive deployment of ML solutions, ranging from the treatment of the raw data to the interaction with the end users. The latter is crucial for the design of useful tools to help the operators in managing the whole productive process in a symbiotic way with the filling machine.

[1] https://www.galdi.it/

Chapter 2

Data exploration and features engineering

This chapter explains the data acquisition processes, the storage, the exploration, the subsequent pre-processing to ensure that the data can be handled comfortably and then finally the feature extraction part. All of this refer to the features engineering process, namely the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modeling. The goal of feature engineering and selection is to improve the performance of machine learning algorithms.

2.1 Data acquisition and pre-processing

2.1.1 Machine System Health (MaSH) platform

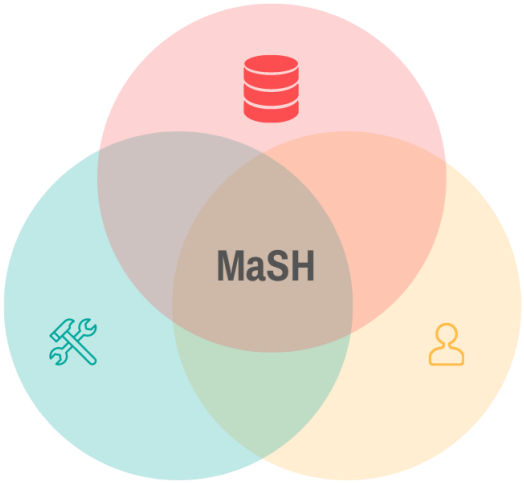
The most complex part associated with data acquisition is carried by the **Machine System Health (MaSH)** platform.

The monitoring of filling machines is one of the fundamental components to enable constant reliability of the machines and compliance with production objectives. In order to explain this, it is interesting to bring some estimates to understand

the importance of the monitoring of these machines, considering the current market situation. Referring to the field of filling machines, we speak of average production when around 7 million cartons/year are produced. This leads to the need to plan packaging and delivery in a precise and reliable way. Obviously, if this happens for a medium-small company, in the case of large food producers things get more complex, production times become even more tight and thus a machine downtime can represent a considerable economic and image damage.

To address this challenge, Galdi has developed a filling machine monitoring system called MaSH ("Machine System Health")[1], created with the aim of:

- Offering a detailed view of the efficiency of the machine
- Anticipating and prevent downtime
- Providing detailed production data
- Improving the work of operators and maintenance workers



HOW MASH WORKS MaSH is an in-house developed IoT system that monitors the efficiency of filling machines by controlling a sensor system that provides data for each part of the machine. The system synthesizes the production data of the machine(s) and translates them into a language useful for those who work on the machine by passing through a cloud platform. The system was developed in order to

[1] https://youtu.be/7xZ1X23KW6Q

adapt to the needs of production managers, maintenance workers or generic operators even with little domain experience by selecting the most relevant and functional information for their work.

To facilitate even more the use of MaSH in everyday life, the system is:

- equipped with a user-friendly interface
- securely accessible from any device: PC, smartphone, tablet
- integrated with management systems (ERP, MES, PLMS, CRM)

The MaSH system has been expressly designed to simplify the work of:

- OPERATORS

  The great news introduced by the MaSH is linked to the possibility of anticipating the downtime of the filling machines, signaling any malfunctions even before the operator can warn them. At the operator level it is very difficult to filter the information provided by the machine as the quantity of alarms received is enormous and this often leads to the use of a 'if it works then do not touch' policy with the consequence that the problem is reported when it is too late. With MaSH, on the other hand, the system itself selects and communicates the anomalies of the packaging machine.

- MAINTENANCE WORKERS

  To reduce machine downtime and in particular Mean Time Through Repair (MTTR), the maintenance technician must be able to identify the cause of the possible malfunction of the filler in the shortest possible time. To facilitate the resolution of the problems of the most complex cases, MaSH gives access to the alarm history, highlighting the recurring ones.

  As mentioned, however, the system is designed precisely to prevent problems with the filler: if the sensors signal a repetitive alarm or a drop in performance in a certain area from the filler monitoring analysis in real time, MaSH sends a push notification to the maintenance technician allowing it to carry out the necessary checks before a problem occurs.

- REMOTE SUPPORT

  Having machines scattered around the world, certain cases must be solved remotely as some manufacturers cannot afford lasting stops. In cases where

Help Desk support is required, MaSH helps to significantly speed up the work of technicians, both in terms of problem solving and resolution.

2.1.2 Data store and management

From the sensors, through the MaSH the management of data is done via Influx DB. Influx DB is an open source time series database written in Go language which is developed by InfluxData. It is optimized for high-availability, faster retrieval of data, and storage of time series data in fields such as operations monitoring, application metrics, IoT sensor data and real-time analytics. InfluxDB is purpose-built for time series data. Relational databases can handle time series data, but are not optimized for common time series workloads. InfluxDB is designed to store large volumes of time series data and quickly perform real-time analysis on it.

So, to summarize, the huge amount of data produced by the machine is saved in a database managed by Influx DB. At this point, the data is ready to be recovered and used.

2.1.3 Raw data structure

	table	serial	time	ASR
0		M1	2021-02-01 00:05:00	A1
1		M1	2021-02-01 08:15:00	A1
2		M1	2021-02-01 08:19:00	A100
3		M1	2021-02-01 08:25:00	A5
...		M1	...	...
10		M1	2021-02-01 21:05:00	A7

Figure 2.1: Table 1 structure example

The data used to build the AD system is downloaded from two tables, each of them conveys different information.

The two tables have different structures due to the different aims for which they are used. The first table keeps track of the alarms that have triggered associated

with timestamps and other information including the serial number of the machine in question and other fields associated to the recipe, the format or the speed. In the Active Stop Reason (ASR) field there are the IDs of the alarms that have caused a problem to the machine, each alarm has a meaning and it is associated with a macro zone of the machine such as the transfer chain or some photocell. In Fig.2.1 an example of the appearance of the table is shown. The second table carries different information, in particular the part that is taken into consideration concerns the production states of the machine. Basically, this table describes the status changes, pointing out the type of state, the alarms that made the machine switch to a different state and the amount of time that the machine remained in this state. The available states are described in the following table.

State	Meaning
idle	the machine is inactive waiting for new instructions
production	the machine is working
downtime	the machine is stopped by some non planned event
scheduled downtime	the machine is stopped due to a planned event, for example cleaning
performance loss	scheduled or unscheduled times, e.g. due to start-up or emptying times or supply of processing materials

In Fig.2.2 an example of the appearance of the table is shown.

	table	serial	time	ASR	type	elapsed
0		M1	2021-02-01 00:05:00	A1	idle	11111
1		M1	2021-02-01 08:15:00	A1	idle	122022
2		M1	2021-02-01 08:19:00	A100	production	12662332
3		M1	2021-02-01 08:25:00	A5	downtime	601
...		M1	...	...		
10		M1	2021-02-01 21:05:00	A7	downtime	500

Figure 2.2: Table 2 structure example

2.1.4 Data preparation

Before extracting the features from the data, the latter need to be processed. The acquired data are made up of alarm sequences associated with the instant in time when the alarm was triggered. Since it is not the single alarm that carries the whole information but the sequence of alarms or conditions that carry information, what is done is to break up the data into sequences of variable length, for example 2H, 4H, ..., 1D, obtaining a representation in which the various information are grouped in the time window they belong to. In such a way, information about the content of the series is stored in those sequences. This allows also to have different observation points on the data, from a vision linked to the particular to a more global vision, alleviating the sparsity and providing a more general perspective. Obviously going up with the dimension of the time window will require more data. Consider 24H of available data: if the splitting is performed per hour, then the result is about 24 sequences. On the other hand, if the chosen time window is daily only one sequence is obtained, the example shown is very simple but makes the substance understand well. Based on this representation, it is possible to extract the features.

2.2 Feature Engineering

Feature engineering is the automatic creation of new variables by extracting them from the raw data. The purpose of this step is to automatically reduce the data volume to a more manageable set for modeling. In the following sections, an explanation of the techniques used to extract the feature and what kind of information they carry in will be provided.

2.2.1 Alarms information section

Some methods of feature engineering include cluster analysis (Caruso et al., 2018), text analysis, edge detection algorithms (Katiyar and Arun, 2014) and principal component analysis (Shalev-Shwartz and Ben-David, 2014).

The approach used to extract features from the data draws inspiration from the **Natural Language Processing** (NLP)(Manning and Schütze, 1999), in a nutshell all the methods that embed ML and linguistics to understand how a language work, so it could be seen as a statistical approach to the language analysis. The applications

associated with this tool are infinite and range from machine translation to sentiment analysis, but focusing in the case in question the main usage is to structure data, transform unstructured text into normalized, structured data suitable for analysis or to drive machine learning algorithms. In our case, the techniques used refer to the Bag-of-Words technique (Qader, M. Ameen, and Ahmed, 2019), an approach for natural language processing that extracts the words (features) used in a sentence, document, website, etc. and represents them by frequency of use and other heuristics. To do this *sklearn.feature_extraction.text* provide *CountVectorized* and *TfidfVectorizer*. *CountVectorized* implements a very simple mechanism, that is tokenizing of the document provided, in which the document is the set of grouped alarms code, and count the words' occurrences, in which the "words" in this case are represented by the alarm IDs, by doing so, a matrix of counts is obtained. The matrix of counts is a matrix in which the columns are associated with the alarm IDs, the rows are divided by time bands and in them there are the counts associated with the alarms, an example of this and the overall process can be observed in Fig. 2.3.

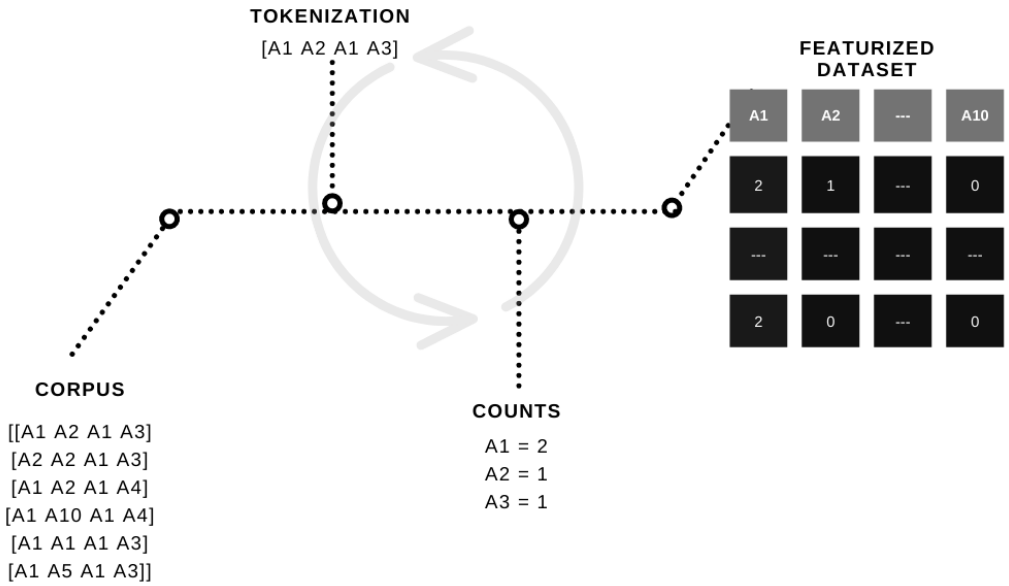


Figure 2.3: Counts visualization example

A common problem seen when extracting features from text, is the problem

of "stop words" or words that appear frequently in the text and are uninformative, for example, in English, “the”, “is” and “and”, would easily qualify as stop words. This is even more emphasized in AD, as the anomalies can be seen as very peculiar and rare sequences, so surely eliminating the redundancy will help to be more effective. For this the second approach, a common choice is to use *TfidfVectorizer*. It is associated with TF-IDF technique, which abbreviation stands for Term Frequency - Inverse Document Frequency, which is a way to attribute the importance of a word in the text by referring to the entire collection of documents. Given a token t its TF-IDF attribute is:

$$tf - idf(t) \quad = \quad tf(t,d) \quad x \quad idf(t) \tag{2.1}$$

Where d is the document that is taken into account of which t belongs. The term frequency is given by:

$$tf(t,d) \quad = \quad \frac{Number \quad of \quad occurences \quad of \quad word \quad t \quad in \quad document \quad d}{Number \quad of \quad words \quad in \quad document \quad d} \tag{2.2}$$

which basically is a normalized count of the number of occurrences of t in a given document. While the idf term is given by:

$$idf(t) \quad = \quad log \frac{Number \quad of \quad documents}{Number \quad of \quad documents \quad with \quad word \quad t} \tag{2.3}$$

With the inclusion of this term, a word will not get an “inflated” rating if it is present in many documents. In conclusion, these types of techniques are capable of providing a lot of information about the words present in documents, but like everything that is enjoyable, they also have downsides. First, no attention is paid to the sequential order of words in the text, this alone leads to the loss of context after the text has been vectorized. Secondly, the semantic values of words are not taken into account, each word is assumed to be independent of all others. Furthermore, as regards TF-IDF, the calculation of the term idf is computationally expensive especially if we go into the specific case of the alarm sequences in question as the amount of data is not small and if it is thought from a dynamic perspective, then this inconvenience becomes even worse.

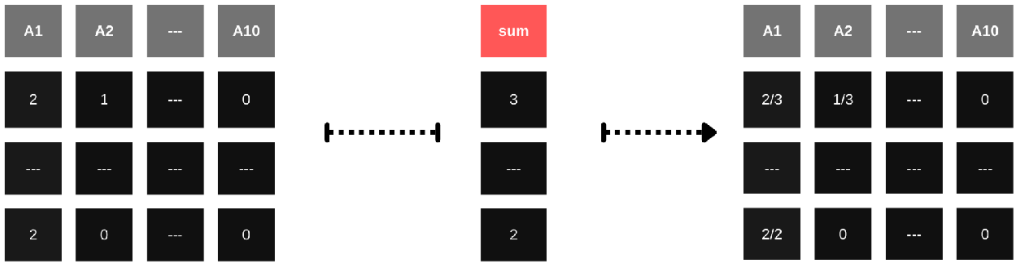


Figure 2.4: Normalized counts visualization example

A trade-off between simple counts and TF-IDF, can be achieved by constructing "normalized" counts. The basic idea is very naive and simple, consider a row of the dataset, in a NLP perspective the "document", transformed through simple counts. At this point, take the values of the non-zero counts and add them together, then divide the simple count associated with each alarm or word by the value of the cumulative sum obtaining the normalized count, the process can be visualized in Fig. 2.4. In conclusion, this representation uncovers a middle ground between simple counts and the TF-IDF, since it is not exactly a count because there is this "normalization" but that normalization process is more soft w.r.t TF-IDF in which it is done considering the whole corpus of documents, while in the normalized counts framework it is done considering only the current document.

In Fig. 2.5 it is possible to observe the result of the application of one of these techniques in which, summarizing, each document which coincides with the set of alarms that are triggered in the same time window, is "compressed" and represented in features. The display has been enriched by explicitly adding counts and meanings associated with the alarms to try to obtain something that can be easily interpreted by the operators. The description of the alarms in the image has been obscured for confidentiality reasons.

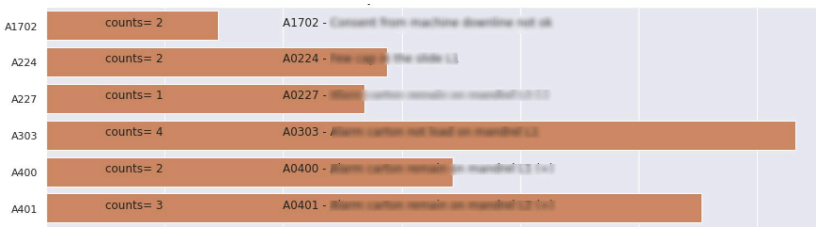


Figure 2.5: Alarms representation example

2.2.2 Machine states

As introduced above, beyond alarms, the second kind of information is associated to the production intervals. The access to this further information allows to build a more informative representation, enriching it with information that are quite different from the previous ones but that are able to describe well the machine behavior. Fig. 2.6 shows the average statistics about those production intervals. This is a good way to describe the "average" behavior of the machine, for example it is shown that the machine spends most of its time in production, downtime intervals cover a smaller section, as follows from intuition. Anomalous sequences can have a structure that deviates significantly from this average scheme. For example, anomalous sequences could be linked to strong variations of this basic structure such as equal distributed states, namely a chart in which each state covers an equal area.

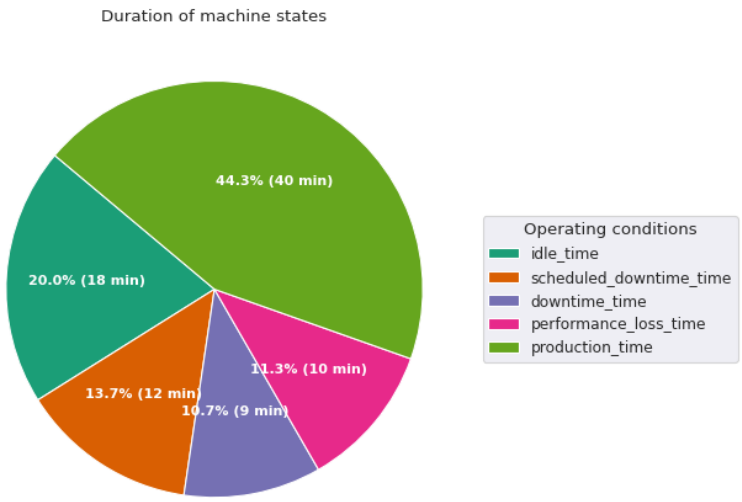


Figure 2.6: Time window composition over the whole dataset

Then putting the puzzle of concepts backs together, the first piece of information that is interesting to deal with is associated with the time a machine remain in a certain productive state. For example, considering a fixed time window, it is useful to know how long the machine has been in production or in some other state. To unify the representation, the result is not described in minutes or seconds but in percentages, to provide information on the composition of the time window without being linked to the size of the considered window.

To this it is also added a second piece of information, i.e., the knowledge of how many times the machine changes state within the time window, because as mentioned it is interesting to observe the composition of the time slot taken, but not only that. A very important case that refers to this, which was provided by domain experts, is the case where the machine stops many times but for a very short duration. This is certainly an unwanted phenomenon and entering the number of times the machine passes from one state to another can address the machine learning algorithm to understand these kinds of situations. An example of the representation that can be obtained is shown in Fig. 2.7.

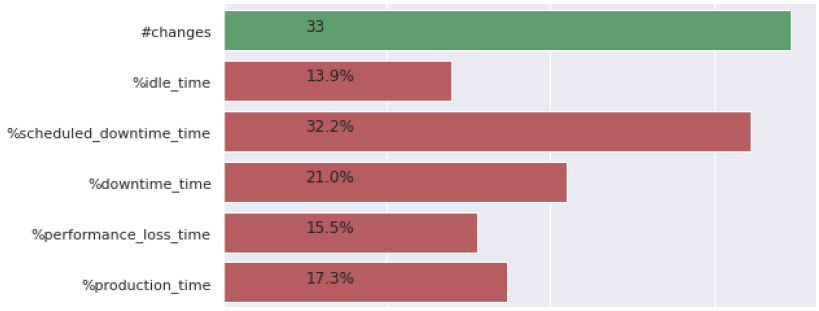


Figure 2.7: Time windows composition representation example

The representation is composed by two sections, the red colored section represent the time window composition, while the green bar is linked to the "#changes" features, namely it describes the number of times the machine change its state.

2.2.3 Machine states transitions

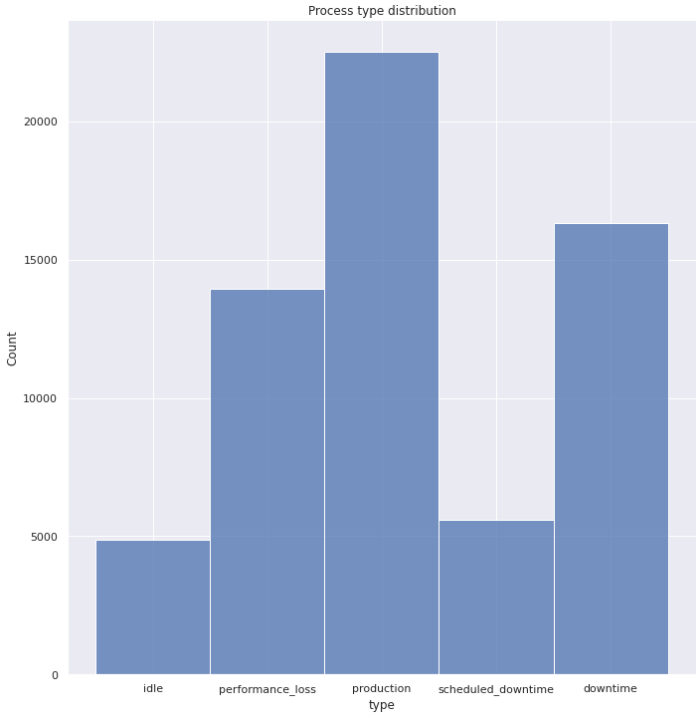
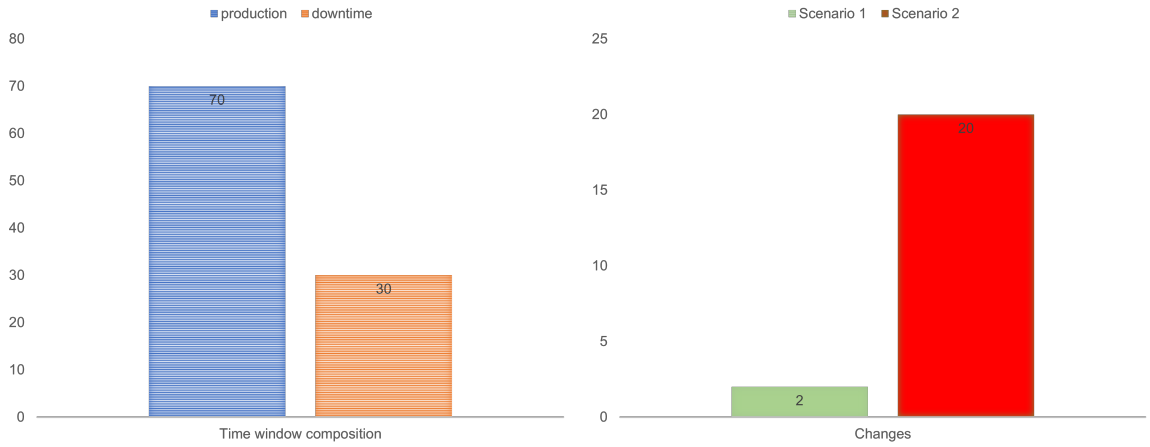


Figure 2.8: Machine state changes

The previous section explains how and why the information related to the composition of the time window is treated, however, the composition is not the only important information, other knowledge can be provided to Machine Learning algorithms to provide different concepts. As already introduced in the last part of the last section, the number of times the machine changes state is an informative feature. The idea is to break into pieces this information in order to obtain a more clear representation. To explain better, looking at the pie chart that takes into account the average machine behavior in Fig. 2.6, the only information that can be extracted is about the time window composition, all the information about the transitions from one state to the other are lost. In Fig. 2.8 is show as a histogram the term "#changes" as defined in the previous section but splitting the information referring to the type of change, for example, if an alarm has altered the machine state to idle then the type change is so labeled as idle. This type of visualization is very powerful because it shows the same things of the previous section from a different point of view, focusing on "frequency" and not on "composition", so for example, one thing that can be noticed looking at the histogram is that the downtime bin is quite high, so this means that a lot of changes leads to downtime, and this cannot be observed in Fig.

2.6 where the only insight about downtime is that it covers a small portion of the overall time window.



(a) State composition **(b)** Changes comparison

Figure 2.9: Scenarios comparison

So with the same overall duration, it is interesting to understand how many "interruptions" of the different types there have been. Let's focus on an explanatory example, consider two different time windows, both share the same time composition and the productive states are two, for simplicity. The time window composition shared by the two scenarios is shown in Fig. 2.9(a). The number of changes of state in the two cases considered is shown in Fig. 2.9(b). Notice that while in the first case there was only two change of state, approximately the machine start in production that move in downtime and then once the problem is solved, again in production, in the second there were many more changes, the situation described before is repeated multiple times.

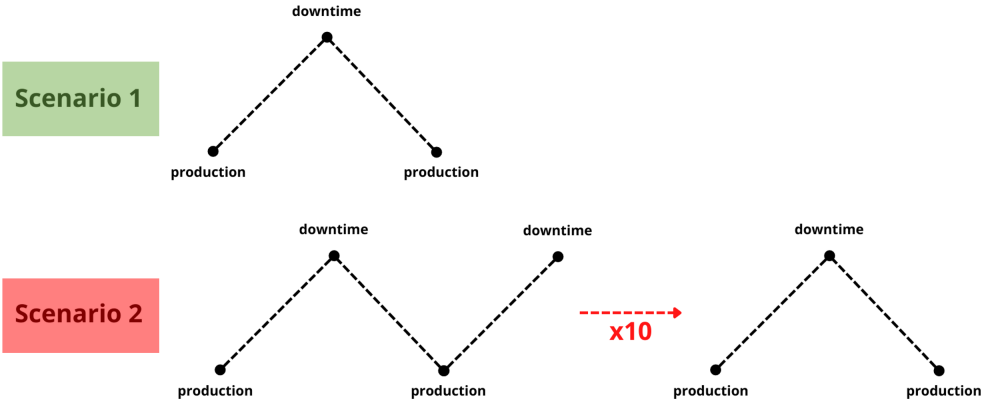


Figure 2.10: Example visualization

This simplified example helps to point out that also the frequency of the change of state must be taken into consideration because having a machine that, as in the second case, changes state many times hopping from one state to another is certainly not efficient, or it can be a sign that the machine needs a maintenance check. The two scenarios can be observed in Fig. 2.10.

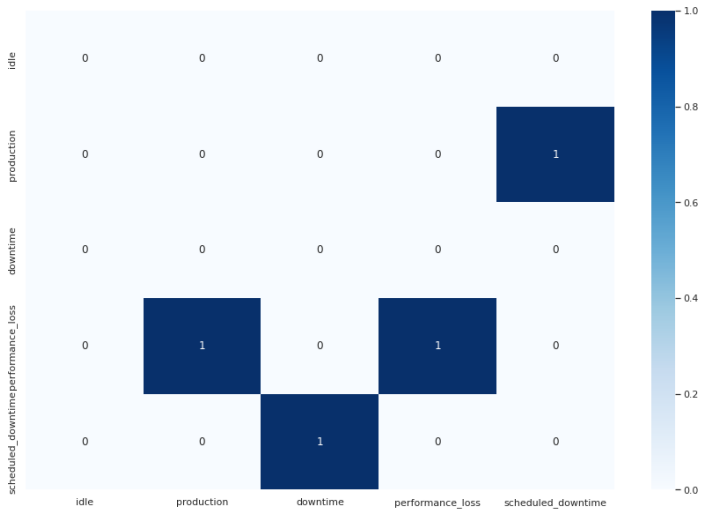


Figure 2.11: Transitions visualizations

To add further information it is interesting to observe not only where the alarm led but also what state it came from, for example there are situations in which there

are many changes, but they are between a state and itself for which they are not so relevant, this kind of changes depends on the data communication system, while there exists other cases in which the machine moved from one state to an other, in this case the information can be useful to debug the behavior of the machine. Substantially one dimension is added to the histogram of before, in such a way not only the "next" can be observed but also the "previous", an example can be observed in Fig. 2.11 where we can see the counts of the type of changes in a heatmap. To help visualize and handle the data, the counts used are normalized by the total number of changes in the time window. An example of what can be done connecting the dots can be observed in Fig. 2.12 where the heatmap of before is flattened and plot as a bar plot, also here a representation in percentages is used for convenience and because it is better in transmitting the result.

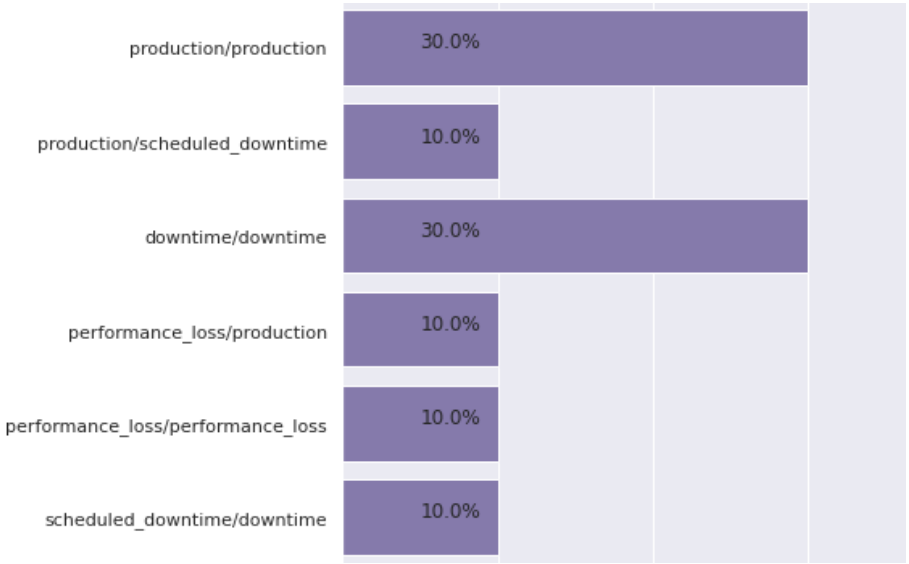
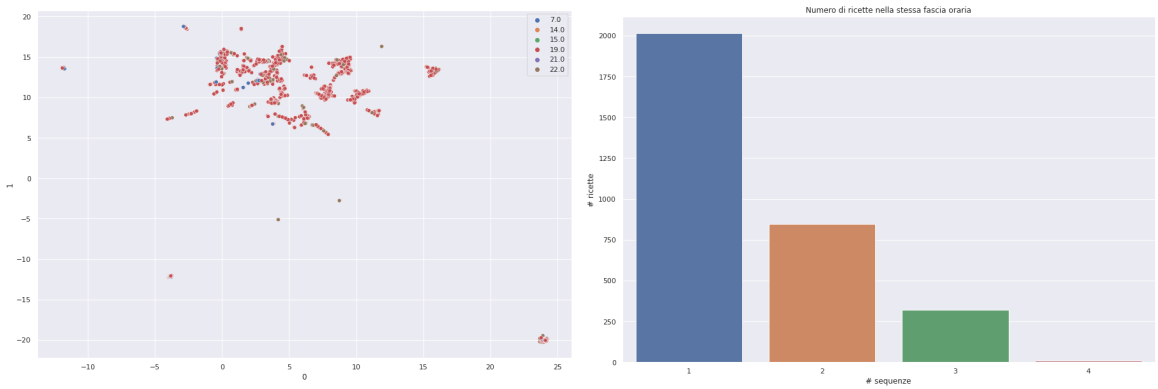


Figure 2.12: Transitions representation example

2.2.4 Recipe/Format/Speed

Since this is an unsupervised scenario, as already mentioned above, the creation of a representation is not scheduled but there is a need to combine various pieces and the results often turn out to be inconclusive, although the basic ideas may seem interesting. In this regard, other information has been added to the representation as the type of recipe, the format used and also the working speed of the machine. At first sight this is interesting information to deal with. For example, a recipe may be more difficult than others to make and therefore it may lead to anomalous behaviors with higher probability, or a similar thing can happen if the production speed is very high.

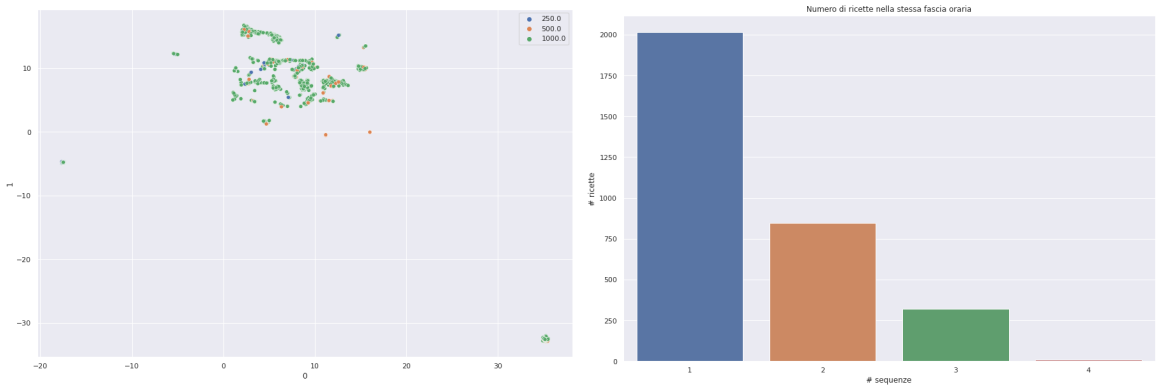


(a) UMAP visualization **(b)** Histogram counts

Figure 2.13: Recipe

However, discussing the results with domain experts, it turned out that none of this information leads to useful results, i.e., no interesting pattern was detected. The main causes attributed to this are associated to the fact that these quantities cannot easily blended with the rest of the information and the fact that there is not enough variability. This is highlighted in Fig. 2.13(a) and Fig. 2.14(a).a where the UMAP representation of alarms plus the recipe and format information respectively does not lead to the creation of interesting structures, and most of all it can be observed that the machine task is modified very rarely. In Fig. 2.13(b) and Fig. 2.14(b) are shown the number of different recipes or formats counting the number of the changes in a time window of two hours. As it can be observed, typically things does not change a lot in the time windows. As said, this does not carry a lot of information, unless the time window to be considered is greatly enlarged.

Therefore, although this information is interesting, their use is not very useful or in any case it requires some precautions such as the amount of data that would make their use more valuable. This goes to highlight that finding ourselves in a real case not every strategy can be applied and this is not a problem linked only to the type of strategy but also to the trade-off made between effort and results.



(a) UMAP visualization

(b) Histogram counts

Figure 2.14: Volume

2.2.5 Visualization techniques

Dimensionality reduction is a powerful tool to visualize and understand large, high dimensional datasets. For the development of the AD system mainly two different techniques were used, namely PCA and UMAP, in the next paragraphs a brief introduction of the two techniques will be made.

PCA Principal component analysis (PCA) (Shalev-Shwartz and Ben-David, 2014) is the process of extracting principal components and use them to obtain a "compressed" representation of the data, what is typically done is to use the first principal components and ignore the rest, for example in order to visualize some high dimensionality data only 2 or 3 dimension are kept. In PCA, the reduction is performed by applying a simple linear transformation to the original data. Assume the original data is in $\mathbb{R}^d$ and we want to embed it into $\mathbb{R}^n (n < d)$ then we would like to find a matrix $W \in \mathbb{R}^{n,d}$ that induces the mapping $\mathbf{x} \mapsto W\mathbf{x}$. Typically, this is achieved by performing Singular Value Decomposition (SVD), since it can be shown that the principal components are eigenvectors of the data's covariance matrix ordered by the eigenvalues magnitude. The n principal component are then the fist n eigenvectors which describe the direction in which data spread more, so what is done is

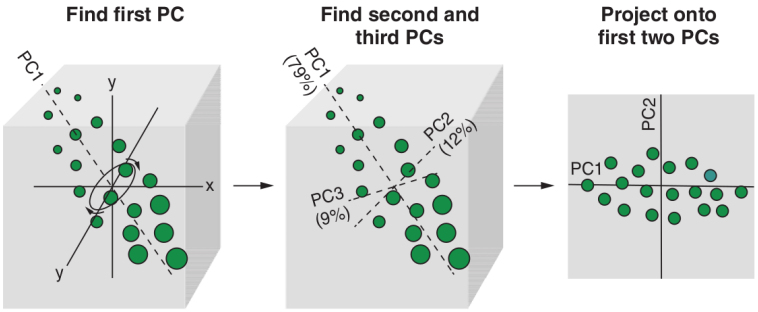


Figure 2.15: PCA visualization

to project the data exploiting the information contained in the eigenvectors. PCA tries to preserve the global structure of data, when converting a high dimensional data to a smaller one exploiting the information about the principal components it preserves the global structure, but all the clusters are typically mapped as a whole, due to this local structures might get lost. In conclusion, strengths of PCA are due to the fact that PCA performs a linear transformation so it admits perfect recovery, it is computationally inexpensive and it creates new axes that are directly interpretable in terms of the original variables. On the other hand, the weakness of PCA are related to the simplicity of such technique, mapping from an high dimensional space to a low dimensional one some times cannot be done by linear methods.

UMAP Sometimes the information contained in a set of variables can't be extracted as a linear combination of these variables. In such situations, there are a number of nonlinear dimension-reduction algorithms we can turn to, such as *t-Distributed Stochastic Neighbor Embedding (t-SNE)*(Maaten and Hinton, 2008) and *Uniform Manifold Approximation and Projection (UMAP)* (McInnes, Healy, and Melville, 2018). In brief, t-SNE is one of the most popular nonlinear dimension-reduction algorithms. It measures the distance between each observation in the dataset and every other observation, then randomizes the observations across (usually) two new axes. The observations are then iteratively shuffled around these new axes until their distances to each other in this low-dimensional space are as similar to the distances in the original high-dimensional space as possible. UMAP is another nonlinear dimension-reduction algorithm that overcomes some of the limitations of t-SNE. It works similarly to t-SNE (finds distances in a feature space with many variables and then tries to reproduce these distances in low-dimensional space) but differs in the way it measures distances.

UMAP is state of the art, having only been published in 2018 from McInnes et al, it has a few benefits over the t-SNE algorithm. First, it's considerably faster than t-SNE, where the duration it takes to run increases less than the square of the number of cases in the dataset. To put this in perspective, a dataset that might take t-SNE hours to compress will take UMAP minutes. The second benefit is that UMAP preserves both local and global structure, taking proper precautions.

So how does UMAP work? Well, UMAP assumes the data is distributed along a manifold. A manifold is an n-dimensional smooth geometric shape where, for every point on this manifold, there exists a small neighborhood around that point that looks like a flat two-dimensional plane. In the simplest sense, UMAP constructs a high dimensional graph representation of the data then optimizes a low-dimensional graph to be as structurally similar as possible.

Without delving into the theory related to the two algorithms, we note that the biggest difference between the output of UMAP and t-SNE is this balance between local and global structure: UMAP is often better at preserving the global structure in the final projection. This means that inter-cluster relationships are potentially more significant than in t-SNE. However, it is important to note that since UMAP and t-SNE both necessarily deform the high dimensional shape of the data when projected to smaller dimensions, any axis or distance in smaller dimensions is still not directly interpretable in the way of techniques such as PCA. For this reason, treating the UMAP representation with techniques that use distances, e.g. clustering with k-means, could give misleading results if the representation is not done properly.

Although both UMAP and t-SNE produce somewhat similar results, the higher speed, better preservation of the overall structure and more understandable parameters make UMAP a more effective tool for visualizing high-dimensional data especially in practice. Some results of the application of UMAP can be observed in (Becht et al., 2018), the article contains a very thorough comparison between UMAP and t-SNE which is performed on real data.

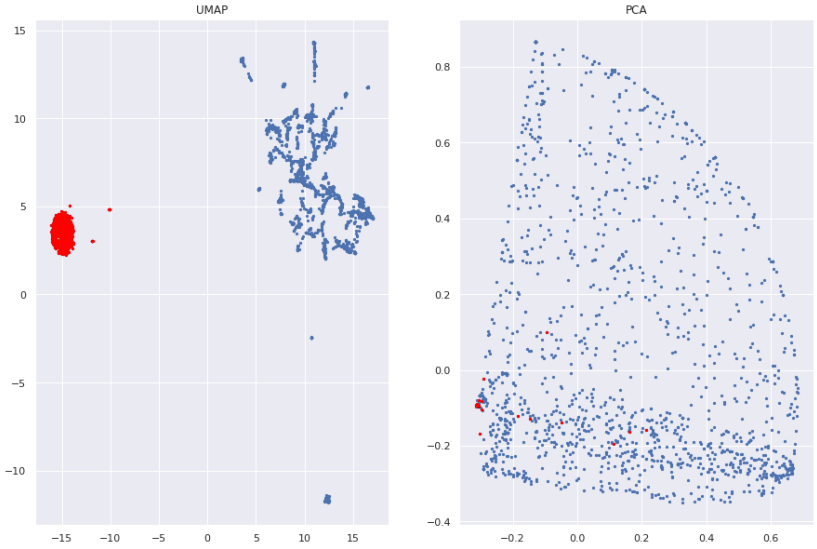


Figure 2.16: Visualization comparison

In practice, both PCA and UMAP tend to be used because while one gives us a global perspective, the other maintains the local structure. Furthermore, techniques such as t-SNE and UMAP are robust to outliers so in the case in question their use is very useful. It can be observed in Fig. 2.16 what has been said, in particular as regards the local structure of the graph.

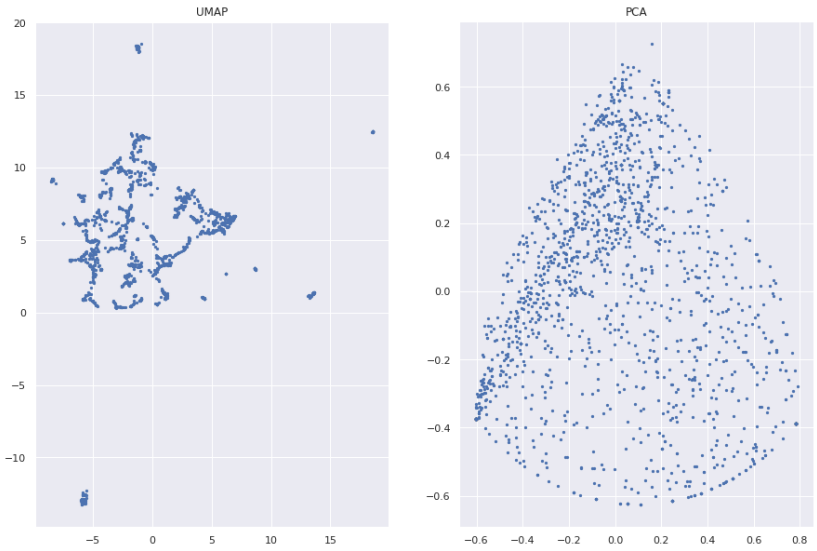


Figure 2.17: Visualization comparison

As in everyday situations it is important to evaluate from different points of view to extrapolate information that we may not be able to grasp directly, so using

PCA and UMAP at the same time can help, especially during the data preparation phase. The ability of UMAP to preserve the local structure of the clusters has made us pay attention to the dense area that can be seen in Fig. 2.16. Notice that the PCA visualization does not highlight any conglomeration of points. From a posteriori analyses it was noted that this area is associated with sequences that are not very informative, in particular these sequences contain only alarms with id "0", these alarms indicate a reset of the machine from which we can also deduce why the large number, however these sequences are useless in an AD scenario. This made it possible to remove and further clean up the data obtaining a representation like the one in Fig. 2.17 which is much clearer and furthermore this helps the machine learning algorithm as it provides better data.

2.3 Representations: featurized data

The application of techniques related to ML in a real problem suffers from many problems, and most of them are related to data and in particular to the quality and the information content that the latter has. Often, when it comes to ML, it is thought that the central part of the work is the modeling part. Usually, this part represents a minimum percentage of the whole process, whereas the central part of a ML solution is the data processing, because the performance of a ML solution is limited by the quality of the training data. So the process that leads to the creation of a representation that is representative enough, that has no informational biases is not simple and can be said to be the most critical part of the design and this is even more important in an unsupervised scenario. In this section, we want to highlight the whole process that led to the creation of a representation that seems to have some positive results. The representations that have been created are five, which are outlined in the following table.

Representation	Contents
1	counts
2	normalized counts
3	TF-IDF
4	counts/normalized counts/TF-IDF + machine state
5	counts/normalized counts/TF-IDF + machine state + state transitions

2.3.1 Representations 1,2,3

This section explains how the first three representations are made, the considerations made for them are grouped into a single section since although the methods exploited for their realization are slightly different, they should, in theory, carry the same information. The information packed in them refers to the alarms part which is the central and most informative part for the end user, so the first representations start from these basic and fundamental data. The extraction of the features takes place as described in Sec. 2.2.1, respectively using simple counts, normalized counts and TF-IDF.

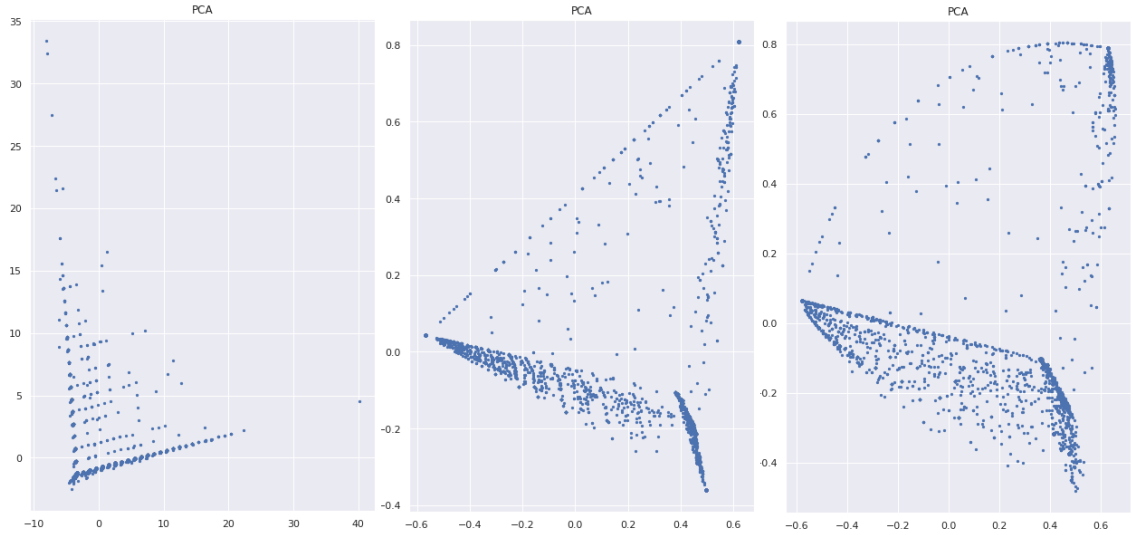


Figure 2.18: PCA comparison

In Fig. 2.18 the three representations can be compared using PCA which gives us a representation that is very reliable on a global level as PCA in a nutshell eliminates the "non-principal" dimensions. At first analysis, it immediately appears that representations 2 and 3 are very similar and differ a little with respect to 1. Given the problem unsupervised nature, the deductions made largely refer to the theory because it is not easy to understand what is right and what is not. From a theoretical point of view TF-IDF certainly provides us with a solution more robust than the other two cases which, however, should not be neglected because on a practical level they are much more efficient and simply achievable. Already at this level of information it seems that representation 2 and 3 are very similar.

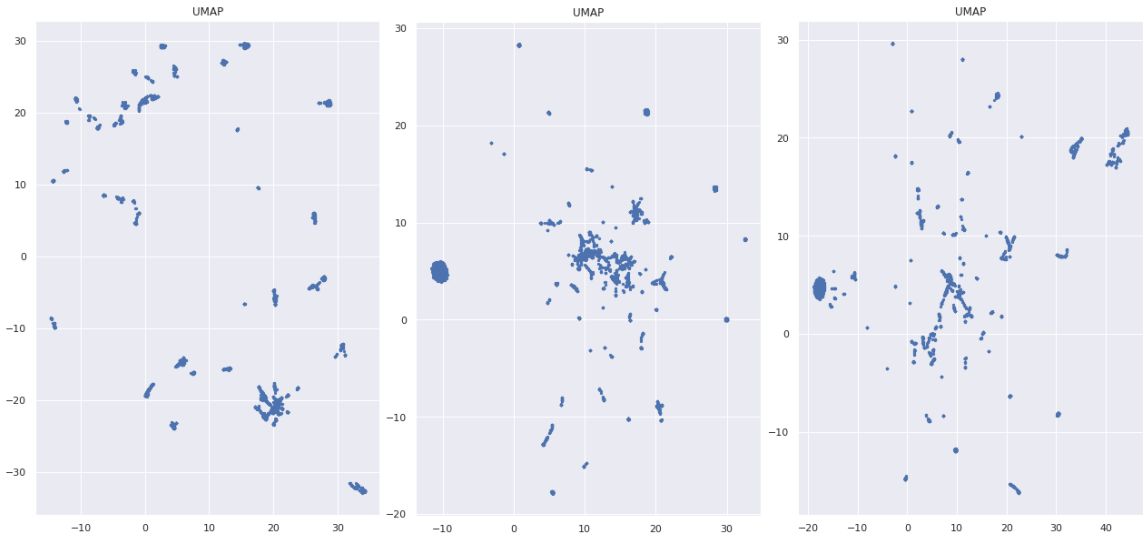


Figure 2.19: UMAP comparison

In Fig. 2.19 the three representations can be compared via UMAP. This type of visualization provides us with some more insight at the local level, since UMAP can be tuned to be more focused on a global or local vision. In this case, noticing differences or similarities between the three representations requires a little more attention, however due to the bias of the previous analysis, it seems also here that the representations based on normalized counts and TF-IDF have some similarity as in the previous case. Here, unlike PCA, the structure at a local level is a little more emphasized, in fact clusters formations can be immediately noticed.

2.3.2 Representation 4

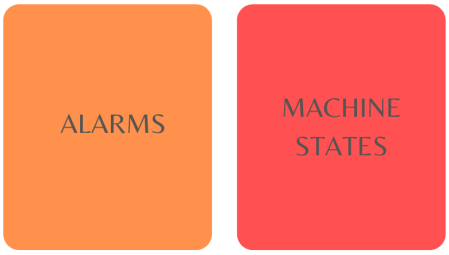


Figure 2.20: Representation 4 scheme

The representation illustrated in this section is composed of two parts, it can be seen as a more general extension of the previous ones. The first part of the representation is realized as one of the representations 1/2/3, than only one of these will be used, further consideration will be make in the following about this topic.

The second part is carried out using the information from the time window composition as described in Sec.2.2.2, the data of these two blocks are intertwined using the date and time information, in Fig. 2.21 a PCA and UMAP visualization is available, in particular the representation make use of TF-IDF for the reasons already mentioned above. Summarizing the union of the two different information leads to a representation that follows a diagram as in Fig. 2.20 which combines the information of the alarms to that of the state of the machine. This allows to have a representation that is able to better describe the link between an alarm and the time window state composition. Where to be a little more explicit, the part of machine states is made up of the percentage in which the machine remains in a certain production state, the available states are listed in table in Sec. 2.1.3

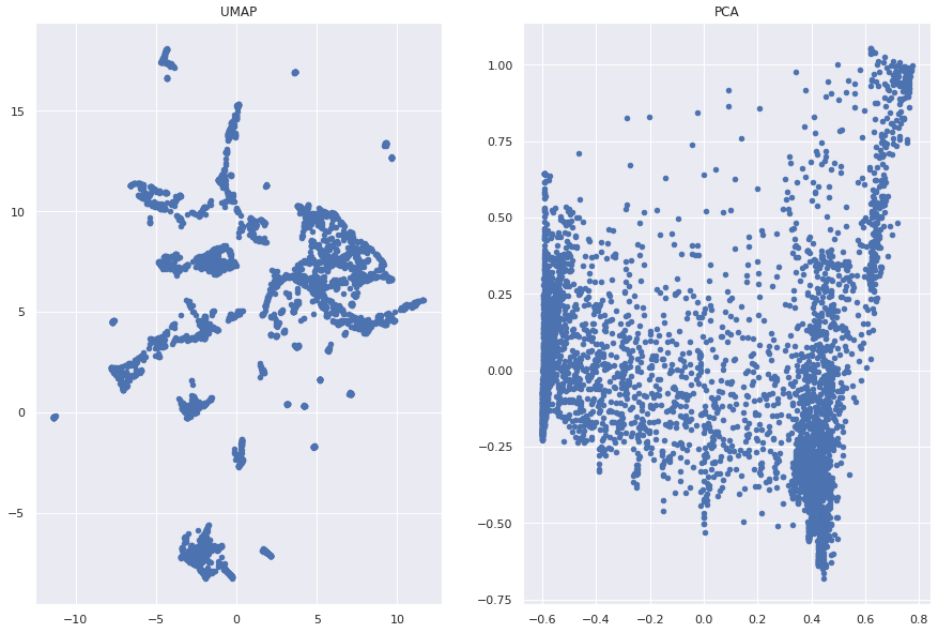


Figure 2.21: Representation 4

The insertion of the information relating to the state of the machine makes the UMAP and PCA visualization, observable in Fig. 2.20, vary considerably from what has been seen in the previous paragraph. In detail, it is observed how the data give rise to more clusters, indicating that similar situations exist, patterns that resemble

each other, unlike what happened in previous representations in which the graphical visualization was more scattered. It is important to keep in mind that the visualization is in any case the result of a reduction in dimensionality which therefore contains approximations, but it is important to make all the necessary considerations to try to help the intuition to draw some insight that could prove useful for the purpose of the analysis.

2.3.3 Representation 5

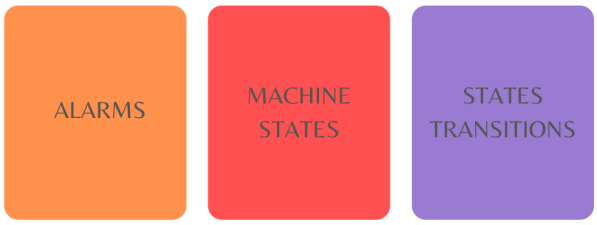


Figure 2.22: Representation 5 scheme

The last representation tries to condense even more information trying to help even more the algorithm and the user, it comes up as a result of the comparison with operators and domain experts regarding representation 4 and it tries to encapsulate all the information useful for monitoring the behavior of the machine. Representation 5 is made up of three basic sections, the first created from the table with alarm information, while the other two sections related to information from the productive intervals table. So here too, we find an extension of representation 4 to which information relating to state transitions is added.

Recall from Sec. 2.2.3 that the state transitions part describes all transitions that occur from one state to another, keeping track of the number of changes and labeling them using the previous and next state. As already mentioned in the Sec. 2.2.2, the "#changes" field has a considerable weight on the final result, so as consequence the block associated to state transitions is added to the representation.

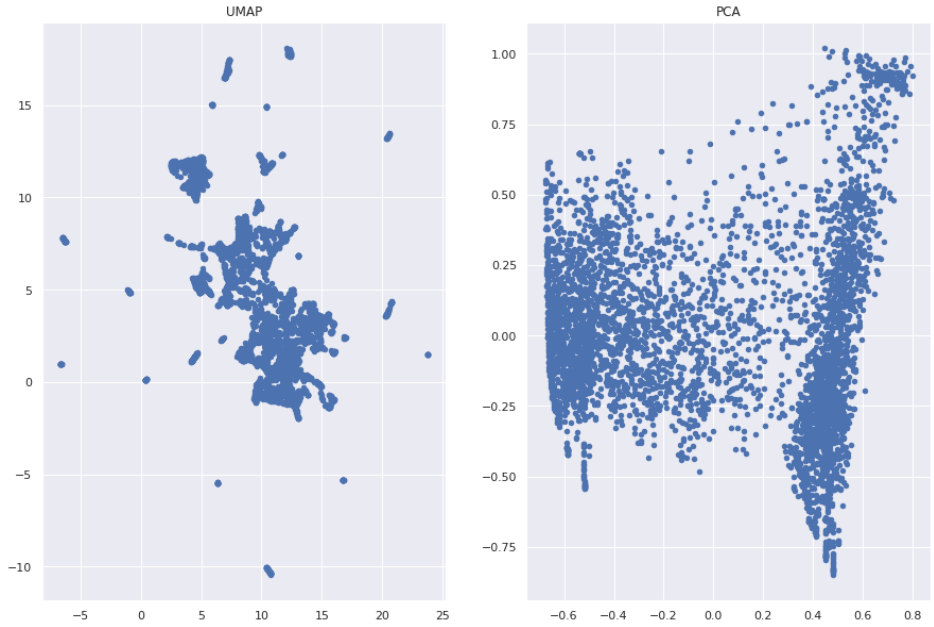


Figure 2.23: Representation 5

From the comparison of Fig. 2.23 and Fig. 2.21 not many differences are noticed, this shows how the last added portion is certainly very important for the end user, but it goes to level out minimal things as similar information is already contained in the "#changes" field.

2.3.4 Some considerations

As already anticipated during the whole process of creation of the representation described in the previous paragraphs, there is no answer to the question "is this representation correct?", we can make considerations and more but since there is no supervisor it is not at all certain.

The most important considerations that have led to prefer one or another representation will be made in the next chapters also using the results obtained by the AD and various techniques of interpretability.

Another important consideration that has to be made concerns the amount of data that is necessary for the algorithm to work without going into suboptimal conditions. The Fig. 2.24 shows three years of data extracted from sequences concerning the alarms part only, and each point is labeled with the year. What can be noticed is that the structure of the points is substantially similar, there are no clusters that stand out by partitioning the data for years, so this becomes an indicator that the machine has always performed the same or similar task overall. Obviously this is strongly related to the machine, in general the behavior is similar to that shown, as a machine very rarely totally changes what it is doing. This factor is certainly to be taken into consideration when bringing the system into production, as the system must work in real-time, and it is necessary to make a trade-off of information loss and memory efficiency.

Figure 2.24: Data amount comparison

Referring to the latter, it is necessary that the machine analyzed is only one and data taken from multiple machines are never processed for various reasons, including the main one, i.e. the fact that each machine performs slightly different functions, for example it uses different formats, different recipes and also different speeds, so using one machine already has a lot of variability, using data from multiple machines further complicates the anomaly detection task emphasizing the variability of the data.

Chapter 3

Anomaly Detection

At this point, having the various representations obtained in the last chapter available, the next step will be discussed, i.e., anomaly detection. In this chapter the chosen AD technique will be described starting from the definition of anomaly, passing through the various algorithms available, up to the discussion of the results obtained.

3.1 Anomaly Detection: an overview

What is Anomaly Detection? Starting from the basis it is necessary to provide a definition associated to the concept of **outlier** , the definition is not unique, but typically an outlier is simply a data point that deviates considerably from the rest of the data points in a given data set (Chandola, Banerjee, and Kumar, 2009). The process of trying to identify these outliers or anomalies is referred to as anomaly detection.

When it comes to large and high dimensional datasets, very complex patterns can be included that cannot be found simply manually inspecting the data, looking for the point that is the furthest away from the data, adding the need for models and algorithms.

Therefore, the study related to anomaly detection is very important and must address various problems, ranging from the dimensionality problem to the interpretability of these anomalies in a real world scenario.

Types of anomalies

In the domain of data science, there are basically four types of anomalies (Zoppi, Ceccarelli, and Bondavalli, 2018). Understanding the logic behind them can have a

big impact on how they are handled, so an *ad hoc* solution can be developed to better focus on one or to one other. Those macro categories are:

- Global Anomalies: Corresponding to data points that differ significantly from the rest of the data points, global anomalies are known to be the most common and intuitive form of anomalies. Usually, global anomalies lie very far from the mean or median of any data distribution.
- Contextual or conditional anomalies: These anomalies have values that differ significantly from those of other data points in the same context. For example, let's get into the case of Galdi, since each machine works in a similar but not equal conditions, namely two machines are filling with two different products, the response of the two machines can be different. Anomalies in one dataset may not be anomalies in another, so in this case some information related to where the data comes from is very relevant.
- Local Anomalies : Local anomalies are quite difficult to detect, since those points can be seen as normal record since them are not too far from the normal points. The name local anomaly came from the fact that such a kind of point is only anomalous when compared with its close-by neighbourhood.
- Collective Anomalies: Anomalies can be clustered into an anomalous cluster, in a nutshell anomalous objects that are tightly grouped because they have the same anomalous character and are referred to as collective anomalies. Those are quite difficult to be separated from the rest of the data.

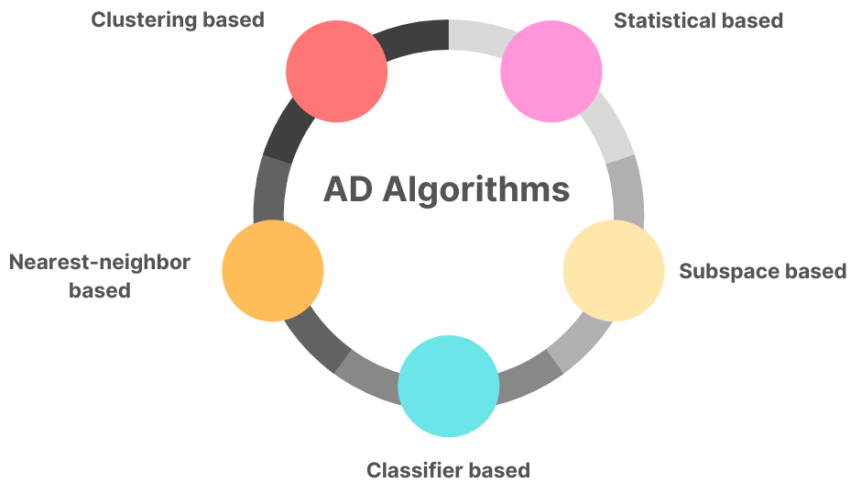


Figure 3.1: AD algorithms

AD algorithms

Now that the concept of anomaly has been introduced, it is useful to make a brief summary of the anomaly detection techniques that are available. A majority of the techniques can be categorized into classification-based, nearest neighbor-based, subspace based, clustering-based and statistical techniques. An intuitive visualization of the different categories can be observed in Fig. 3.1. For each of these approaches, the nature of the data may be supervised, semi-supervised or unsupervised. In the following some of them will be introduced, focusing more on their pro and cons.

Classification-based techniques Classification-based anomaly detection techniques operate in a similar two-phase fashion. The training phase learns a classifier using the available labeled training data. The testing phase classifies a test instance as normal or anomalous, using the classifier. Associated to this family of algorithms there are neural networks-based, SVM-based and rule-based methods. The advantages of classification-based techniques are as follows:

- Classification-based techniques can make use of powerful algorithms that can distinguish between instances belonging to different classes

- The testing phase of classification-based techniques is fast, since each test instance needs to be compared against the precomputed model

The disadvantages of classification-based techniques are as follows:

- Rely on the availability of accurate labels, which is often not feasible

Nearest neighbor-based techniques Nearest neighbor-based anomaly detection techniques require a distance or similarity measure defined between two data instances, the main assumption is associated to the fact that typically normal points have many neighbors while the anomalies are located far from other points. The advantages of nearest neighbor-based techniques are as follows:

- A key advantage of nearest neighbor-based techniques is that they are purely data driven, unsupervised by nature and do not make any assumptions regarding the generative distribution for the data.
- Adapting nearest neighbor-based techniques to a different data type is straightforward and primarily requires defining an appropriate distance measure for the given data.

The disadvantages of nearest neighbor-based techniques are as follows:

- For unsupervised techniques, if the data is not dense enough, or if the anomalies hiding between normal points have close enough neighbors, the technique fails to label them correctly resulting in missed anomalies.
- For semi-supervised techniques, if the normal instances in the test data do not have enough similar normal instances in the training data, the false positive rate for such techniques is high but that is, if you want, a problem related to the data that affects the final result.
- Performing a closest neighbor technique relies heavily on a distance measure, defined between a pair of data instances, which can effectively distinguish between normal and anomalous instances. Defining the distance measurements between instances can be difficult when the data is complex, such as graphs, sequences and so on.

Clustering-based techniques Clustering is used to group similar data instances into clusters. Even though clustering and anomaly detection appear to be fundamentally different from each other, several clustering-based anomaly detection techniques have been developed. Several clustering-based techniques require distance computation between a pair of instances, the key difference between the two techniques, however, is that clustering-based techniques evaluate each instance with respect to the cluster it belongs to, while nearest neighbor-based techniques analyze each instance with respect to its local neighborhood. The advantages of clustering based techniques are as follows:

- Clustering-based techniques can operate in an unsupervised mode.
- Such techniques can often be adapted to other complex data types by simply plugging in a clustering algorithm that can handle the particular data type.
- The testing phase for clustering-based techniques is fast since the number of clusters against which every test instance needs to be compared is a small constant.

The disadvantages of clustering-based techniques are as follows:

- Performance of clustering-based techniques is highly dependent on the effectiveness of clustering algorithms in capturing the cluster structure of normal instances.
- Many techniques detect anomalies as a byproduct of clustering and hence are not optimized for anomaly detection.
- Several clustering algorithms force every instance to be assigned to some cluster. This might result in anomalies getting assigned to a large cluster, thereby being considered as normal instances by techniques that operate under the assumption that anomalies do not belong to any cluster.
- Several clustering-based techniques are effective only when the anomalies do not form significant clusters among themselves.
- The computational complexity for clustering the data is often a bottleneck

3.2 Computational Aspects

This exploration is aimed at highlighting the practical significance of various computational aspects that one may come across typically when dealing with the outlier detection problem. Thus, the idea here is to point out what one should be concerned with, in order to develop efficient techniques for outlier detection.

Impact of the Type of Data The data one encounters with regard to outlier detection could be either categorical or numerical. Several algorithms that are popularly used on numerical data can't be used on categorical data. Therefore, it is necessary to devise acceptable techniques to process data described using categorical attributes. In the context of outlier detection, the effectiveness of the detection algorithm strongly depends on the interpretation or one's perception about outliers in the sense of categorical attributes. So, it becomes imperative to have an intuitive definition for characterizing outliers and develop the detection algorithm accordingly.

Unavailability of Labeled Training Data It is known that any machine learning activity primarily belongs to one of the two learning modes: supervised and unsupervised. The exact nature of the learning task is decided based on the availability of labeled data. As discussed in the previous section, outlier detection is generally considered as an unsupervised learning task due to lack of awareness on the kind of outliers present in the data.

Exploring Outliers in High Dimensional Spaces Data related to different real-life applications are accumulated by having a high cardinality which could mean a high information content or a high redundancy. Typically, a small subset of features is sought that retains the properties of the data without loss of important information. This requires a feature selection process to reduce the impact of high dimensionality on learning activity such as data clustering. When a data sample is represented as a high-dimensional vector, it is difficult to distinguish between its closest and most distant neighbors due to the curse of dimensionality. Feature selection should be used as a means of dimensionality reduction formulating acceptable measures to assess the relevance and redundancy of a specific feature associated with outlier detection.

3.3 The isolation paradigm

All these techniques have advantages, but typically they are linked by some drawbacks that in some way make them similar. The vast majority of existing anomaly detection approaches build a profile of normal instances, then identify anomalies as those that do not conform to the normal profile. Their anomaly detection abilities are usually a "side-effect" or by-product of an algorithm originally designed for an other purpose than anomaly detection. This leads to two major drawbacks:

- these approaches are not optimized to detect anomalies as a consequence, these approaches often under-perform resulting in too many false alarms (having normal instances identified as anomalies) or too few anomalies being detected
- any existing methods are constrained to low dimensional data and small data size because of the legacy of their original algorithms

The next move is to introduce the concept of isolation, where the term isolation means “to separate one instance from the rest of the instances”, to achieve this, the *isolation* paradigm takes advantage of two quantitative properties of anomalies:

- they are the minority consisting of few instances
- they have attribute-values that are very different from those of normal instances

In other words, anomalies are “few and different”.

3.4 Isolation Forest

Isolation forest is a machine learning algorithm associated to the field of anomaly detection(Liu, Ting, and Zhou, 2009). In particular, it is classified as an unsupervised learning algorithm that aims to identify anomalies by “isolating” outliers in the data. In this particular case, the scenario can be considered unsupervised, i.e. no such labels are known. One thing that should be clarified prior to the algorithm explanation is the concept of the “instance isolation”. Most of other algorithms are trying to model the normal behaviour, to learn the profile patterns. What is the problem with that? Well, defining the normal behaviour. The boundaries of normal behaviour are not typically well-defined. In most cases, it is too expensive and

time-consuming to label the data and get the information of normal and anomalous behaviour. The isolation forest tries to separate the anomalous instance from the rest of the data (thus the term "isolation") based only on the value of the features, without referring to any measure of similarity or labels.

3.4.1 How it works?

Isolation Forest works by building an ensemble of trees, called isolation trees, for a given dataset. A particular isolation tree is built upon a feature by performing the partitioning. If we have a feature with a given data range, the first step of the algorithm is to randomly select a split value out of the available range of values. When the split value is chosen, the partitioning starts, each instance with a feature value lower than the split value is routed in the one side of the tree, while each instance with a feature value higher or equal than the split value is routed in the opposite side of the tree. In the second step, another random split value is selected, out of the available range of values for each side of the tree. This is recursively done until all instances are placed into terminal nodes (leaves), or some of the criteria set in the input parameters is met.

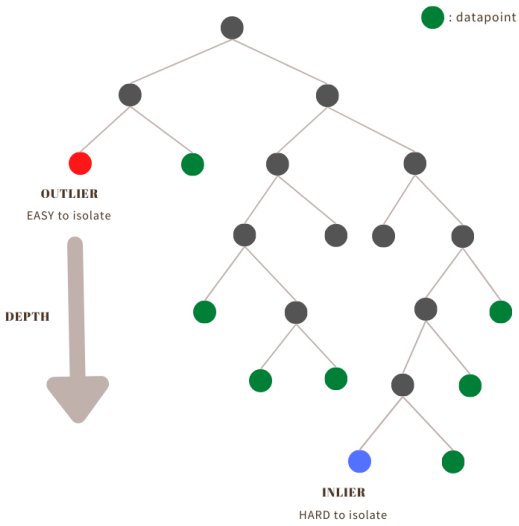


Figure 3.2: Isolation tree example

An **anomaly score** is used to have a measure of anomalous. So, in the iForest algorithm, the anomaly score is determined by the path length from the root node to the leaf in which the instance is placed. Since is it an ensemble, the average of

all the path lengths for a given instance is taken. Presumably the anomalies need fewer random partitions to be isolated compared to "normal" points in the dataset, so the anomalies will be the points which have a smaller path length in the tree, path length being the number of edges traversed from the root node.

Outliers are usually few objects which are very different from the rest of the data, therefore by using isolation trees there is a higher probability to pick a split early in the tree-building process that may separate the outlier from the rest of the data. In other words outliers are likely to be isolated after few splits and they will end up in leaves at a small depth. This phenomenon can be visualized in Fig. 3.2.

3.4.2 Background: the Isolation Forest formally

The following contains a more detailed description of how the isolation forest works. As said the isolation forest is an unsupervised AD algorithm leveraging an isolation procedure to provide a measure of anomalousness, called *anomaly score*. As previously said, it works in a recursive way, partitioning data based on some features. The basic idea is associate to the fact that the isolation procedure for outliers requires a limited number of iterations, since those points typically have a structure that differs a lot from the remaining data and for the inliers generally the isolation requires more partitions. Although the algorithm takes the name of isolation forest, the structure at its base is the isolation tree (IT). ITs are data-induced random trees. To build an IT the algorithm consider two random quantities namely a splitting feature *f(v)* and a splitting threshold $\tau(v)$.

Given a dataset $\mathcal{D} = \{x_1, x_2, ..., x_n\}$ of p-dimensional data points each IT say t is assigned to a bootstrap sample $\mathcal{D}_t \subset \mathcal{D}$ and carries out an isolation procedure based on the split tests associated to internal nodes. Each bootstrap sample share the same size ψ. Data points that belong to $\mathcal{D}_t$ are called *in-bag* samples from the tree perspective. Those subsets of the dataset are recursively partitioned until either all point are isolated, namely it ends up in a leaf, or the IT reaches a predetermined depth limit $h_{max} = \lceil log_2 \psi \rceil$. To each data point $\mathbf{x_i}$ is associated a score:

$$s(\mathbf{x_i}) \quad = \quad 2^{-\frac{\overline{h}(\mathbf{x_i})}{c(\psi)}} \tag{3.1}$$

where the term $\overline{h}(\mathbf{x_i})$ is the mean of $h(\mathbf{x_i})$, which represents the length of the branch from root to leaf node, on the other hand the term $c(\psi)$ is a normalization term

dependent on n that is the number of data points. It ranges in [0, 1] in particular if it tends to "1" the point is classified as anomalous and vice versa around "0" there are “normal” points.

The last step of the isolation forest algorithm labels data points thresholding on the anomaly scores, building two partitions based on the "contamination" level:

- predicted inliers $\mathcal{P}_{\mathcal{I}} = \{x_i \in \mathcal{D} | \hat{y}_i = 0\}$
- predicted outliers $\mathcal{P}_{\mathcal{O}} = \{x_i \in \mathcal{D} | \hat{y}_i = 1\}$

Isolation Forest not only detects anomalies faster compared to other AD algorithm, but it also requires less memory and this is one of the characteristics that makes the algorithm applicable in an IoT scenario. In a nutshell, this happens because Isolation Forest isolates anomalies in the data points instead of profiling normal data points. As anomalies data points mostly have shorter tree paths than the normal data points, trees in the isolation forest does not need to have a large depth so a smaller max_depth can be used resulting in low memory requirement. By its nature, Isolation Forest exhibits enviable performance when working with large or redundant data as it is very efficient. stFalling in the specific case another strong point is due to the fact that the model is always the same and does not require particular tuning, on the other hand the use of neural networks would require a particular tuning for each machine, which makes everything more expensive.

Besides that there are two other problems that can be encountered when dealing with anomaly detection: swamping and masking. Swamp is a situation of misidentifying normal instances as abnormal, which can occur when normal and abnormal instances are close to each other. Masking is the situation of mistakenly identifying anomalous instances as normal, which frequently occur when they are collectively in a dense area, in order to “hide” their presence, as described in Sec. 3.1. Subsampling in isolation forest allows to build a model that is resistant to these two effects.

In short, the characterizing aspects associated with isolation forest are:

- The characteristic of isolation trees enables to **exploit subsampling**, something that it is not feasible in all the other existing methods.

- isolation forest exploit **no distance or density measures** to detect anomalies. This eliminates a major computational cost of distance calculation in all distance-based and density-based methods.
- isolation forest is **efficient** as its execution requires a linear time complexity with a small constant and a minimal memory requirement; it is an algorithm with constant training time and space complexities.
- isolation forest has the capacity to **scale up** to handle extremely large datasets and high dimensional feature vectors.

3.4.3 Results

The isolation forest is a classic machine learning algorithm based on a random tree so the parameters that control its structure are easily understandable. For training the isolation forest the parameters that have to be considered are:

- **number of isolation trees** (n_estimators), the number of IT employed by the algorithm
- **number of samples** (max_samples), the number of samples to draw from the dataset to train each estimator (*isolation forest sampling*)
- **number of features** to draw from the dataset to train each base estimator (max_features)
- **bootstrap**, used to select if the bootstrapping is done with or without replacement
- **n_jobs**, can be used to scale up the training by making use of more workers
- **contamination** of the data set, i.e. the proportion of outliers in the data set

The PyOD library[1] was used for the implementation, but a Sklearn one is also available[2], the parameters in both cases are more or less the same, only slight alterations on the denominations. The model was trained on the previously representations, with contamination value set to 5%, using 100 isolation trees as default. The results obtained can be observed in Fig. 3.3, making use of UMAP and PCA respectively. From the results obtained, it can be observed that the algorithm performs its task, isolating the outliers which are highlighted in red while normal points are

[1] https://pyod.readthedocs.io/en/latest/_modules/pyod/models/iforest.html
[2] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

color in grey. Consider that to train the model about one year of data coming from a single machine is used, the training duration is in the order of a minute or less so even without scaling up, excellent results are obtained considering the amount of data.

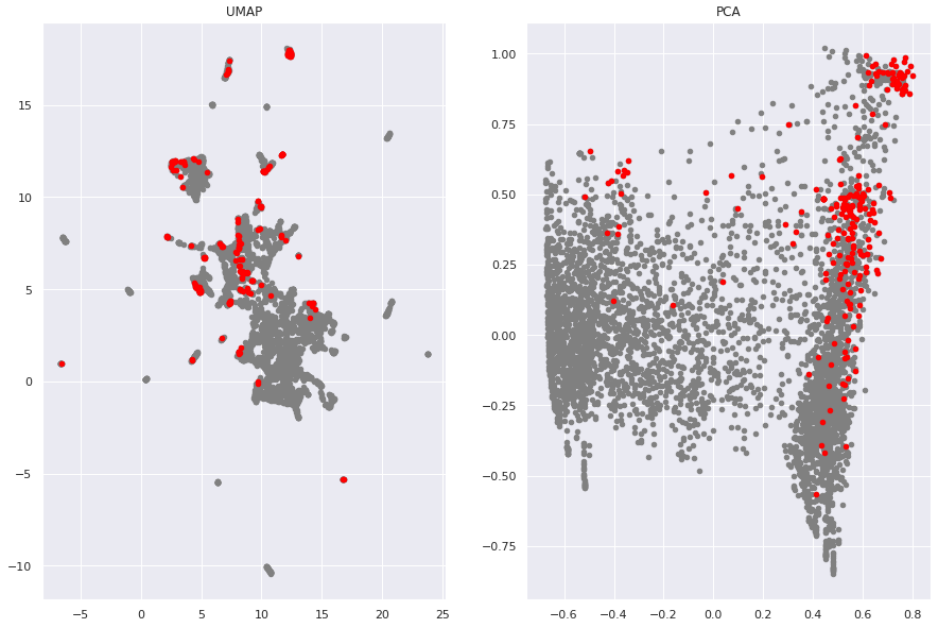


Figure 3.3: IForest results visualization example

Obviously this type of graph only provides a visual intuition of the result, to understand well, it is useful to observe the anomaly score distribution, the result in question can be observed in Fig. 3.4. Looking at the histogram some insight about the outliers distribution can be obtained, in particular one can observe that the algorithm perform quite well breaking the range of variation of the anomaly score in a fairly defined way. From the results it can be seen that the vast majority of points are classified with a low score namely as "normal", while the more the score increases and one moves to the right in the histogram, the points are classified as anomalous, as said the separation between anomalous and normal is well defined. To create the histogram in the figure, the *decision_function (X)* function was used, due to this the resulting anomaly scores are remapped to [-0.5, 0.5], because it is the standard for all the AD algorithms of the library, so it differs from the original anomaly score [0, 1] interval, but this is only a clarification because the concept is the same.

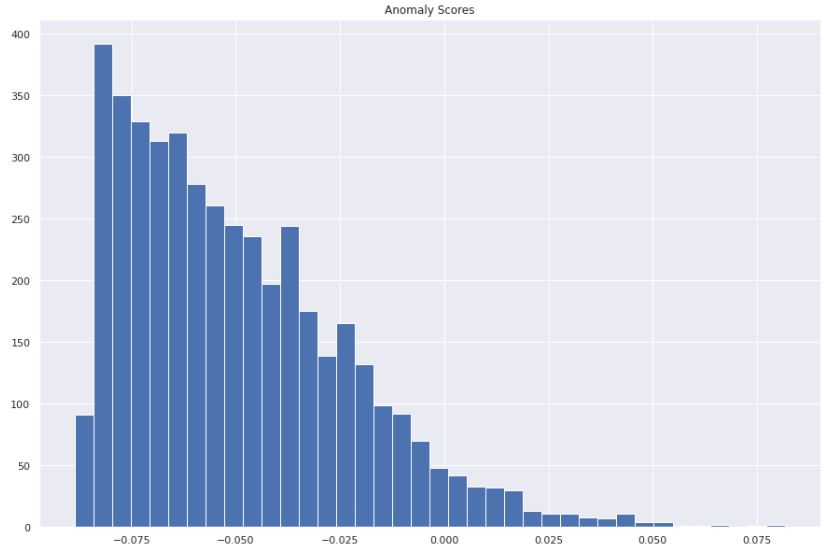


Figure 3.4: Anomaly score

3.4.4 AD results and considerations

In the following paragraph, it is explained how the results of the anomaly detection process are analyzed and "validated". As previously mentioned, the isolation forest model has been trained using all the defined representations, in particular a year of data is used with an hourly time window using data extracted from a single machine. To analyze the results, the process is composed by two steps, the first step is performed with no knowledge about the machine, an intuitive analysis is done using only data knowledge and then searching for recurring patterns or intriguing statistics, if something is discovered then move to the second step, the results have been passed under the attention of domain experts, who have confirmed whether or not their labeling as anomalous or not or in general some feedback were provided. This two-step feedback process permits to have a little “supervision” even if everything is done in a completely unsupervised way. The evolution of the representation is the result of this validation process, in particular of the exchange of information with the domain experts which has led to a final representation that tries to cover the most critical aspects from the point of view of the machine and also go to help the detection algorithm. The first three representations turned out to be quite inconclusive as despite some fairly recurrent structure was recognized, no abnormal behaviors were noted. The consequence of this was to think to a more complex representation, as the fear was that the first three representations were a bit sparse and because of this not able to carry enough information. The previous

deductions lead to the construction of representations 4 and 5, which contain much more information.

Starting from representation 4, what have been notice is that recurring structures emerge much more easily, both labeled as normal and as anomalous. In particular, what is observed is very interesting. Points classified as normal have a very low "#changes" value and typically the state in which the machine has spent most of its time is that of production, recall that this came out also in Sec. 2.2.2, in which the analysis on the average behaviour of the machine shows that the machine stay mostly in *production* state, while the information about the state changes was not detected in any of the previous studies. On the other hand, points classified as anomalous show a high "#changes" value and the composition of the time window is also very diversified. This then identifies strange sequences as sequences in which the machine state hops in various configurations frequently, this is consistent with the intuition as if in a single time window many changes are observed one can think of something that is actually strange. The results in this case were confirmed by the domain experts, in addition to what can be observed simply without knowing anything about the machine, they confirmed that the combinations of alarms associated with the anomalous sequences were actually correlated or not.

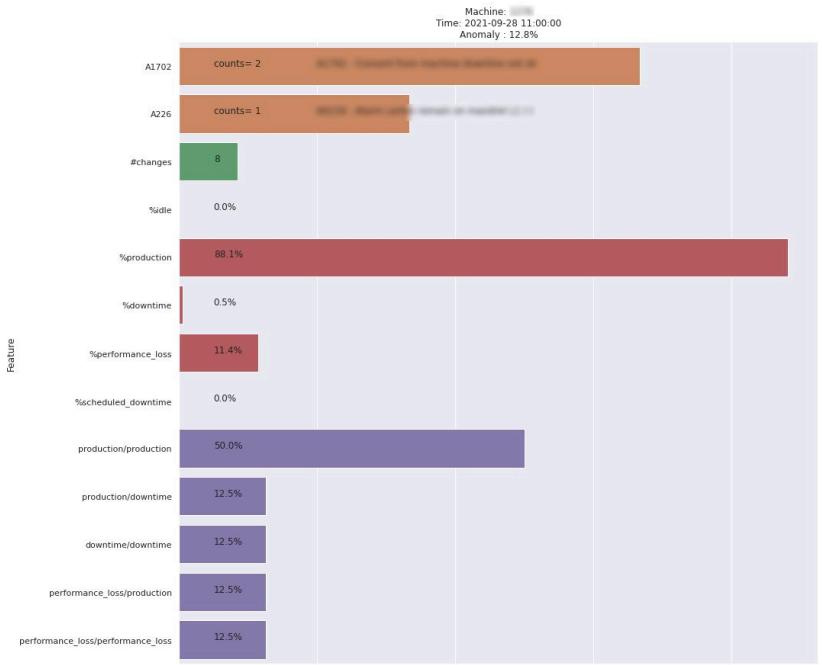


Figure 3.5: Normal point example

It is important to point out that there exists two family of high "#changes" value points, the first group contain all the points in which the machine hops from one state to another very frequently, while the second group include all the point in which the machine has a high value of state changes, but the state is always the same. This second situation could seem strange but it is very common as it describes a situation in which the machine restart multiple times, so basically it does not change its state.

The construction of the last representation is the result of this last analysis in which it is noted that the number of state changes is important but must be differentiated since, as already said, situations can be found that have a structure similar to an anomaly but which they are not due to the type of status change. Therefore, the content related to transitions is introduced, which does not seem to provide a great improvement, however it certainly helps to increase the precision of the detection, "forcing" the algorithm to observe certain aspects more carefully.

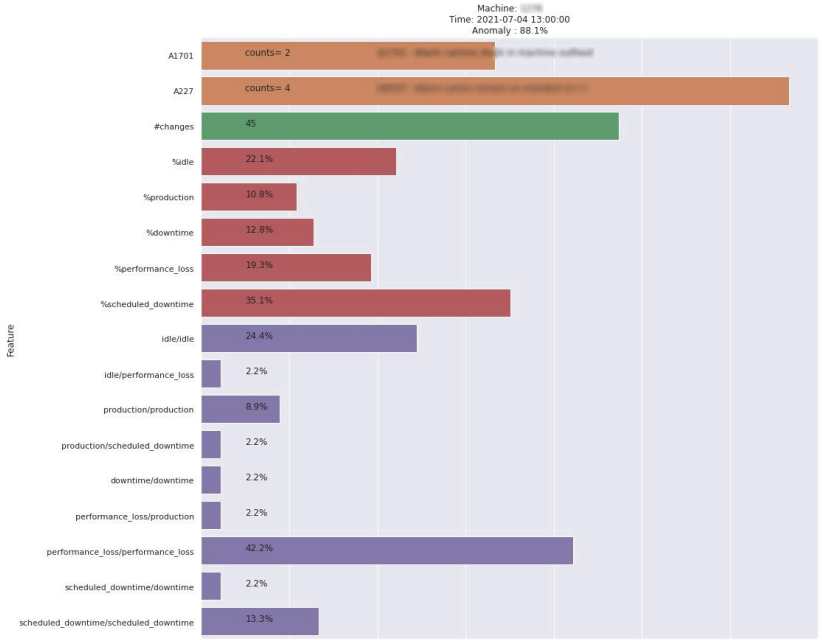


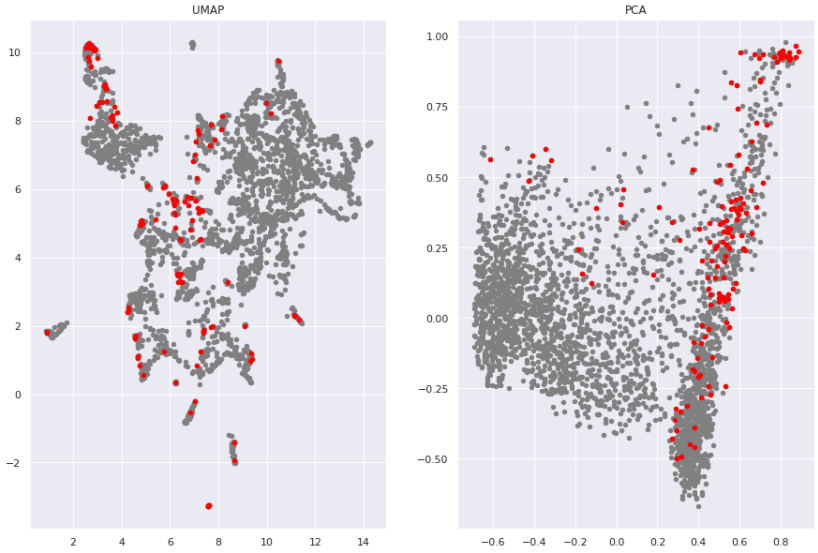
Figure 3.6: Anomaly point example

In Fig. 3.5 and Fig. 3.6 two examples can be observed, in which representation 5 is used, respectively a normal and an anomalous point, in which it is possible to observe what has been highlighted. No points relating to representation 4 are reported, as the result is almost the same for the two images provided if the part of the representation relating to the state transitions is removed. While the first

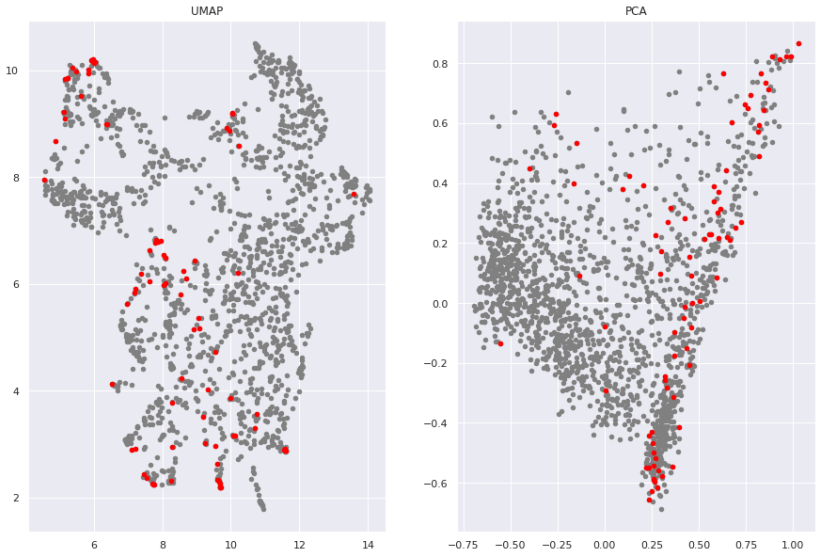
three representations are not reported in the form of a bar graph as they did not lead to actual results but were a fundamental step to understand how to enrich the representation. In the App. A other examples are available. The description of the alarms in the images has been obscured for confidentiality reasons.

Different time resolutions

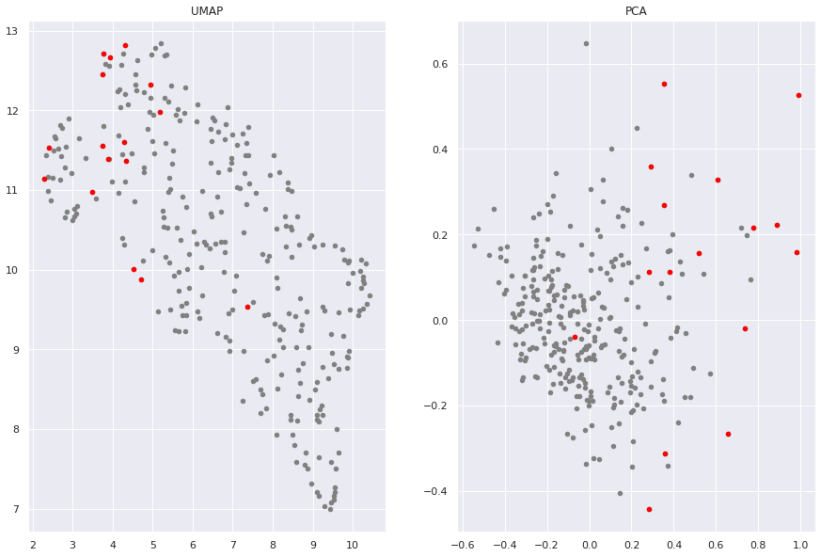
As already mentioned in Sec. 2.3.4 the amount of data that is necessary for the algorithm to work without leading in underfitting conditions is a point that cannot be neglected. As seen before the type of data does not change a lot throughout the life of a machine as the machine always performs very similar tasks. The problems arise for the learning algorithm when the amount of data is not enough, although isolation forest is a classic machine learning algorithm it can fail if the data is extremely few. One of the characteristics that the developed AD system must have is that of being able to use different resolutions, as already mentioned above, the use of resolutions with "large" time windows can be dangerous. In the following are reported the AD results visualization using time window respectively of 2H,4H and D. The amount of data considered is always one year, the visual effect is clear enough to show how the amount of sequences considered decreases as the time window size grows. The consequence of that is that the learning algorithm suffers a bit the data scarcity and the results obtained with the time windows set as daily appears strange.



(a) 2H time window



(b) 4H time window



(c) D time window

Figure 3.7: AD results visualization

In Fig.3.8 an example of anomaly extracted with daily resolution is shown, this sequence is labelled as anomalous with a very high score but on the other hand it seem quite "normal". In conclusion, it is necessary to carefully analyzes the amount of data available before proceeding with the AD procedure. This topic is extremely important in practice, because the algorithm have to deals with a fixed amount of data and it have to be updated in order to guarantee the functioning of the isolation forest. In practice, the strategy adopted to tackle this problem is the one of oversampling in order to obtain more sequences even though the data quantity is fixed. The description of the alarms in the image has been obscured for confidentiality reasons.

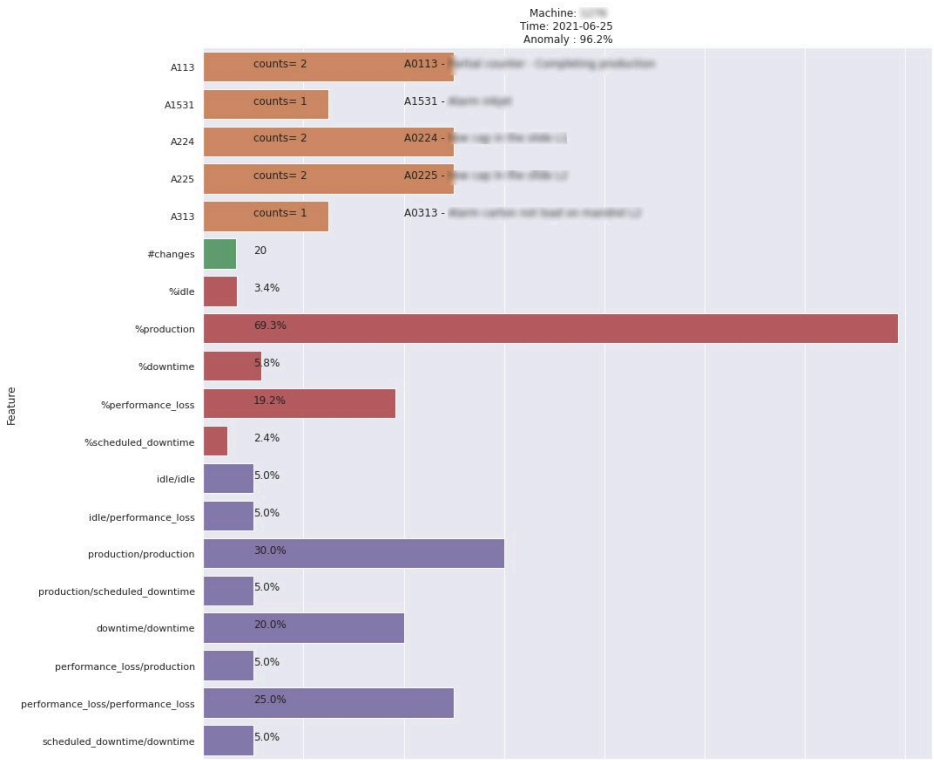


Figure 3.8: Anomaly point example

Test on other machines

The analysis of the development of the AD system described in this thesis mainly refers to a single machine, this in addition to providing coherently linked results appears necessary since extracting and analyzing data coming from several machines blend all data together is not a good choice. The explanation for this is due to the fact, as already explained, that although the machine model is always the same then

it is adapted to the customer's requests, every customer use different recipes or different kind of products, this means that each machine performs similar functions but not exactly the same. Recall that already consider only one machine embeds a lot of variability due to factors such as the recipe, the speed, the format an other small changes that make the machine's task vary a lot. In conclusion, using data from multiple machines at this level is dangerous if consistent results are to be obtained.

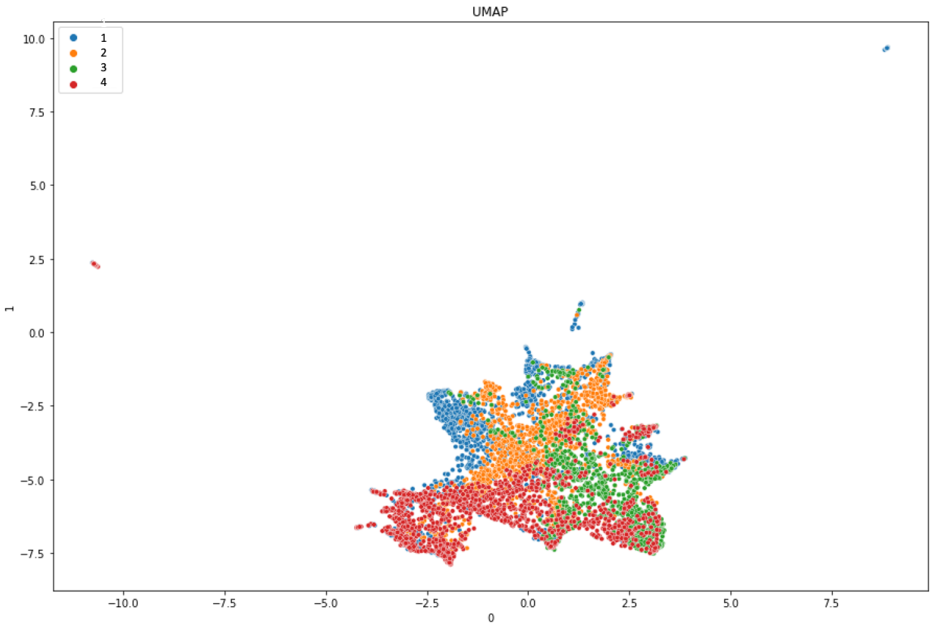


Figure 3.9: UMAP representation of multiple machines

However, it is important to provide complete results in order to test the anomaly detection system on other machines. Some tests were performed on four other machines. The representation used is the fifth exploiting the normalized counts, the time windows chosen were those of one hour and two hours. In conclusion, it can be said that similar results are obtained testing the system on different machines but at the local level things are a bit different. In Fig. 3.9 the previous exposed concepts can be visualized, the visualization consider data obtained from four different machines, those data have been featurized and then plotted using UMAP with a focus on the global structure. The result are interesting, because they confirm the fact that the use of blended data can be dangerous or misleading, in the sense that as it can be observed that the four representations gather together and the division among them is very unambiguous. In conclusion, results that are obtained from one machine cannot be exploited for the other unless the situation allows it, namely the

machines have to perform exactly the same function. Maybe this could be used for future developments in going to optimize the work of a machine by comparing it with other machines to improve performance, perhaps by changing speed, the type of formats or other.

Data quality lack

Another interesting point that was revealed by the AD results is the one associated to the data quality lack. This problem is linked with the data but the isolation forest empathize it because of how it works, recall that the algorithm label as anomalous points that deviate significantly from the rest of the data then if a sequence appear as very rare that for sure it will be labelled as anomalous. The problem with this is that anomalous does not mean faulty from the standpoint of the machine.

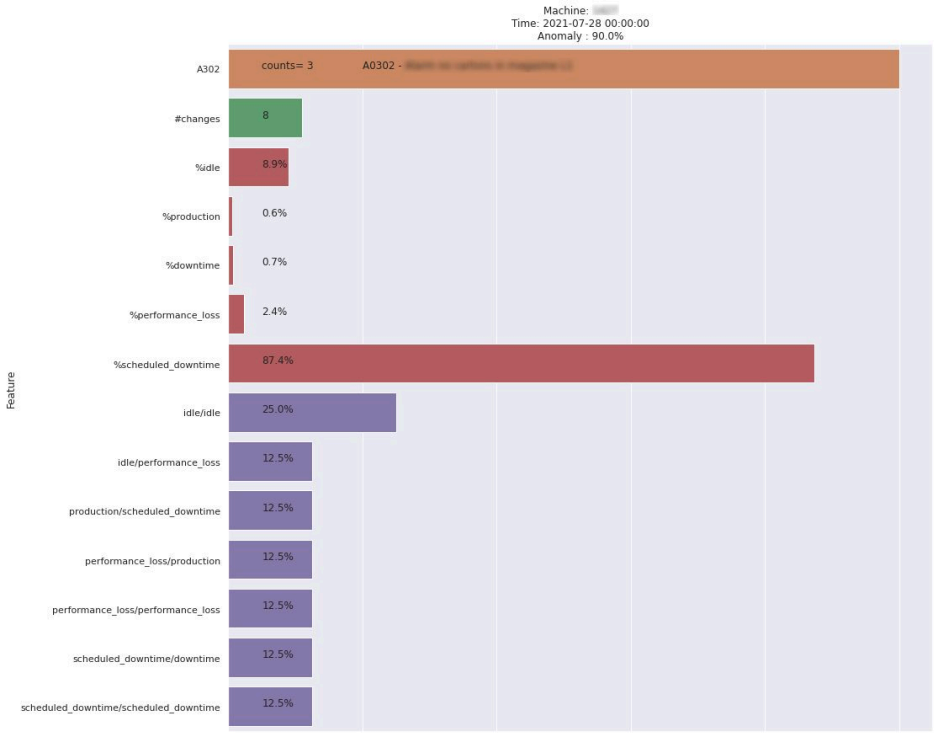


Figure 3.10: Example of misleading sequence

In Fig. 3.10 it is shown one case that highlights this problem, as it can be observed the sequence has a very high anomaly score, looking at it one can observe that the value of the feature *%scheduled_downtime* is very high, with some domain knowledge one can understand that there is nothing of anomalous because the machine is cleaning itself. The algorithm label this point as anomalous, the problem is

linked to the scarcity of this kind of data that will make this sequence rare and so, strange. To sum up, it is important to check the data quality, or in order to alleviate the problem one can show to the algorithm more sequences similar to this in order to decrease the rareness of this sequence. The description of the alarms in the image has been obscured for confidentiality reasons.

3.5 HDBSCAN

HDBSCAN is a clustering algorithm developed by Campello, Moulavi, and Sander (McInnes, Healy, and Astels, 2017). It extends DBSCAN by converting it into a hierarchical clustering algorithm and then using a simple but working idea to extract a flat clustering based on the stability of clusters.

In the following an intuition about how HDBSCAN works will be provided. Starting from the fundamentals, the problem that has to be solved is to perform clustering on data in which the clusters have different shapes and sizes and can also have very different densities and furthermore could be corrupted by noise or some kind of outlier. Tackling the problem with a classic algorithm such as k-means surely does not get good results as this algorithm exactly assumes that the clusters have certain geometric properties and moreover it takes a predetermined number of clusters. HDBSCAN approaches the problem does not look for clusters with a predefined shape, but it looks for regions in the data that are denser than the surrounding space, it behaves basically as a density clustering algorithm. Formally defining what a cluster is can be misleading or in any case strictly context-specific, with this clarification in the HDBSCAN context a cluster is an highly dense regions separated by sparse regions, a visual insight can be provided by fig.3.11.

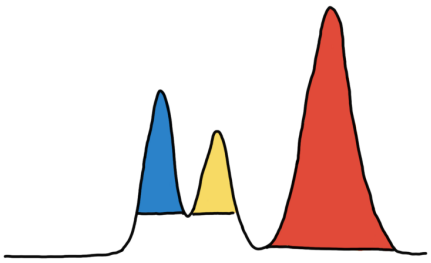


Figure 3.11

The fact is that in practice this type of well-defined distributions are nowhere to be found, but something a little more convoluted as shown in Fig. 3.12 is faced, however here it is not so easy to understand what can be defined as a cluster or not. To cope with this, the elder brother of the algorithm in question, DBSCAN sets a resolution to have a clear definition of what is or is not a cluster. The fact is that this doesn't work well for clusters with different densities. To get rid of this problem,

HDBSCAN first builds a hierarchy to figure out which peaks end up merging together considering also the order, basically by getting multiple level-sets at different values of λ as in Fig. 3.12. Some may naively think that this is done by running DBSCAN multiple times, however this is not exactly the case but it approximates well the basic functioning, it also can be useful to visualize what the algorithm does. So if basically it behaves as a density based method, in the second step it exploits the hierarchy to cluster data, so as point out HDBSCAN is not a standard clustering method, indeed it can be defined as a mix between a density based and a hierarchy based algorithm.

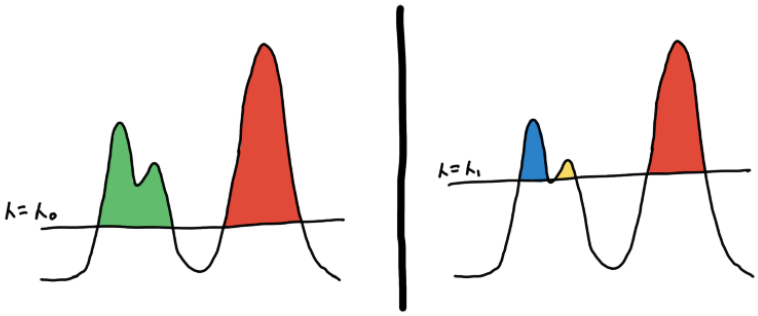


Figure 3.12: Running DBSCAN with different resolutions

To choose if is it better to keep this cluster or split it up into its sub-clusters, it looks at which one "persists" more. Where the persistence measure that is associated to the cluster stability is represented by the areas of the different colored regions in the hierarchy plot. So exploiting this artless idea, HDBSCAN is able to decide how to cluster together points.

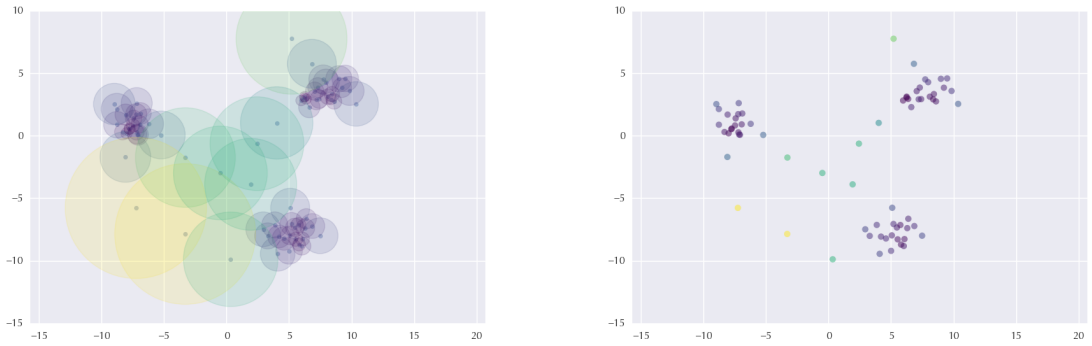


Figure 3.13: HDBSCAN

3.6 HDBSCAN: interpretability

After what has been explained in the previous section, obviously it is not so obvious to understand where this method can be exploited in the case in question. Despite the vast generality linked to clustering techniques, in this case the approach serves as a "interpretability" technique to try to extract a summary structure of the points classified as anomalous from the anomaly detection procedure. After several attempts with other clustering techniques, some problems linked with the data has arisen. The data treated has substantially two problems, it is strongly scattered and the dimensionality is high, in particular the p features are in general around the thousand since this is the number of available alarms. A small summary scheme on clustering techniques can be observed in Fig. 3.14.

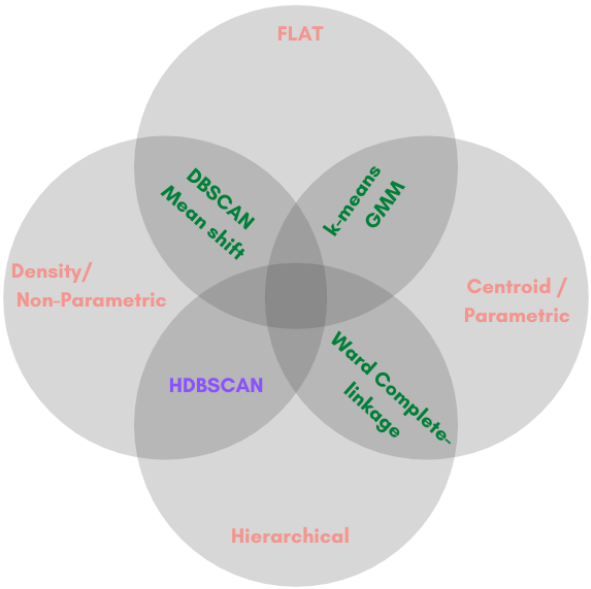


Figure 3.14: Clustering algorithms

The use of techniques such as k-means that make assumptions about the type of distribution or shape of the cluster are certainly to be discarded a priori given the type of data and its sparsity. A different reasoning occurs regarding hierarchical clustering techniques, since these are not suitable for large datasets due to high time

and space complexity. Therefore, density-based techniques remain outside, however here the problem remains that they present resolution parameters that cannot be tuned without some information on the data or concrete display that cannot be obtained because all the visualization are strong approximation of the data distribution. Recall also that highly sparse data does not help those kinds of methodologies. Therefore, the use of HDBSCAN which is quite "unsupervised" compared to the other algorithm is a good choice, as shown by the results, even if it is placed in front of large and high dimensional data it provides interesting results, obviously without scaling up the whole process. It must be point out that also HDBSCAN fails with very high dimensional data, the official documentation point out that the optimal number of features lies in the range beetween 50 and 100[3], from what was said before there seem to be some inconsistencies but this is because in general there are thousand of alarms and so thousand of features but in a single machine the amount of alarms that are actually observed are much less, however a number that lies within the optimal range for HDBSCAN.

After this extensive introduction on why HDBSCAN was preferred over other methods, the following section illustrates how it was used and the results obtained.

To extract a summary structure of the anomalies the whole dataset is used, the isolation forest model is trained on it, as before the contamination level is kept to 5%. At this point, the isolation forest has classified a 5% of the data as anomalous. Afterwards, HDBSCAN is run on this anomalous subset, the clustering results can be observed in Fig. 3.15, the extracted cluster are nine and it is also interesting to observe that many points are labelled as noise. Recall that the aim of this procedure is about interpretability, points grouped together in cluster represent families who share similar characteristics providing standard structure anomalies that can be used as comparison with new points or can describe the whole anomalous behaviour of a machine.

[3]https://hdbscan.readthedocs.io/en/latest/faq.html

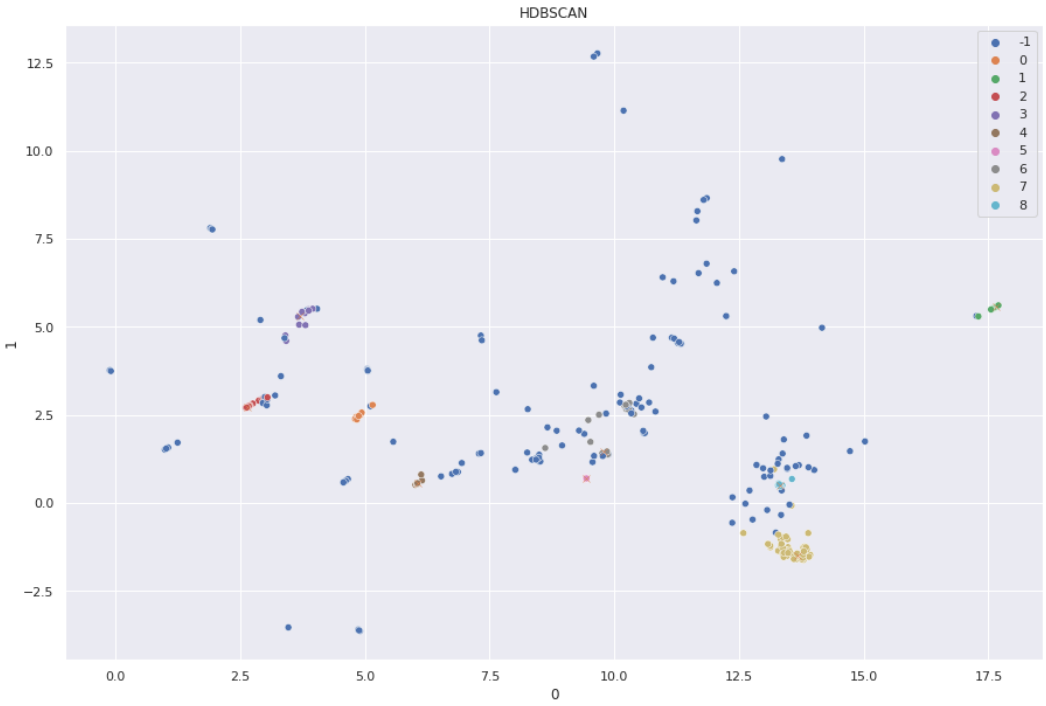


Figure 3.15: HDBSCAN results on anomalies

The medoids of each cluster were analyzed, the medoid in this case represents the typical structure that the points have in a certain cluster, thus obtaining what was first called the summary structure of the cluster.

Obviously, another possible choice would have been to use the centroid, however the centroids were not used as they would have formed "synthetic" centers not linked to the real datum, which would have had a logic only in a mathematical sense. The use of the medoid allows to find the real point closest to the center without adding other errors since the data is already very complicated to summarize.

Chapter 4

Interpretability

This chapter deals with the issue of interpretability. It is divided into two macro-sections, the first introduces the concept of interpretability and the techniques used, describing how they work and their pros and cons, while the second part deals with the use of these techniques, in particular the central part concerns the evaluation of representations obtained through these tools.

4.1 Interpretability in Machine Learning: a brief introduction

Machine learning models can be surprisingly good at making predictions, but they often cannot provide explanations for their predictions in terms that humans can easily understand. The number of characteristics from which they draw conclusions can be so huge and their calculations so complex, that it may be impossible to determine exactly why an algorithm produces the answers it provides.

This is why the concept of "interpretability" is introduced (Molnar, 2022). There is a common misconception in the context of machine learning between explainability and interpretability. While they are closely related, it's worth eliminating the differences, if only to see how complicated things can get once you start delving deeper into machine learning systems.

Interpretability concerns the extent to which cause and effect can be observed within a system. Or, to put it another way, it is the extent to which you are able to predict what will happen, given a change in the input or algorithmic parameters. It is the ability to look at an algorithm and say yes, I can see what's going on here.

Explainability, meanwhile, is the extent to which the internal mechanics of a deep learning machine or system can be explained in human terms. It's easy to miss the subtle difference with interpretability, but consider it this way: Interpretability is about the ability to discern mechanisms without necessarily knowing why, while Explainability is being able to literally explain what's going on.

Why interpretable results? The title of the paragraph is associated with a question that naturally arises when a problem is observed from this point of view. Interpretability is crucial for several reasons, currently must compensate for incomplete interpretability with judgement, experience, observation, monitoring and diligent risk management including a thorough understanding of the datasets used. The need for interpretability arises from an incompleteness in problem formalization, which means that for certain problems or tasks it is not enough to get the prediction but the model must also explain how it came to the prediction, because a correct prediction only partially solves your original problem. So in a schematic way not only the *what* but also the *why* and the *how* are important.

Interpretable machine learning techniques can generally be grouped into two categories (Molnar, 2022): *intrinsic* interpretability and *post-hoc* interpretability, depending on the time when the interpretability is obtained. Intrinsic interpretability is achieved by building self-explanatory models which integrate interpretability directly to their structures. The family of this category includes decision tree, rule-based model, linear model, attention model, etc. In contrast, the post-hoc one requires creating an external model to provide explanations for an existing model.

In a nutshell, intrinsically interpretable models typically are a simpler model and can be interpreted by looking directly at their internal parameters, of course simplicity make easier the interpretability part but on the other and make those model less performing. The post-hoc ones requires an external model to deal with interpretability due to their complex structure that cannot be explained by looking at the internal parameters, consider for example a deep convolutional neural network. So conversely to before high performance model are typically more opaque than simpler models.

Similarly, but not equally, *model-specific* or *model-agnostic* methods can be treated. As the names suggest, model-specific interpretation tools are limited to specific model classes, while model-agnostic tools can be used on any machine learning model.

Based on the above categorization, further differentiation split the interpretability types in two: *global* interpretability and *local* interpretability. Global interpretability means that users can understand how the model operates globally by inspecting the structures and parameters of a complex model. On the other hand, local interpretability locally examines an individual prediction of a model, trying to figure out why the model makes the decision it makes. These two types bring different benefits. Global interpretability could clarify the inner working mechanisms of machine learning models and thus can increase their transparency. Local interpretability will help uncover the causal relations between a specific input and its corresponding model prediction. Those two help users trust a model and trust a prediction, respectively, providing explainability and interpretability. Since the other main topic of the thesis is about anomaly detection, it is also interesting to study the link between the latter and the interpretability. Anomaly detection is a machine learning task that can be considered in general unsupervised, a tool as the interpretability can be very useful, consider for example a local interpretation about a point classified as anomalous, this can help the user to perform Root Cause Analysis and subsequently identify suitable corrective actions, if necessary. On a design level, interpretability can be useful to evaluate in a more quantitative way the algorithms employed and much more.

4.2 Methods

4.2.1 AcME

AcME is novel a methodology, called **Accelerated Model-agnostic Explanations** (Dandolo et al., 2021), it works with each regression or classification model, providing both global and local interpretability, and at the same time it is also computationally efficient, a requirement that in a IoT scenario is wonderfully desirable. Among the interpretability methods that provide both global and local explanation there is **SHapley Additive exPlanation (SHAP)**. This technique is quite popular and provided with some specific extensions in order to work better adding some model informations. The fact is that due to its logic it is computationally burdensome and it is also quite complex from a theoretical point of view. SHAP is considered state-of-art, in the following many comparisons will be performed with AcME reporting the results obtained in the paper.

Computational speed is a factor that must kept into consideration but also the amount of data and the way it process is a main factor, this becomes even more important in an IoT context where the amount of data is considerable, often in the form of live streams with extremely fast update. Besides, a fast update of data leads to a more frequent retrain of the machine learning models, and when a model changes, it is necessary to rerun the entire interpretability procedure, which worsens the computational burden associated to SHAP.

AcME basically perform a sensitivity analysis relying on perturbation of the data based on quantiles of the empirical distribution for each features, perturbations w.r.t. a so called *baseline vector* x^b. A *baseline vector* x^b is used to deal with both local and global interpretability, the choice of this vector changes a bit in the two cases.

How it works ?

Let is start by treating first global interpretability. Assume that the dataset is made of data points composed by p features, the procedure starts by computing the baseline vector:

$$\mathbf{x^b} = \overline{\mathbf{x}} = [\overline{x}_1, \overline{x}_2, ..., \overline{x}_j, ..., \overline{x}_p] \tag{4.1}$$

For global interpretability it is composed by the average point computed on the dataset. Then it creates a varaible-quantile matrix $Z_j \in \mathbf{R}^{Qxp}$ where Q is the selected number of quantiles that will be used to evaluate the importance of each feature. The rows of such matrix coincide with the baseline vector except for one component which is substituted by the Q quantiles.

$$\mathbf{Z_j} = \begin{bmatrix} \mathbf{z}_{j,0} \\ \mathbf{z}_{j,1/(Q-1)} \\ \vdots \\ \mathbf{z}_{j,1} \end{bmatrix} = \begin{bmatrix} \bar{x}_1 & \bar{x}_2 & \dots & x_{j,0} & \dots & \bar{x}_p \\ \bar{x}_1 & \bar{x}_2 & \dots & x_{j,1/(Q-1)} & \dots & \bar{x}_p \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \bar{x}_1 & \bar{x}_2 & \dots & x_{j,1} & \dots & \bar{x}_p \end{bmatrix} \tag{4.2}$$

To be more robust, it can be avoided the use of quantile 0 and 1, limiting the range of Q to an interval that excludes the possible outliers, typically Q is made range from 0.1 to 0.9. At this point the predictions associated with the variable-quantile matrix

are computed :

$$\hat{\mathbf{y}}_j = \begin{bmatrix} \hat{y}_{j,0} \\ \hat{y}_{j,1/(Q-1)} \\ \vdots \\ \hat{y}_{j,1} \end{bmatrix} = \begin{bmatrix} f\left(\mathbf{z}_{j,0}\right) \\ f\left(\mathbf{z}_{j,1/(Q-1)}\right) \\ \vdots \\ f\left(\mathbf{z}_{j,1}\right) \end{bmatrix} \tag{4.3}$$

By comparing the synthetic predictions associated with the rows of the variable quantile matrix with the prediction associated to the baseline vector, an importance score representing the relevance of the j-th feature can be computed.

$$I_j = \frac{1}{Q} \sum_{q=1}^{Q} \left|\Delta_{j,q}\right| \tag{4.4}$$

$$\Delta_{j,q} = \frac{\hat{y}_{j,q} - f\left(\mathbf{x^b}\right)}{\sqrt{\text{var}\left(\hat{\mathbf{y}}_j\right)}} \left(\max\left(\hat{\mathbf{y}}_j\right) - \min\left(\hat{\mathbf{y}}_j\right)\right) \tag{4.5}$$

Where the term $\Delta_{j,q}$ is defined as standardized effect, close to the well-known standard score in which the second multiplicative factor accounts for the overall impact of the variable in terms of how the applied perturbations spread out the predictions. Indeed, it is reasonable that the wider the range of changes in the prediction, the more relevant the variable is for the model. For the interpretation of individual predictions, i.e. the case of local interpretability, the baseline changes from $x^b = \overline{x}$ to $x^b = x^*$, namely it is set equal to the specific data point x^*.

AcME is typically less expansive in terms of computing powers, since it only needs to apply the model on Qxp observations. The number of samples N only affect the computations of the quantiles, but the effect is not comparable to what happen in SHAP.

Given ad dataset of N points, for each data point x_i KernelSHAP samples K coalitions, where K is a parameter that depends on p. Then, it applies the model that has to be interpreted to each coalition mapping each group to the original feature space. Finally, KernelSHAP uses the K predictions as a training dataset to fit a local linear model, than once the Shaply values are obtained an importance measure can be provided.

AcME: Anomaly Detection interpretability tool

Another feature that has increased the interest in AcME is the fact that the algorithm can be used in an AD scenario, in particular an unsupervised scenario in which each data point is associated with a score that quantifies the abnormal behavior of themselves. Exploiting the local interpretability, we could focus on a sample and use AcME to evaluate how changes in the input feature values would impact the corresponding anomaly score.

This kind of analysis can be performed thanks to the nature of AcME. In order to analyze the feature importance in terms of AD, the model is fitted locally, so provided a data point it builds the feature importance table associated. The table contain the predictions associated to the variable-quantile matrix, which are expressed in terms of anomaly score. From this table, the features that lead to a change in anomaly score such that a point changes from anomaly to normal or vice versa are taken and return. As example, consider the predictions associated to only one feature if the there is a change of sign in the anomaly score predictions this means that such feature is able to change the classification of such point otherwise it is not.

The visualization can be a bit complex, but it is substantially very simple. The output graphics of this tool allow you to view the trend of the anomaly score according to the value of a certain feature. So when the value of the feature "j" vary, this variation is displayed on the axis y, the corresponding value of the anomaly score can be observed in x. In addition to this, the view adds the colors to discriminate between values that lead to the abnormal or normal.

The use of this tool provided by AcME is extremely powerful and interesting, especially if provided as an extra to try to best eviscerate the results of the AD algorithm that run on the machine.

The results obtained using the tool on Representation 4 will be illustrated below, obviously only some examples will be proposed of all the cases analyzed. Recall that the Representation 4, as described in Sec. 2.3.2, is composed of two parts, the first one is linked to the alarms while the second one describe the time window composition. The points observed using this tool can be divided into three families according to their behavior: extremely anomalous, strongly normal and “hopping” points.

Regarding the first category we find points with an extremely high anomaly score, vice versa the points that are in the second category have a very low score. The points that are most interesting to observe with this tool are the points of the third category, the elements that compose this family are quite peculiar, they are not too much anomalous but not enough normal, any small change can cause the point to fall into one of the other two categories.

Let's analyze the results starting from the first category, those points are extremely anomalous there are no values of their features that can make them more "normal", this is quite an interesting point since this means that the algorithm is strongly "convinced" that those are anomalies. The most interesting results were obtained by observing the other two categories. Points belonging to the second category tend to have a very similar behavior, they have some features that in some way can make the point anomalous. What typically happens is shown in Fig. 4.1, the point remains normal as long as the value of the feature in question, in this case the alarm A401, takes on very high values, in this way an anomaly is obtained, otherwise the point remains classified as normal. This obviously brings attention, in the specific case in question, to the alarm A401 which in a practical case should be analyzed more carefully. Another interesting thing, referring to the representation that uses counts and information relating to the state of the machine, is that in this case the phenomenon only concerns the features associated with the alarms and not those relating to the state of the machine.

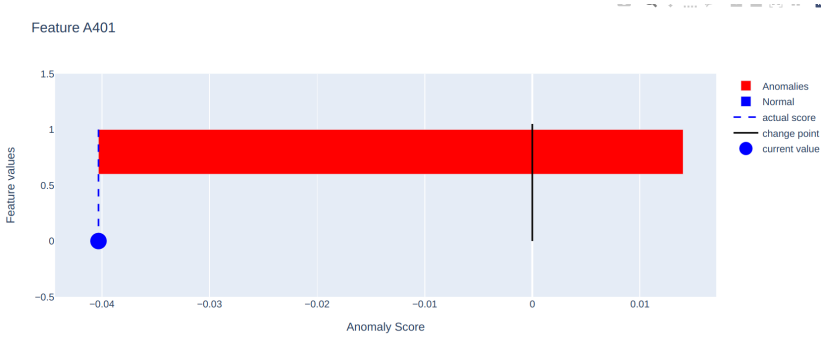


Figure 4.1: Typical behavior for points of the second category

More interesting phenomena can be observed by analyzing the points that have a score close to 0, namely the points that are part of the third defined category. In Fig. 4.2 we find a behavior that is often observed by analyzing this type of points, those present features that could have a very influential effect on the classification

of this type of points, it is observed that typically for low values of these features the points become "more normal" and vice versa if the feature assume higher values. If you want, this is consistent with intuition as, as mentioned, these points are “neither too anomalous nor too normal”.

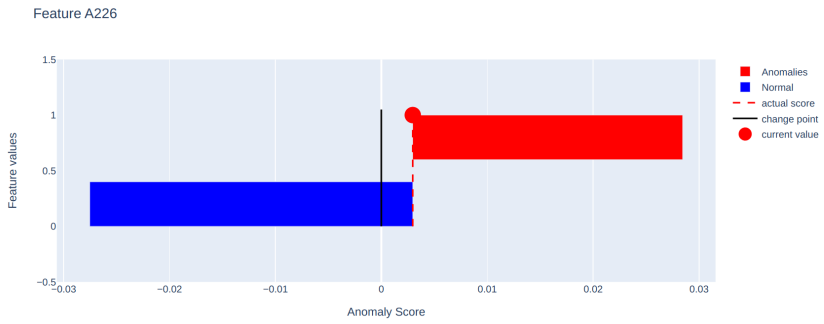


Figure 4.2: Comportamento tipico per punti della terza categoria

Also in this category it seems that the features related to the state of the machine have a greater weight. For example, in some points it is observed that the effect of the "#changes" feature which, as mentioned in Sec. 3.4.4, seems to have a considerable effect when it comes to distinguishing anomalies from normal points. In particular, it is observed that for low values of this feature the points are classified as normal and vice versa if the value is very high, it is also noted that when it assumes extremely low values the point risks being classified as anomalous.

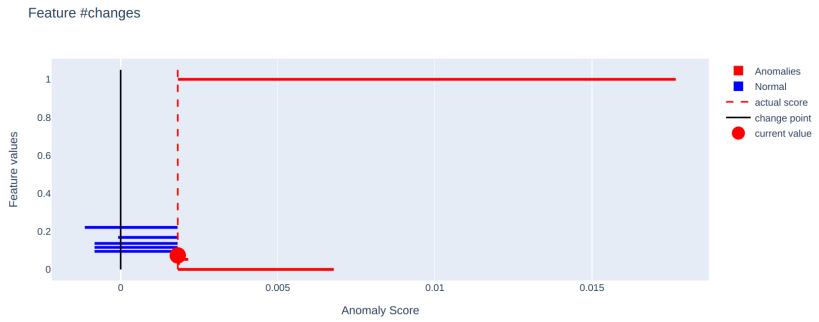


Figure 4.3: Typical behavior for points of the third category

Even when we go to analyze the effect of the “% production_time” feature, we observe interesting things that are partly consistent with what was said in Sec. 3.4.4, that is, for high values of this feature the points have a "normal" behavior, vice

versa the less the machine remain in production the more the behavior is labeled as anomalous.

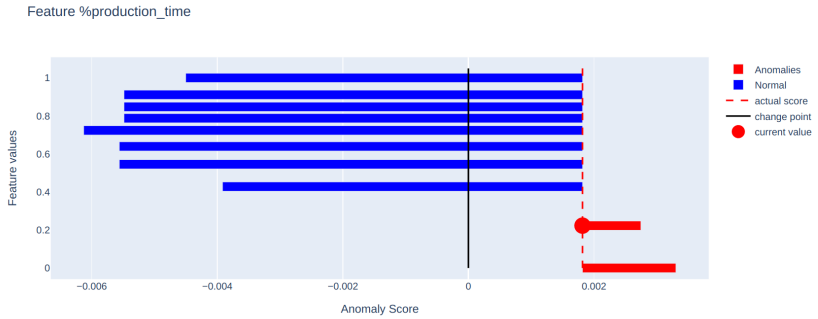


Figure 4.4: Typical behavior for points of the third category

Something similar to the previous considerations can be said for the "% downtime_time" feature, from what can be observed in Fig. 4.5 it is noticed an opposite behaviour w.r.t. the previous analyzed feature, namely high value of this feature classify the point as anomalous. Also in this case the intuition is confirmed by the data, namely more time the machine stops in a single time window more an anomalous situation is detected.

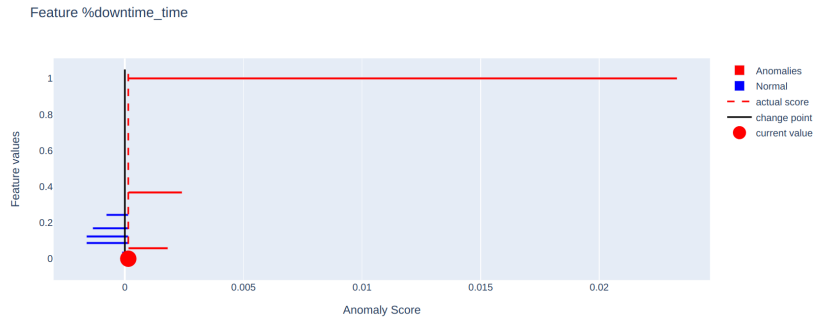


Figure 4.5: Typical behavior for points of the third category

4.2.2 DIFFI

Among the most prominent model-agnostic approaches algorithms that are highly exploited there are LIME for local explanation and PDP (Partial Dependence Plots) to provide information about the global behavior or as last there is SHAP. It must be said that model-agnoistic tools for interpretability typically shows some drawbacks, the main one is associated to the fact that the process that permits to those tools to get some prediction is delicate, as typically it is based on the manipulation of the input and the evaluation of the effect that such changes have on the output. The creation of synthetic data could lead to instability and raising doubts about the actual information conveyed by the method. The name DIFFI means **Depth-based Isolation Forest Feature Importance**(Carletti, Terzi, and Susto, 2020) . The DIFFI approach is based on very basic computations on quantities that naturally emerge from the principles governing the IF model. The nature of such method is model-specific and it tries to reflect the logic inside the IF and a post-hoc method preserving the performance of and established and effective AD focusing on providing global and local explanations. The fact that such method is able to provide both global an local interpretability guarantee a good flexibility.

In the following paragraph a brief explanation of the functioning of the algorithm will be provided. DIFFI relies on two hypothesis which lead to the definition of feature importance in an AD scenario.

A split test associated with a feature deemed as important should:

h1) induce the isolation of anomalous data points at small depths (i.e. close to the root), while relegating regular data points to the bottom end of the trees

h2) produce higher imbalance on anomalous data points, while ideally being useless on regular points

Global DIFFI In order to provide an intuitive point of view of what is inside the algorithm in case of global interpretability, start considering a single Isolation Tree t and associate to it, a subset of the full dataset $\mathcal{D}_t \subset \mathcal{D}$. To visualize it, look at the red square in Fig.4.6. The IF algorithm partition this bootstrap sample $\mathcal{D}_t$ into $\mathcal{P}_{I,t}$ and $\mathcal{P}_{O,t}$, respectively, the partitions associated to inliers and outliers.

At this point define the *Cumulative Feature Importances* (CFIs), those coefficients are associated to the two hypothesis above, in particular CFIs are updates iteratively

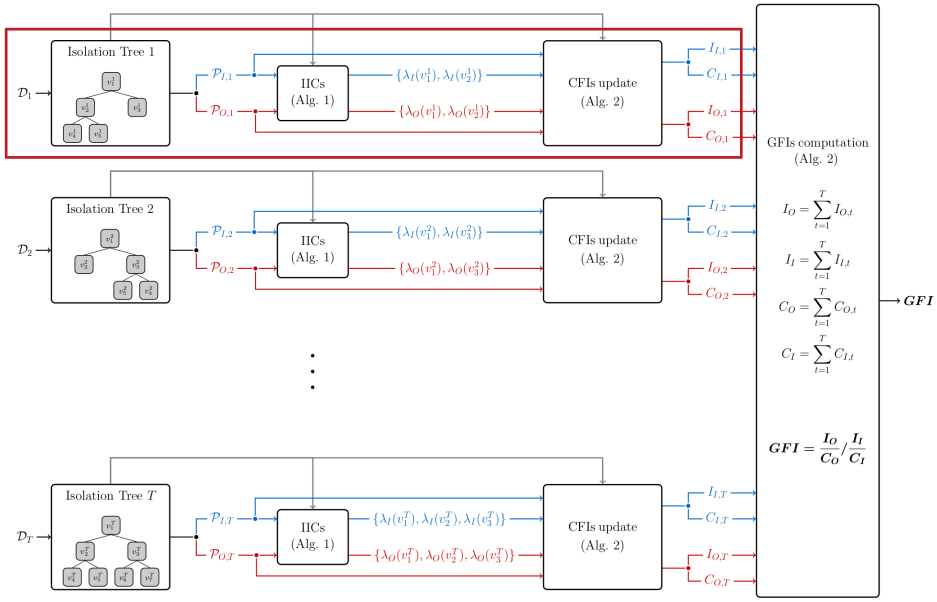


Figure 4.6: DIFFI scheme

by a quantity show in the following:

$$\Delta \quad = \quad \frac{\lambda(v)}{h_t(x)} \tag{4.6}$$

Where the numerator is associated to the hypothesis h2, while the numerator reflects hypothesis h1. Respectively the term $\lambda(v)$ is defined as *Induced Imbalance Coefficient* (IIFs) and measure the strangeness of a split, while the term $h_t(x)$ denotes the depth of the leaf node (in tree t) associated with data point x. At this point extend the reasoning to more trees so in the end the CFIs are averaged all over the forest. The GFI (Global Feature Importance) coefficient is given by:

$$GFI \quad = \quad \frac{\frac{I_O}{C_O}}{\frac{I_I}{C_I}} \tag{4.7}$$

Where the terms C_O and C_I are called *features counter* and have more o less the same purpose of the idf term in TF-IDF.

Local DIFFI At the local level things changes a little due to the fact that in the local case some quantities cannot be computed as the IICs, since only one sample is

considered, and all quantities associated to predicted inliers are neglected, since the focus now is on the interpretation of predicted outliers.

Given a predicted outlier $\mathbf{x_O}$, the corresponding *Local Feature Importance*(LFI) is:

$$LFI(\mathbf{x_O}) \quad = \quad \frac{I_O^{loc}}{C_O^{loc}} \tag{4.8}$$

where C_O^{loc} and the I_O^{loc} have similar meaning as in the global case. The main difference is associated to the update which becomes:

$$\Delta_{loc} \quad = \quad \frac{1}{h_t(x_O)} - \frac{1}{h_{max}} \tag{4.9}$$

Where the second term is a correction term that makes sure that the CFI is no updated when the leaf node associated to the predicted outlier $\mathbf{x_O}$ is at maximum depth of the tree. Basically it prevents the updates of the CFI also in case the point under examination is not isolated.

4.3 Comparison: normalized counts and TF-IDF

A large part of the work done with these techniques of interpretability is linked to the comparison made between normalized counts and TF-IDF, as already exposed in Sec. 2.3.1 the use of one or the other technique has different implications at implementation level, obviously if the choice was done considering only at theoretical perspective then it would fall on TF-IDF. This is because it is the technique that is more robust, it allows to "weigh" every word on all available data and not only on each document, this peculiarity makes it excellent when many "stop-words" are present. The normalized counts, as already said, attempt to emulate this technique using lot less information but being much more efficient both in terms of memory and time. If a dynamic environment is added to the recipe than the characteristics of the last have to be taken into consideration, as the calculation of the term **IDF** in a dynamic framework with a limited memory amount could be really problematic.

The analysis carried out is about global interpretability of the two models trained on the two different representations, because to understand if the two are exchangeable it is necessary to understand if the models "think" in the same way even if trained on different representations. Local interpretability is also used to provide

further results to confirm or deny the global interpretability outcome. The techniques used are those exposed in the previous sections, it has opted for these techniques due to various factors. The use of AcME is certainly linked to its adaptability and computational efficiency, especially if it is put on the level of SHAP. The problem of AcME, if that can be said, is that its logic overlooks the small details, as it does not make any intake on the model but treats a black box playing with the features. Here DIFFI comes into play which, designed and developed to deal with the isolation forest, therefore a technique that is based on the model itself which makes it much less applicable in general but perhaps more precise in a highly specialized case like this one.

The graphs in Fig. 4.7 and Fig. 4.8 show the summary plots provided by AcME which refer to the global interpretability of the isolation forest models trained on the two different representations. The summary plot is composed as follow, on the two axes x and y are represented respectively the values of the standardization effect for the Q perturbations based on quantiles and the corresponding features in order of importance. The color represents the quantile level of the feature, from low (marked in blue) to high (marked in red). What is derived from their observation is interesting as it seems that the first 15 features used by the two algorithms are exactly the same, the order changes a little but the concept is that the models rely on the same things to provide a result.

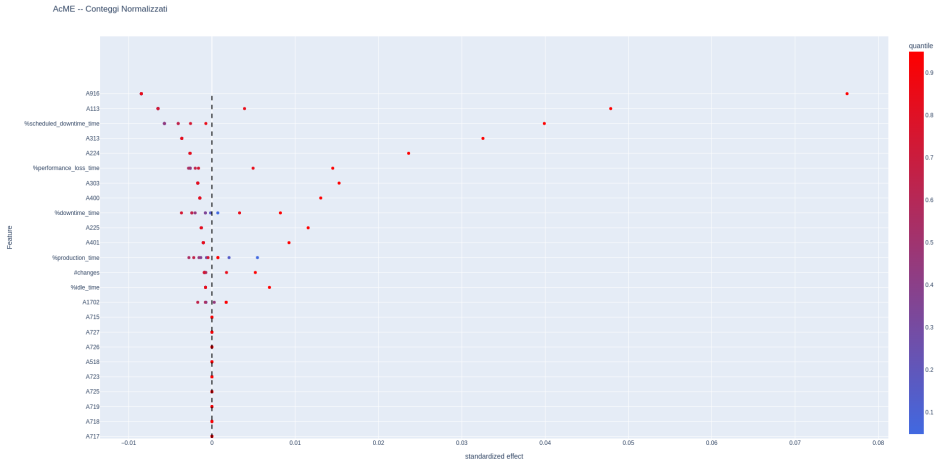


Figure 4.7: AcME summary plot: normalized counts

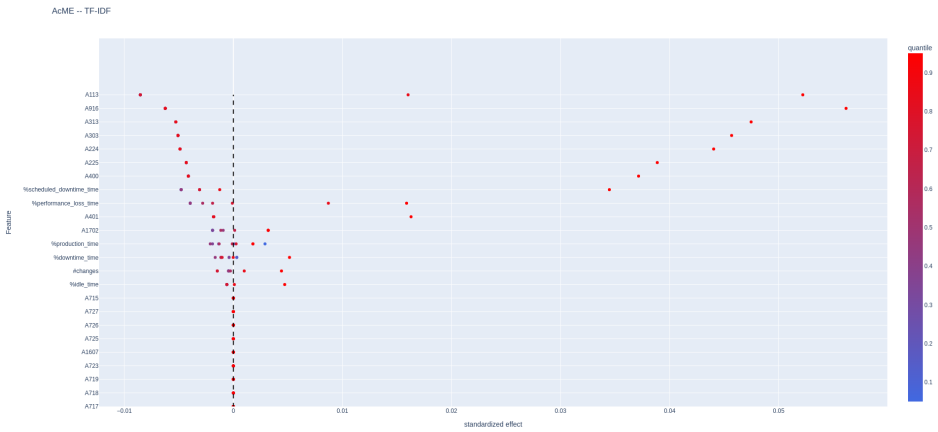


Figure 4.8: AcME summary plot: Tf-IDF

In Fig. 4.9 a comparison can be made in terms of global interpretability of the two models as before but this time the results come from the use of DIFFI. Also in this case the outcomes are similar to the previous ones but not so much as although some features are the same, the two graphs do not exactly coincide as before with AcME, even if as mentioned above there are some similarities.

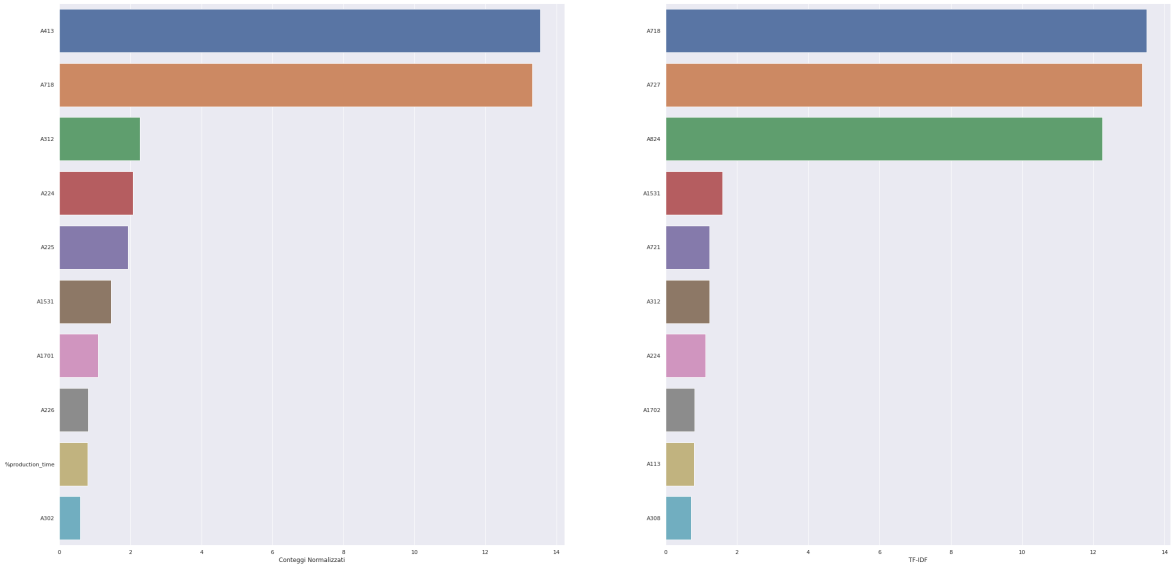
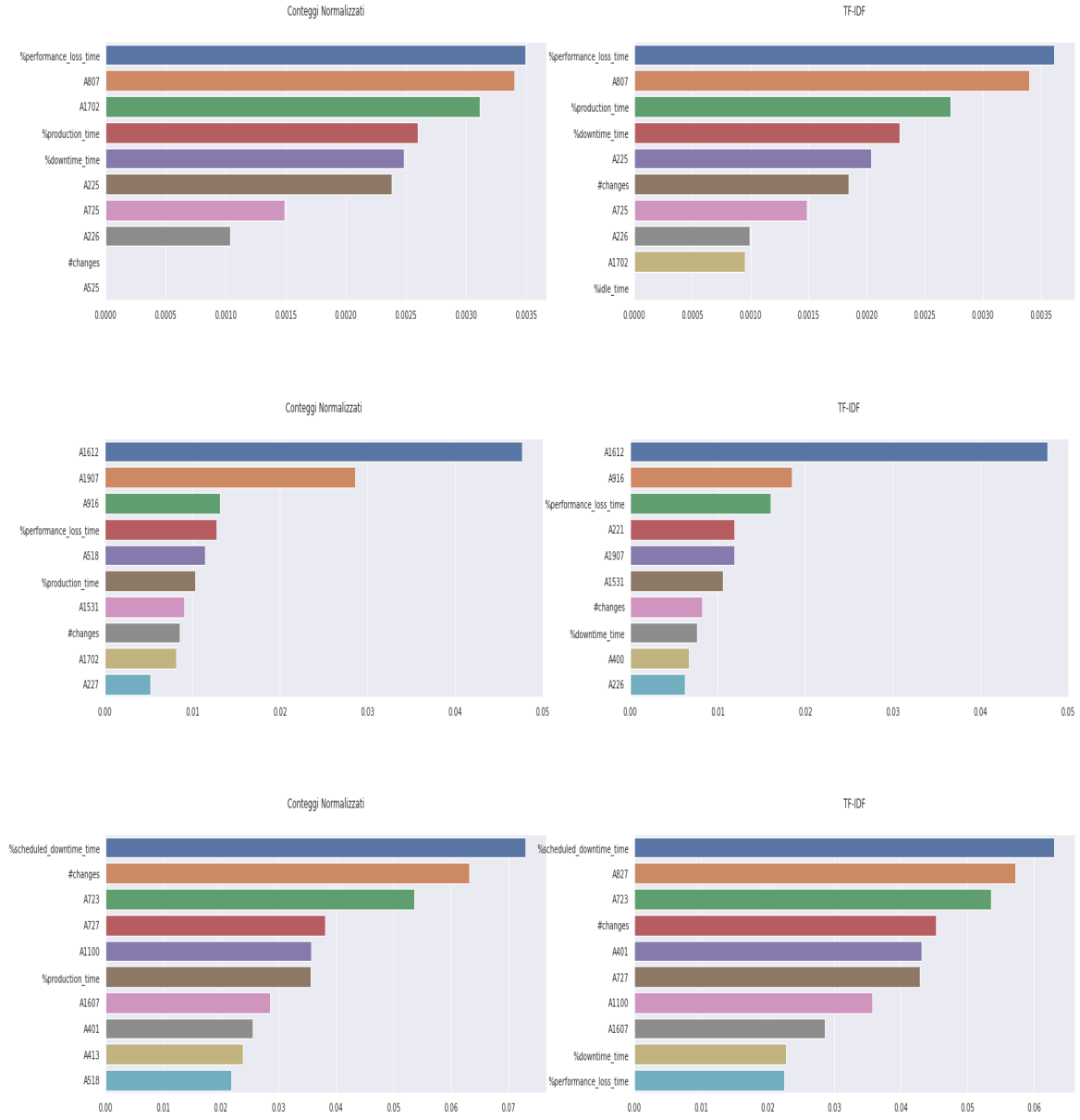
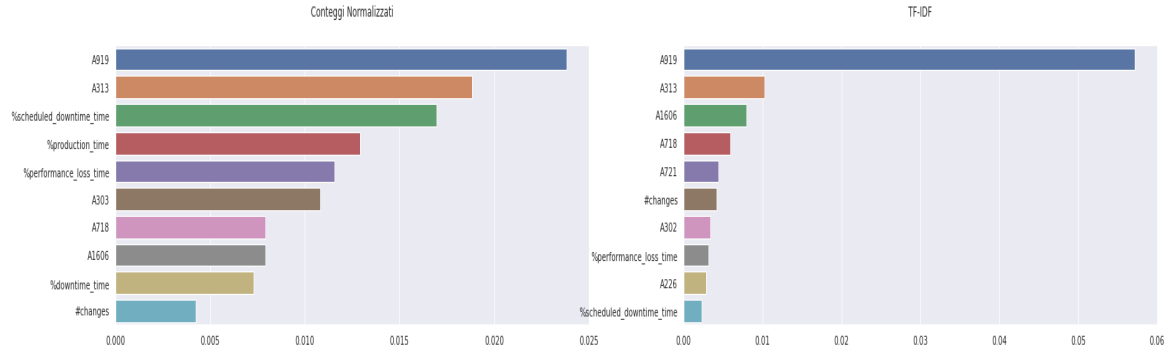


Figure 4.9: DIFFI global importance comparison

To confirm what has been said above, some further analysis are performed as global interpretability brings similar results but not good as using AcME as previously said. In particular, DIFFI is used to perform a local interpretability procedure in which a bunch of points is taken and examined.

The following images contain bar plots that schematize the importance values associated with the features in descending order. On the left we find the results associated with the normalized counts while on the right those for TF-IDF, as can be seen at first sight, although the importance values are different, the features used in the prediction of the single point are exactly the same in both cases, thus confirming the not entirely convincing results obtained previously. Obviously, to say this many points have been analyzed, and it should also be pointed out that treating everything only at the local level would not have led to a complete and coherent analysis. The structure of the points themselves was removed from the images as it was not informative.





At this point, before drawing final considerations, it is interesting to make a comparison between the two interpretability techniques used to try to best extrapolate the information coming from one or the other.

First of all it is necessary to point out that there is no a better technique, but each of them has its strengths and weaknesses. The first method, AcME, perform a sensitivity analysis and certainly allows to detect all the features that are relevant in terms of variation of the result, however the model-agnostic nature of AcME can lead to skip some details that could be relevant. On the other hand DIFFI and its model-specific nature permits to have a different point of view. DIFFI seeks to reflect the actual logic that governs the behavior of the IF, the features that this algorithm classifies as more relevant are those that tend to "unbalance" the tree in a very emphasized way. In conclusion, after highlight that the results provided by the two tecniques cannot be compared because the idea behind them is too different, the fact that the results tend in the same direction is very interesting as it means that despite trying to solve the problem from two different angles the result is the same and therefore this adds a certain degree of robustness to the solution. Therefore the result of this analysis leads to say that the two representations, respectively normalized counts and TF-IDF are interchangeable, consequently the fact that the first of the two is more efficient for the reasons already explained, leads to drop the choice on it.

Chapter 5

Conclusion

5.1 Proposed interpretable anomaly detection system

In order to conclude, in this final section a brief summary about the thesis is proposed and than we will move to the "Proposed interpretable anomaly detection system". The thesis chapters are ordered following the development process of the whole system, from the raw data handling, passing through the creation of the final representation and the application of anomaly detection techniques and interpretability methods.

The main goal was the development of an interpretable anomaly detection system built in order to improve the efficiency of the filling machines produced by Galdi since market standards are growing more and more requiring more and more performing and efficient machines. In the end, the system make use of the Representation 5 the one that is composed by the parts regarding alarms, time window state composition, time window state transitions since this is the representation that better describes the filling machine providing a complete description of the latter.

The anomaly detection algorithm employed is the isolation forest, algorithm that fits perfectly mainly because it works well in unsupervised situations, it is efficient and also very robust. Interpretability has been used on two different levels, at the design level it is used as a tool to evaluate the algorithms and identify the most suitable representation, at the user level instead it performs the function of helping the user to understand well the choices of the algorithm and therefore to make it more transparent. In addition, the validation process was obviously done by referring to domain experts and technicians.

The graphics have been designed together with the Galdi technicians so that they are understandable and useful. An example can be observed in Fig. 5.1, remember that the latter are composed of four parts and knowing the structure facilitates the reading of the graph. The part highlighted in blue contains information about the serial number of the machine, the time window in which the anomaly was detected and also the percentage of what is considered anomalous. The other parts highlighted in yellow, red and purple respectively describe the three components of the final representation. The description of the alarms in the image has been obscured for confidentiality reasons.

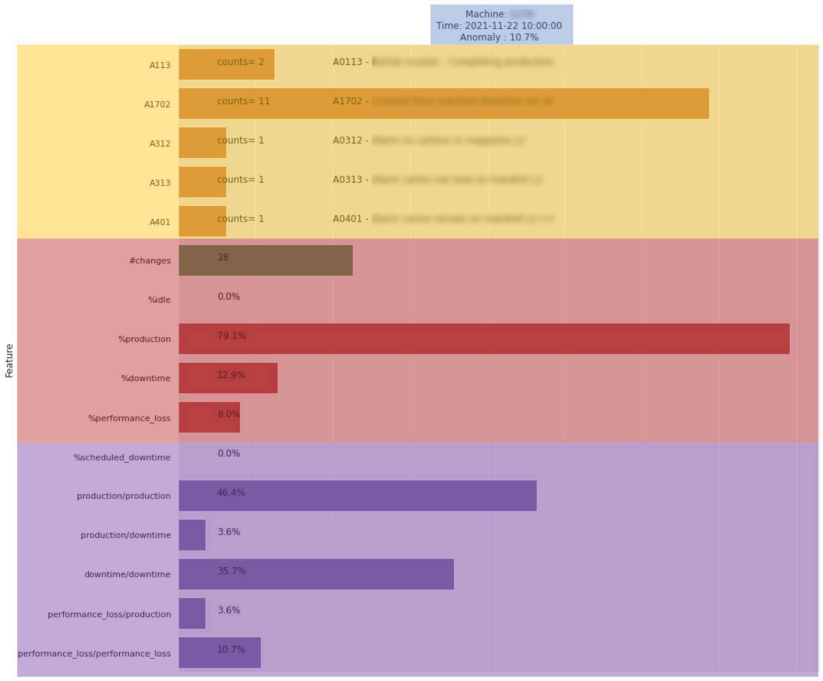


Figure 5.1: Visualization example

The "Proposed interpretable anomaly detection system" in its first version will provide automatic email in case of anomaly, including the graph and a link to the MaSH, something similar to the one in Fig. 5.2, so that the user has this tool available to perform Root Cause Analysis and step in, if required, with suitable actions for the problem encountered.

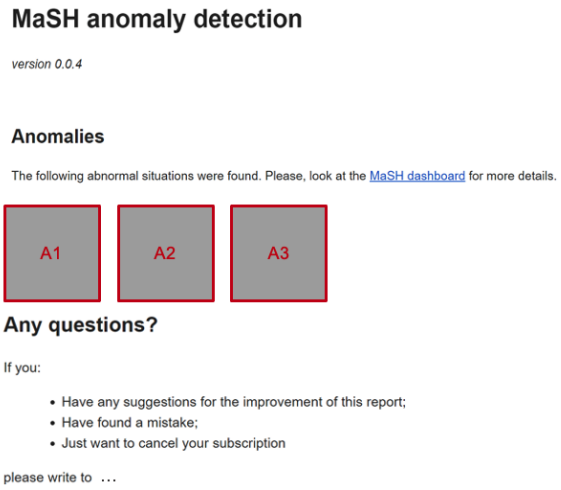


Figure 5.2: Automatic mail example

At the moment the prototyping phase is almost completed and the system will soon be moved to production.

5.2 Future research direction

Machine learning play a key role in enabling Industry 4.0 and IoT scenarios. The thesis shows the strength of this technology which will take root more and more in the next few years. It has been demonstrated that even in a real case subject to real problems a data-driven approach can be used. The effectiveness of the anomaly detection algorithms has already been fully demonstrated in real scenarios, however they still lack a certain transparency which, as already mentioned, is leading toward what is defined as interpretable machine learning. In the case in question, an attempt was made to narrow this gap, to obtain results that are not only an end in themselves but can fulfil the actual needs of the end users.

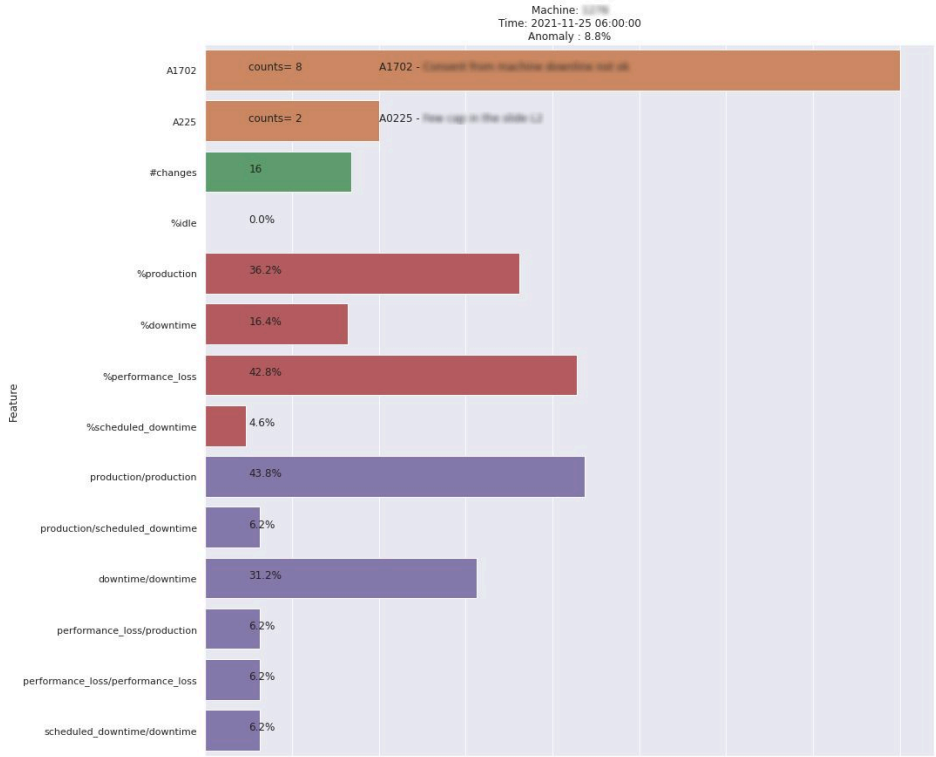
The type of approach chosen is a possible solution, many others are possible as it is possible to introduce improvements that impact on the data, on the representation to try to obtain something more focused on the customer in order to better solve his needs, such as it is also possible to change the algorithms, introduce themes of continual learning or methodologies that allow to improve the customer's performance by observing similar situations and solutions adopted.

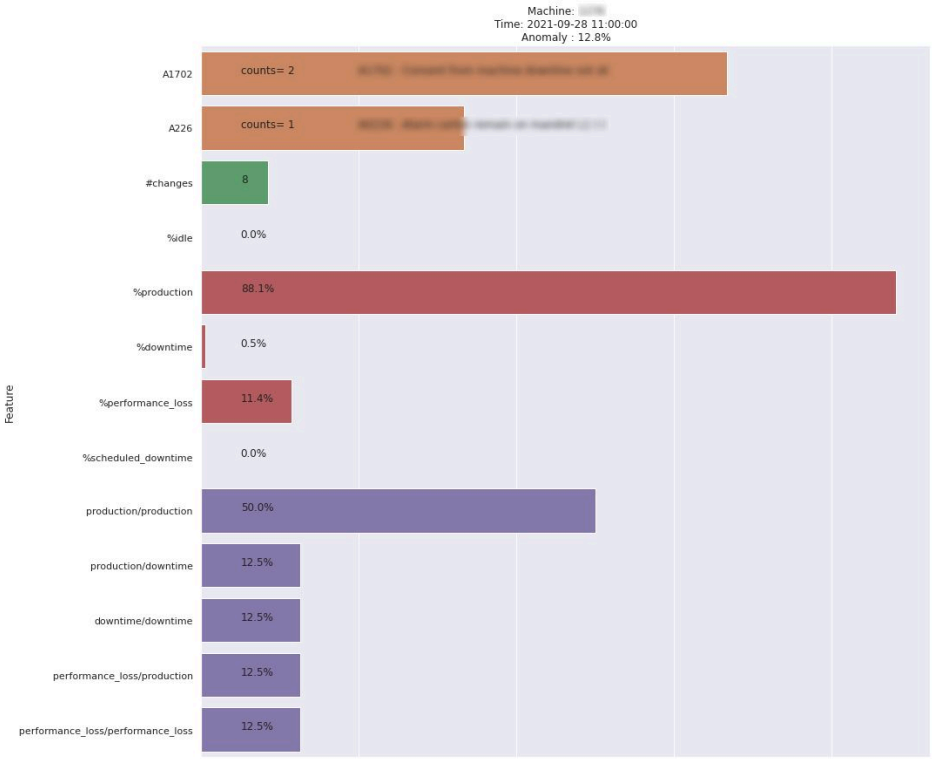
Appendix A

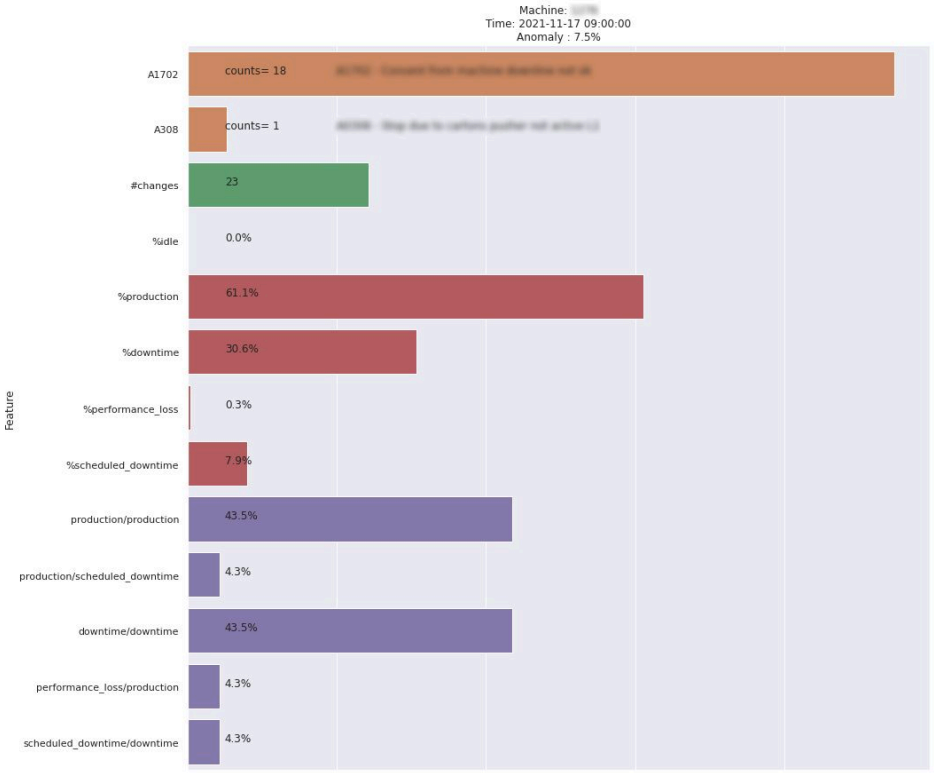
Appendix

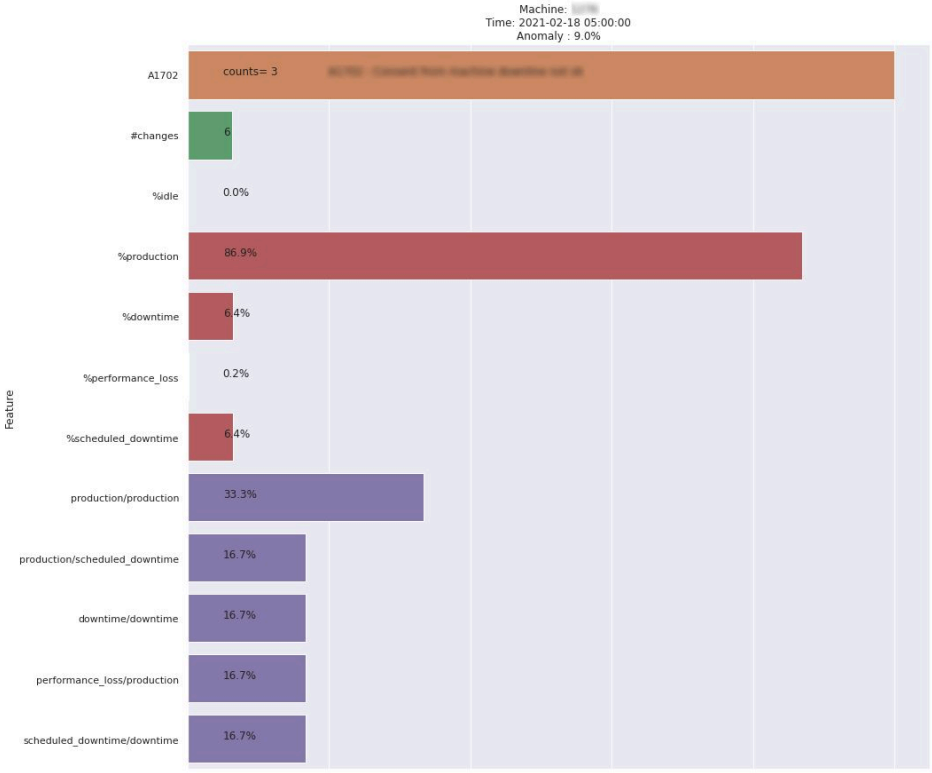
A.1 Anomaly detection results

A.1.1 Normal points

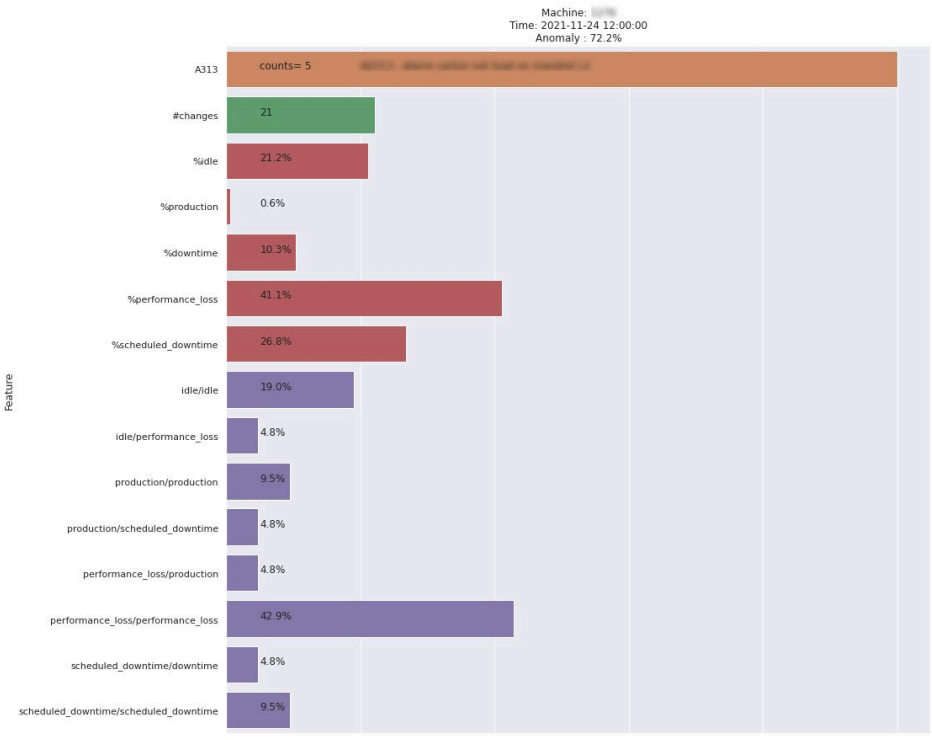


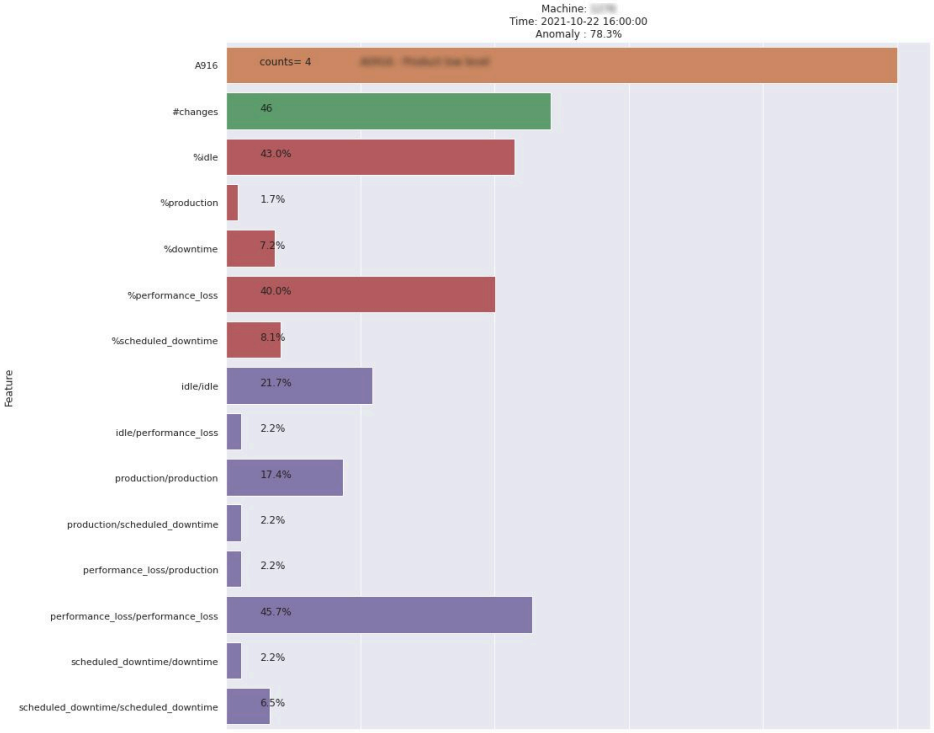

Machine:
Time: 2021-09-28 11:00:00
Anomaly : 12.8%
Feature
A1702
counts= 2
A226
counts= 1
#changes
8
%idle
0.0%
%production
88.1%
%downtime
0.5%
%performance_loss
11.4%
%scheduled_downtime
0.0%
production/production
50.0%
production/downtime
12.5%
downtime/downtime
12.5%
performance_loss/production
12.5%
performance_loss/performance_loss
12.5%

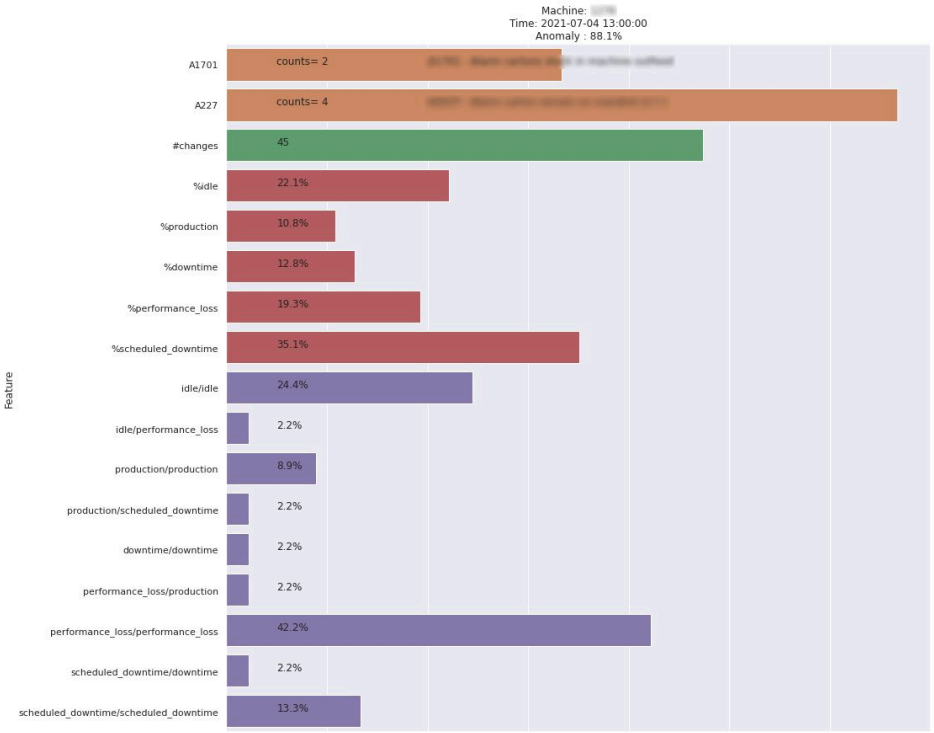

Machine:
Time: 2021-11-17 09:00:00
Anomaly : 7.5%
Feature
A1702
counts= 18
A308
counts= 1
#changes
23
%idle
0.0%
%production
61.1%
%downtime
30.6%
%performance_loss
0.3%
%scheduled_downtime
7.9%
production/production
43.5%
production/scheduled_downtime
4.3%
downtime/downtime
43.5%
performance_loss/production
4.3%
scheduled_downtime/downtime
4.3%

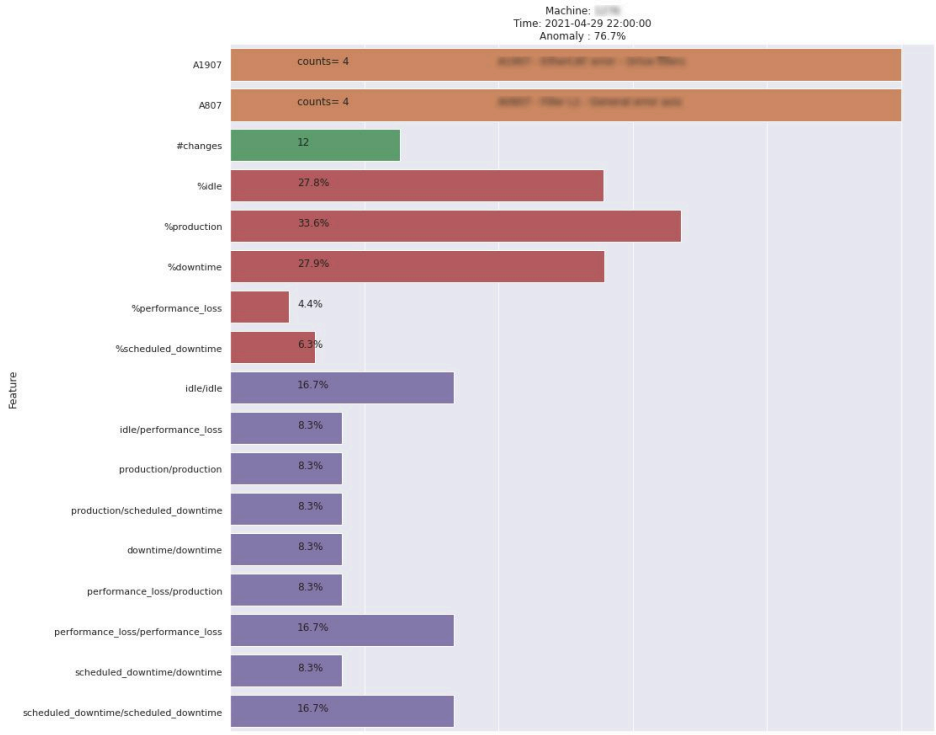


A.1.2 Anomaly points



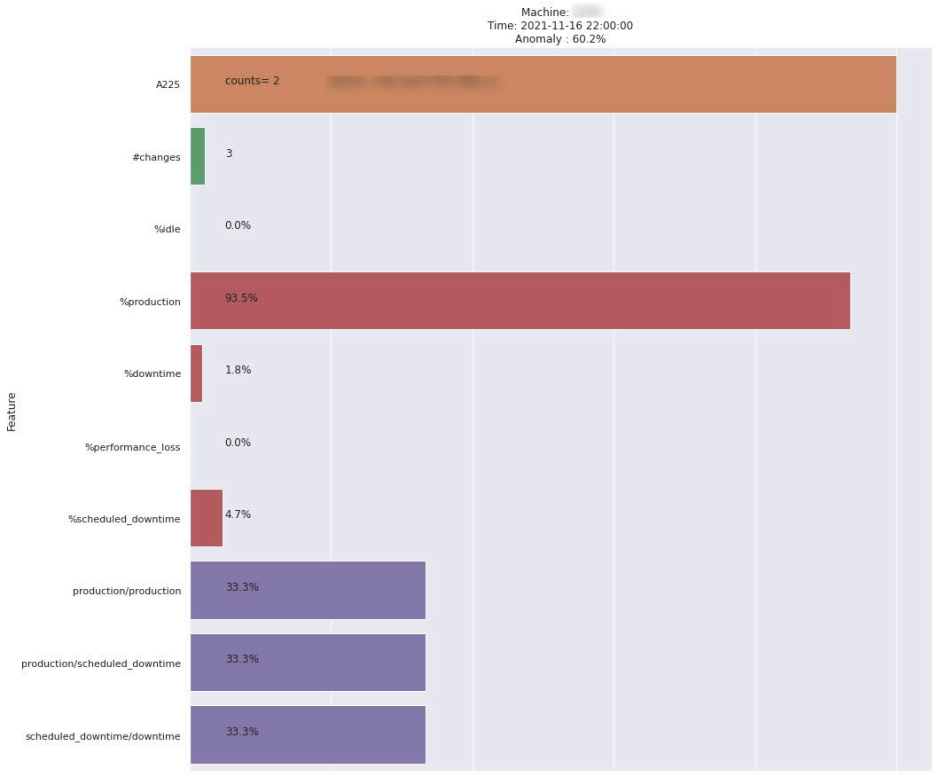

Machine:
Time: 2021-10-22 16:00:00
Anomaly : 78.3%
Feature
A916
counts= 4
#changes
46
%idle
43.0%
%production
1.7%
%downtime
7.2%
%performance_loss
40.0%
%scheduled_downtime
8.1%
idle/idle
21.7%
idle/performance_loss
2.2%
production/production
17.4%
production/scheduled_downtime
2.2%
performance_loss/production
2.2%
performance_loss/performance_loss
45.7%
scheduled_downtime/downtime
2.2%
scheduled_downtime/scheduled_downtime
6.5%

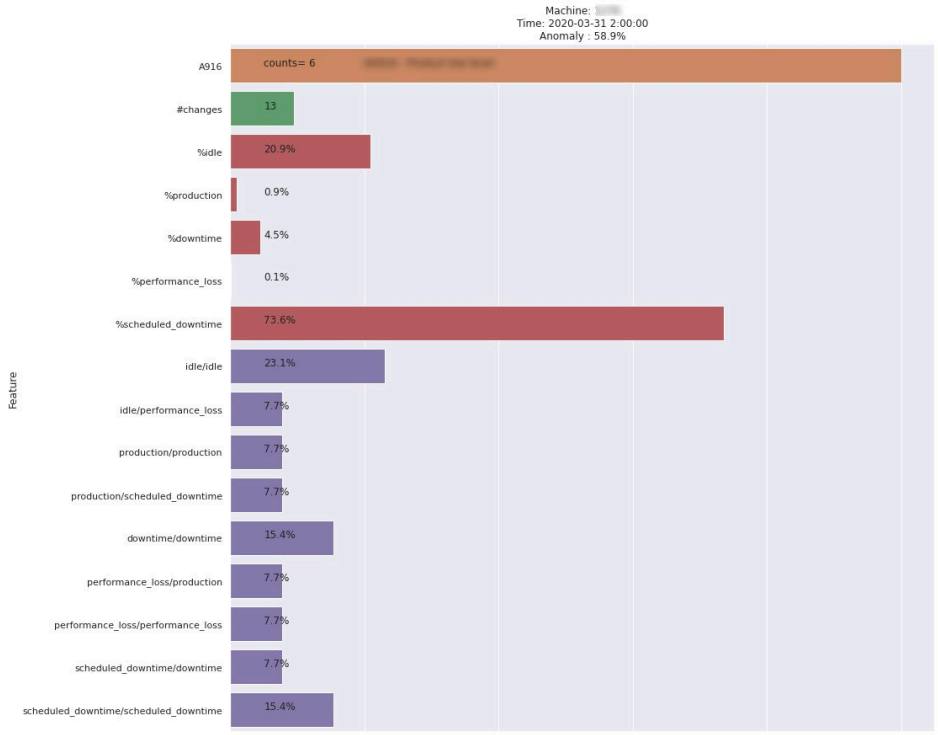

Machine:
Time: 2021-07-04 13:00:00
Anomaly : 88.1%
Feature
A1701
counts= 2
A227
counts= 4
#changes
45
%idle
22.1%
%production
10.8%
%downtime
12.8%
%performance_loss
19.3%
%scheduled_downtime
35.1%
idle/idle
24.4%
idle/performance_loss
2.2%
production/production
8.9%
production/scheduled_downtime
2.2%
downtime/downtime
2.2%
performance_loss/production
2.2%
performance_loss/performance_loss
42.2%
scheduled_downtime/downtime
2.2%
scheduled_downtime/scheduled_downtime
13.3%

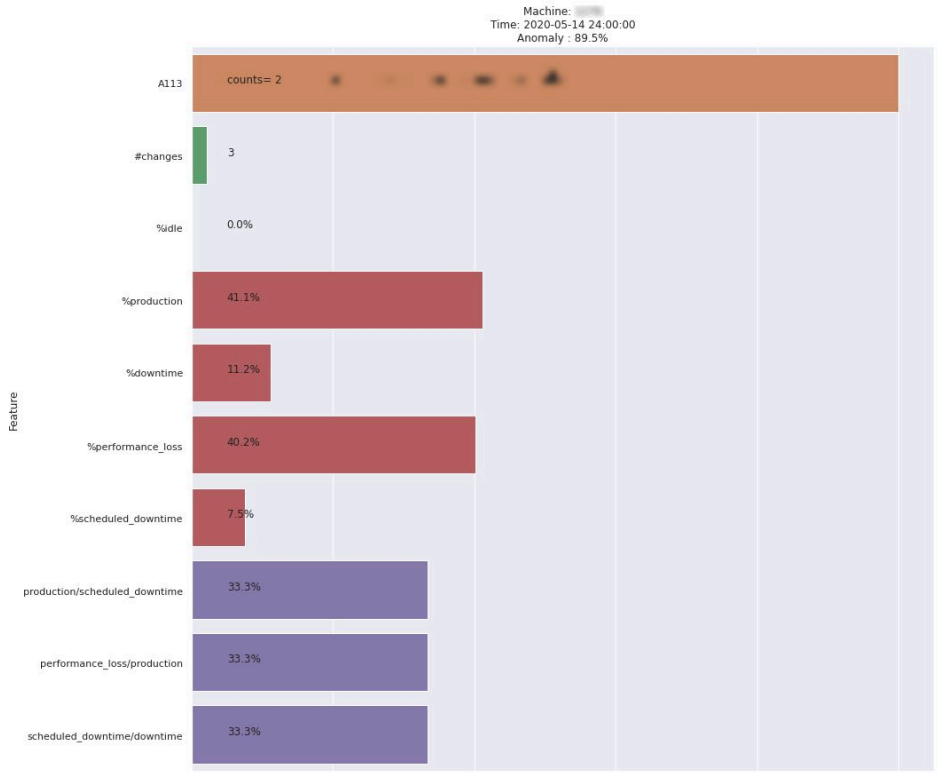

Machine:
Time: 2021-04-29 22:00:00
Anomaly : 76.7%
Feature
A1907
counts= 4
A807
counts= 4
#changes
12
%idle
27.8%
%production
33.6%
%downtime
27.9%
%performance_loss
4.4%
%scheduled_downtime
6.3%
idle/idle
16.7%
idle/performance_loss
8.3%
production/production
8.3%
production/scheduled_downtime
8.3%
downtime/downtime
8.3%
performance_loss/production
8.3%
performance_loss/performance_loss
16.7%
scheduled_downtime/downtime
8.3%
scheduled_downtime/scheduled_downtime
16.7%

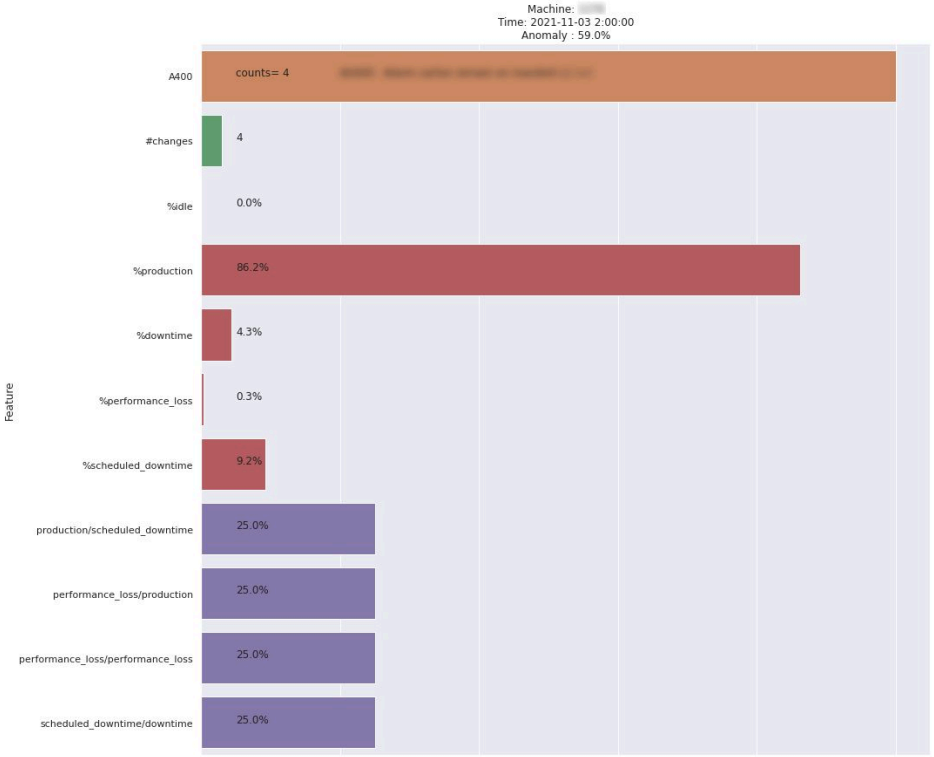
A.2 HDBSCAN interpretability results

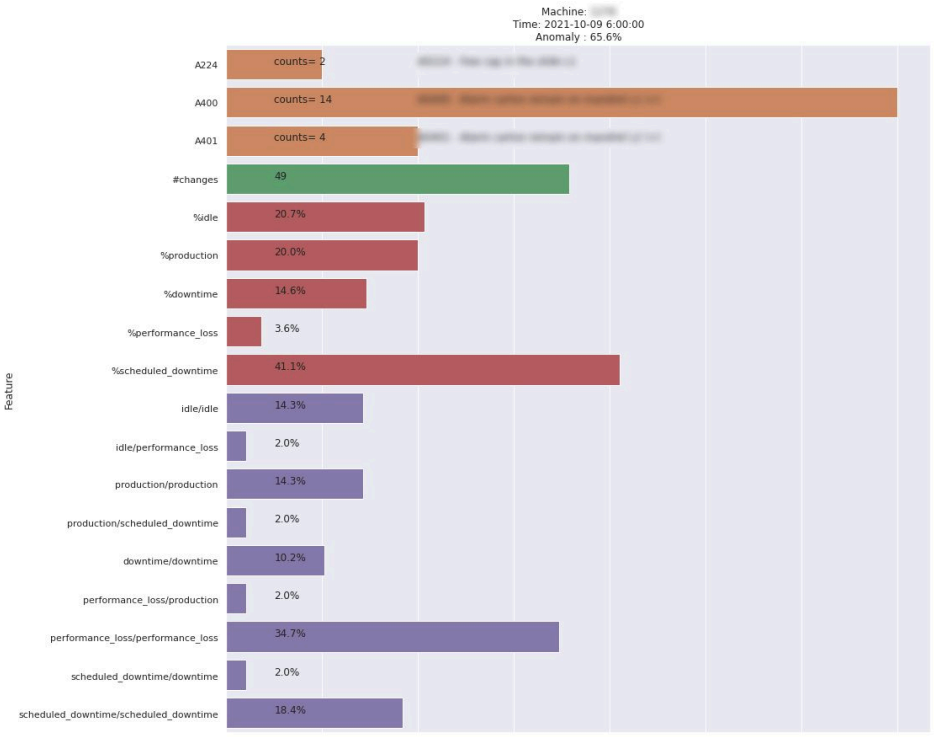
A.2.1 Medoids

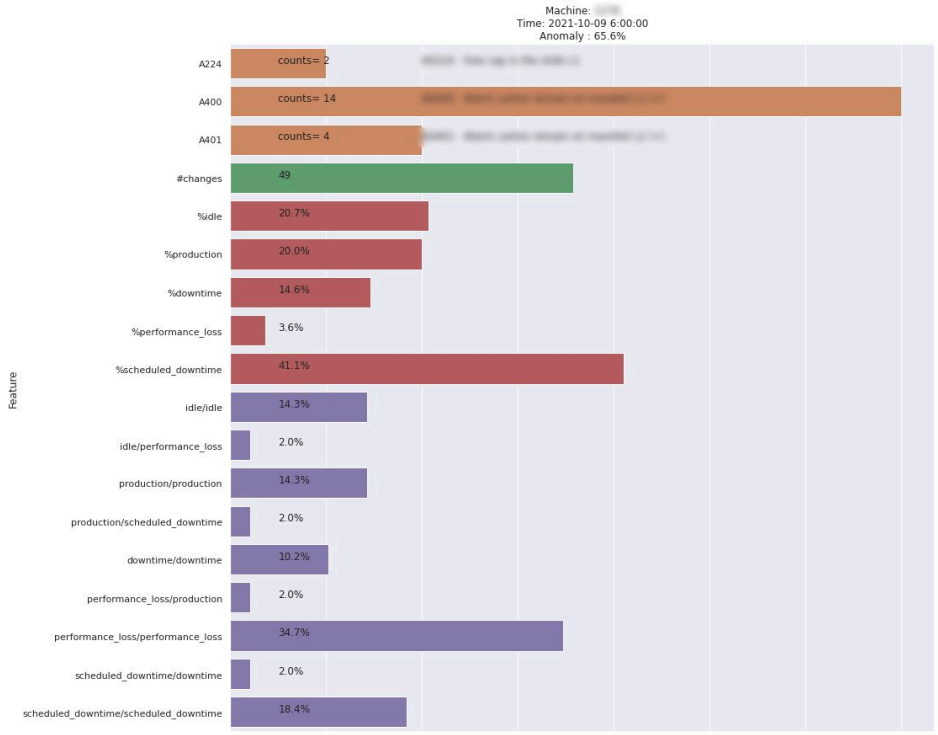


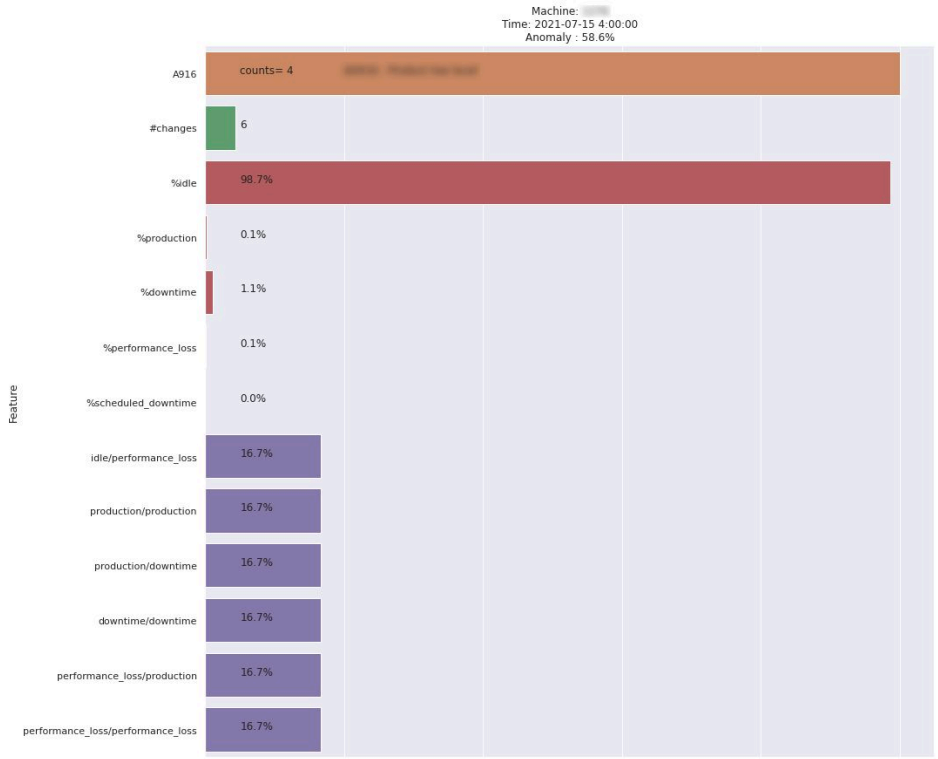

Machine:
Time: 2020-03-31 2:00:00
Anomaly : 58.9%
Feature
A916
counts= 6
#changes
13
%idle
20.9%
%production
0.9%
%downtime
4.5%
%performance_loss
0.1%
%scheduled_downtime
73.6%
idle/idle
23.1%
idle/performance_loss
7.7%
production/production
7.7%
production/scheduled_downtime
7.7%
downtime/downtime
15.4%
performance_loss/production
7.7%
performance_loss/performance_loss
7.7%
scheduled_downtime/downtime
7.7%
scheduled_downtime/scheduled_downtime
15.4%


Machine:
Time: 2020-05-14 24:00:00
Anomaly : 89.5%
Feature
A113
counts= 2
#changes
3
%idle
0.0%
%production
41.1%
%downtime
11.2%
%performance_loss
40.2%
%scheduled_downtime
7.5%
production/scheduled_downtime
33.3%
performance_loss/production
33.3%
scheduled_downtime/downtime
33.3%

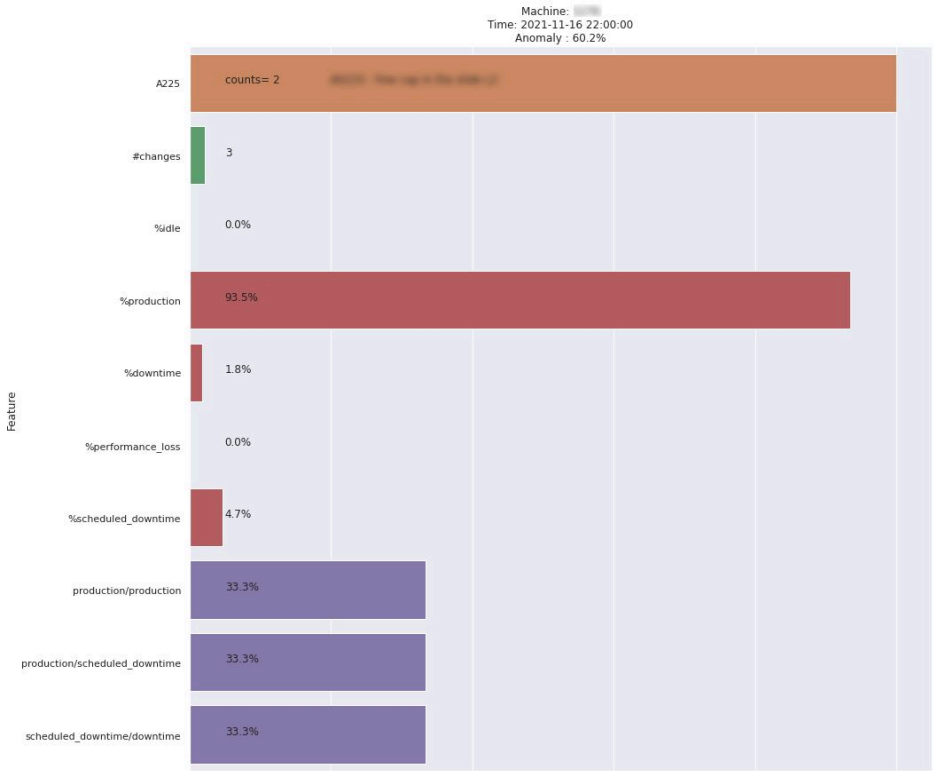

Machine:
Time: 2021-11-03 2:00:00
Anomaly : 59.0%
Feature
A400
counts= 4
#changes
4
%idle
0.0%
%production
86.2%
%downtime
4.3%
%performance_loss
0.3%
%scheduled_downtime
9.2%
production/scheduled_downtime
25.0%
performance_loss/production
25.0%
performance_loss/performance_loss
25.0%
scheduled_downtime/downtime
25.0%


Machine:
Time: 2021-10-09 6:00:00
Anomaly : 65.6%
Feature
A224
counts= 2
A400
counts= 14
A401
counts= 4
#changes
49
%idle
20.7%
%production
20.0%
%downtime
14.6%
%performance_loss
3.6%
%scheduled_downtime
41.1%
idle/idle
14.3%
idle/performance_loss
2.0%
production/production
14.3%
production/scheduled_downtime
2.0%
downtime/downtime
10.2%
performance_loss/production
2.0%
performance_loss/performance_loss
34.7%
scheduled_downtime/downtime
2.0%
scheduled_downtime/scheduled_downtime
18.4%


Machine:
Time: 2021-10-09 6:00:00
Anomaly : 65.6%
Feature
A224
counts= 2
A400
counts= 14
A401
counts= 4
#changes
49
%idle
20.7%
%production
20.0%
%downtime
14.6%
%performance_loss
3.6%
%scheduled_downtime
41.1%
idle/idle
14.3%
idle/performance_loss
2.0%
production/production
14.3%
production/scheduled_downtime
2.0%
downtime/downtime
10.2%
performance_loss/production
2.0%
performance_loss/performance_loss
34.7%
scheduled_downtime/downtime
2.0%
scheduled_downtime/scheduled_downtime
18.4%


Machine:
Time: 2021-07-15 4:00:00
Anomaly : 58.6%
Feature
A916
counts= 4
#changes
6
%idle
98.7%
%production
0.1%
%downtime
1.1%
%performance_loss
0.1%
%scheduled_downtime
0.0%
idle/performance_loss
16.7%
production/production
16.7%
production/downtime
16.7%
downtime/downtime
16.7%
performance_loss/production
16.7%
performance_loss/performance_loss
16.7%


Machine:
Time: 2020-01-27 22:00:00
Anomaly : 64.5%
A224
counts= 2
#changes
3
%idle
0.0%
%production
75.7%
%downtime
7.4%
%performance_loss
0.0%
%scheduled_downtime
16.9%
production/production
33.3%
production/scheduled_downtime
33.3%
scheduled_downtime/downtime
33.3%
Feature


Machine:
Time: 2021-11-16 22:00:00
Anomaly : 60.2%
A225
counts= 2
#changes
3
%idle
0.0%
%production
93.5%
%downtime
1.8%
%performance_loss
0.0%
%scheduled_downtime
4.7%
production/production
33.3%
production/scheduled_downtime
33.3%
scheduled_downtime/downtime
33.3%
Feature

A.2.2 Curse of dimensionality handling

This paragraph explains a topic that is not central to the thesis but that has been tested to try to solve the problem that afflicts HDBSCAN or in general any machine learning algorithm that has to deal with high-dimensional data which are also very scattered, the idea is introduced in Sec. 3.2.

The basic idea is to help the learning algorithm by subjecting the data to a feature selection procedure, then passing through a dimensionality reduction technique before applying the chosen method which goes from a clustering algorithm to improve its preforms, to an anomaly detection algorithm as it can work with clearer data.

Some examples of this are reported in (Allaoui, Kherfi, and Cheriet, 2020) and (Xie, Girshick, and Farhadi, 2016), in particular the technique exploited in the first paper make use of UMAP as dimensionality reduction before applying a clustering algorithm, the choice of UMAP comes from the fact that as said the algorithm maintain both local and global structure providing a compressed representation of that raw data that is quite reliable. Some of the results provided by the article can be seen in Fig. A.1, obviously the results are excellent and for this reason it is important to say a few words about this, even in the case in question.

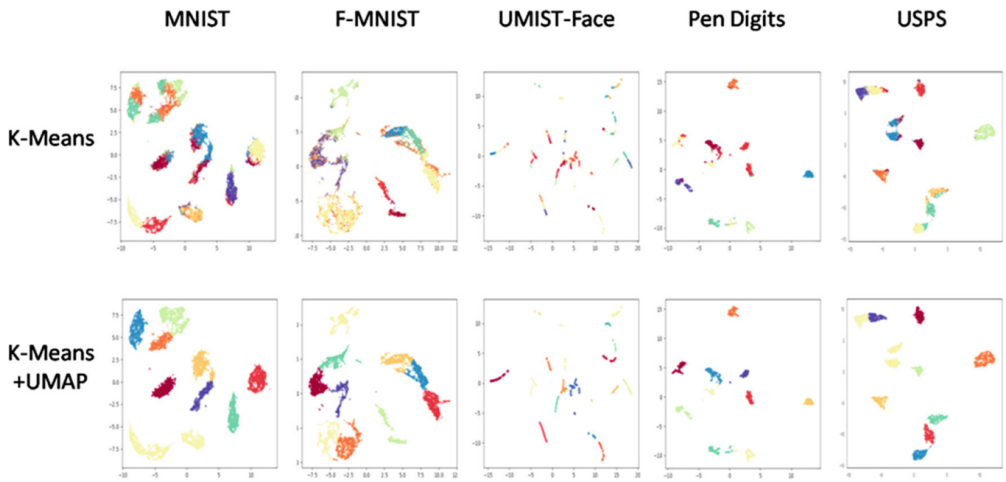


Figure A.1: Results on well known datasets

Following the idea also in our case some nice but not completely reliable results were obtained, although the technique used even at an intuitive level seems able to

function it is very primitive, in a totally unsupervised case, using such a method without any precautions or information about any label could be very misleading.

The manipulation of the output of a dimensional reduction technique must be done very carefully, otherwise any interpretation can be very misleading or wrong because the dimensional reduction help with handle the dimensionality problem but it will certainly result in the loss of functionality, perhaps noisy or true characteristics, but a priori, it is not feasible to provide a concrete answer to this.

In conclusion, the results can be interpreted if there is an a priori labelling or some domain information that can be exploited in some way, focus for example on the well known MNIST dataset, each datapoint have a label or if it did not have it, however, one would be able to distinguish by hand a good result and bad one.

Bibliography

Allaoui, Mebarka, Mohammed Lamine Kherfi, and Abdelhakim Cheriet (2020). "Considerably improving clustering algorithms using UMAP dimensionality reduction technique: a comparative study". In: *International Conference on Image and Signal Processing*. Springer, pp. 317–325.

Becht, Etienne et al. (2018). "Dimensionality reduction for visualizing single-cell data using UMAP". In: *Nature Biotechnology* 37, pp. 38–44.

Breunig, Markus M. et al. (2000). "LOF: Identifying Density-Based Local Outliers". In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD '00. Dallas, Texas, USA: Association for Computing Machinery, 93–104. ISBN: 1581132174. DOI: 10.1145/342009.335388. URL: https://doi.org/10.1145/342009.335388.

Carletti, Mattia, Matteo Terzi, and Gian Antonio Susto (2020). "Interpretable anomaly detection with diffi: Depth-based feature importance for the isolation forest". In: *arXiv preprint arXiv:2007.11117*.

Caruso, Giulia et al. (June 2018). "Cluster Analysis as a Decision-Making Tool: A Methodological Review". In: pp. 48–55. ISBN: 978-3-319-60881-5. DOI: 10.1007/978-3-319-60882-2_6.

Chandola, Varun, Arindam Banerjee, and Vipin Kumar (2009). "Anomaly detection: A survey". In: *ACM Comput. Surv.* 41, 15:1–15:58.

Cheng, Justin, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec (2021). "Antisocial Behavior in Online Discussion Communities". In: *Proceedings of the International AAAI Conference on Web and Social Media* 9.1, pp. 61–70. URL: https://ojs.aaai.org/index.php/ICWSM/article/view/14583.

Chun-Hui, Xiao et al. (2018). "Anomaly Detection in Network Management System Based on Isolation Forest". In: *2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC)*, pp. 56–60.

Dandolo, David et al. (2021). *AcME – Accelerated Model-agnostic Explanations: Fast Whitening of the Machine-Learning Black Box*. DOI: 10.48550/ARXIV.2112.12635. URL: https://arxiv.org/abs/2112.12635.

Doshi-Velez, Finale and Been Kim (2017). *Towards A Rigorous Science of Interpretable Machine Learning*. DOI: 10.48550/ARXIV.1702.08608. URL: https://arxiv.org/abs/1702.08608.

Hall, Patrick and Navdeep Gill (2019). *An Introduction to Machine Learning Interpretability. An Applied Perspective on Fairness, Accountability, Transparency, and Explainable AI*. 2nd ed. URL: https://www.oreilly.com/library/view/an-introduction-to/9781492033158/.

Karev, Dimitar, Christopher B. McCubbin, and Ruslan Vaulin (2017). "Cyber Threat Hunting Through the Use of an Isolation Forest". In: *Proceedings of the 18th International Conference on Computer Systems and Technologies*.

Katiyar, S. K. and P. V. Arun (2014). *Comparative analysis of common edge detection techniques in context of object extraction*. DOI: 10.48550/ARXIV.1405.6132. URL: https://arxiv.org/abs/1405.6132.

Kriegel, Hans Peter, Matthias Schubert, and Arthur Zimek (Dec. 2008). "Angle-based outlier detection in high-dimensional data". English. In: *Proceedings of the 14th ACMKDD International Conference on Knowledge Discovery and Data Mining*. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ; Conference date: 24-08-2008 Through 27-08-2008. United States: Association for Computing Machinery, pp. 444–452. ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1401946.

Kumar, Srijan et al. (2018). "MIS2: Misinformation and Misbehavior Mining on the Web". In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. WSDM '18. Marina Del Rey, CA, USA: Association for Computing Machinery, 799–800. ISBN: 9781450355810. DOI: 10.1145/3159652.3160597. URL: https://doi.org/10.1145/3159652.3160597.

Liu, Fei Tony, Kai Ting, and Zhi-Hua Zhou (Jan. 2009). "Isolation Forest". In: pp. 413 –422. DOI: 10.1109/ICDM.2008.17.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86, pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

Manning, Christopher D. and Hinrich Schütze (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press. ISBN: 0262133601.

McInnes, Leland, John Healy, and Steve Astels (2017). "hdbscan: Hierarchical density based clustering". In: *The Journal of Open Source Software* 2.11, p. 205.

McInnes, Leland, John Healy, and James Melville (2018). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. DOI: 10.48550/ARXIV.1802.03426. URL: https://arxiv.org/abs/1802.03426.

Molnar, Christoph (2022). *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. URL: https://christophm.github.io/interpretable-ml-book.

Qader, Wisam, Musa M. Ameen, and Bilal Ahmed (June 2019). "An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges". In: pp. 200–204. DOI: 10.1109/IEC47844.2019.8950616.

Ranga Suri, N. N. R., Narasimha Murty M, and G. Athithan (2019). "Outlier Detection". In: *Outlier Detection: Techniques and Applications: A Data Mining Perspective*. Cham: Springer International Publishing, pp. 13–27. ISBN: 978-3-030-05127-3. DOI: 10.1007/978-3-030-05127-3_2. URL: https://doi.org/10.1007/978-3-030-05127-3_2.

Shalev-Shwartz, Shai and S. Ben-David (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press. ISBN: 9781107057135. URL: http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/index.html, /bib/shalev-shwartz/shalev2014understanding/Understanding_Machine_Learning.pdf.

Togbe, Maurras Ulbricht et al. (2020). "Anomaly Detection for Data Streams Based on Isolation Forest Using Scikit-Multiflow". In: *Computational Science and Its Applications – ICCSA 2020* 12252, pp. 15 –30.

Wei, Qi et al. (2018). "Anomaly detection for medical images based on a one-class classification". In: *Medical Imaging 2018: Computer-Aided Diagnosis*. Ed. by Nicholas Petrick and Kensaku Mori. Vol. 10575. International Society for Optics and Photonics. SPIE, p. 105751M. DOI: 10.1117/12.2293408. URL: https://doi.org/10.1117/12.2293408.

Xie, Junyuan, Ross Girshick, and Ali Farhadi (2016). "Unsupervised Deep Embedding for Clustering Analysis". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 478–487.

Zoppi, Tommaso, Andrea Ceccarelli, and Andrea Bondavalli (2018). "On algorithms selection for unsupervised anomaly detection". In: *Pacific Rimm Dependable computing Conference (PRDC18)*. IEEE. ISBN: 978-153865700-3. DOI: 10.1109/PRDC.2018.00050.