POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Anomaly detection as-a-Service for Predictive Maintenance

Tesi di Laurea Magistrale in
Computer Science And Engineering - Ingegneria Informatica

Daniele De Dominicis, 10502843

Abstract: Machine Learning as-a-Service (MLaaS) has gained large popularity over the last years and it refers to the wide range of Machine Learning (ML) tools offered as services from cloud computing providers. Basically, it is the use of technology to automatically train, test, and deploy Machine Learning models for a user, typically by leveraging an AI platform with offsite server farms (a "cloud") to run them on behalf of the customer. The goal of MLaaS is to make it easier and more affordable for companies to use ML, so they can get better insights from their data faster than ever before. MLaaS is used for a wide range of use-cases like Natural Language Processing, Forecasting, Regression, Image recognition etc. However, in the literature of Machine Learning as-a-Service very little attention has been posed to the data collected by sensors in industrial environments. From this perspective, the design and development of a Machine Learning as-a-Service tool for the identification of anomalies in the industrial environment is a novel and promising research area. This thesis aims at build an innovative client-server architecture that allows to offer Anomaly detection as-a-Service via RESTful APIs, based on supervised and unsupervised Machine Learning algorithms, with three different types of services: the training of a ML model, the update of a ML model and the inference of a sample data through a ML model. In this perspective, a real case study is treated in which Eisenmann Italia S.r.l. company wants to build a monitoring system for the identification of anomalies in the context of predictive maintenance. Predictive maintenance allows to anticipate the occurrence of certain events on the operation of industrial machinery, improving productivity, extending the life cycle of assets and reducing repair costs and complexity. In the industrial scenario of Eisenmann Italia S.r.l., a series of triaxial sensors mounted on applications for air treatment, burners etc., carry out a periodic data collection. Subsequently, these data were manipulated and used to train supervised and unsupervised ML models, allowing to classify and predict anomalous states of different machines. Moreover, since no data were available in which the machines are in anomalous conditions, synthetic data have been introduced, constructed from nominal data. The results obtained by both approaches are remarkable. In particular, those obtained from the unsupervised approach, based on the One-class Support Vector Machine (One-class SVM) algorithm give hope for a practical use of the entire client-server architecture for the Anomaly detection as-a-Service in the industrial processes.

Advisor:
Prof. Manuel Roveri

Co-advisors:
Dott. Alessandro Falcetta
Ing. Mirco Zanetti

Academic year:
2021-2022

Key-words: Machine Learning as-a-Service, Machine Learning, Anomaly detection as-a-Service, Anomaly detection, Client-Server Architecture, Unsupervised Model

1. Introduction

Cloud Computing has arguably been one of the biggest technology innovation of the past two decades. Cloud Computing is a model for enabling ubiquitous, convenient, on demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. A Cloud provider is an organization responsible for acquiring and managing a computing infrastructure and for making a service available to the parties concerned.
In particular, in the recent years Machine Learning as-a-Service (MLaaS) is playing a fundamental role in the Cloud world. The support of Machine Learning as-a-Service approach resides in the ability to provide users and companies, via web-console or API, with ready-to-use or remotely trainable Machine Learning models leveraging the high performance computing and memory abilities of Cloud Computing systems.
Likewise, nowadays in the production process of every company, machinery and automation play a fundamental role. For this reason, the planning of maintenance interventions on machines, the prevention of faults and breakdowns, based on the technology used each time and the needs of the market, has become increasingly important.
In the recent years the focus has thus moved from "reactive" maintenance interventions to "preventive" or "proactive" interventions, which do not repair the fault but prevent it, including the so-called predictive maintenance; it aims to prevent a certain machine or component from suddenly reaching the end of its useful life at the expense of the detriment of productivity and safety. Moreover it is an activity that is based on the analysis of data collected by machinery, in order to convert this information into a value that can contribute not only to improving their productivity, but also to reduce risks connected to a potential malfunction and minimize the impact of maintenance costs.
As anticipated, predictive maintenance falls into the category of scheduled maintenance interventions, which includes:

- **Preventive maintenance**: it intervenes just before the failure occurs.
- **Cyclical preventive maintenance**: it adds to the preventive nature of the technical interventions the planning over time, based on the wear to which the machinery is subject and the indications provided by the manufacturers.
- **Predictive maintenance**: with the help of additional components, such as valves, sensors, or graphic interfaces, it allows the company to plan interventions based on the exceeding of a certain "alert" value, the residual life time, the actual wear of a certain component. This allows to maximize the interventions, which are planned according to real needs and not only on the basis of estimates.

In particular, sensors are among the most suitable preventive maintenance solutions. High-quality static sensors are installed on the machinery that monitor the system in real time, collect and transmit important information, and report any anomalies.
Detecting anomalies has been a research topic for a long time and it has broad field of applications including fraud detection in financial transactions, intrusion detection in a computer network, monitoring sensor readings in an aircraft, spotting potential risk or medical problems in health data, and predictive maintenance.
Basically anomalies are data points which deviate from the normal distribution of the whole dataset, and Anomaly detection is the technique to find them. Input data can be broadly classified into sequential (e.g. voice, text, music, time-series) or non-sequential data (e.g. images, other data) and there are various types of approaches to identify anomalies such as statistical methods, classical Machine Learning methods, methods using neural networks (Deep Learning).
The main problem regarding Anomaly detection is the lack of anomaly data. Since the systems work properly for the most of their time, it is difficult to detect whether and when an anomaly event may happen. And therefore if this type of data does not exist, it is necessary to rely only on nominal data.

1.1. Goal and Results

This thesis aims at build an innovative client-server architecture that allows to offer Anomaly detection as-a-Service via RESTful APIs, based on supervised and unsupervised Machine Learning (ML) algorithms, with three different types of services: the training of a ML model, the update of a ML model and the inference of a sample data through a ML model.
In the literature of Machine Learning as-a-Service most of the efforts focused on images, video and textual information, while very little attention has been posed to sensor data sampling in industrial environments. From this perspective, the design and development of Cloud-based Machine Learning as-a-Service tool for the identification of anomalies in the industrial environment is a novel and promising research area.
Moreover, a real case study is treated in which Eisenmann Italia S.r.l. company wants to build a monitoring system for the identification of anomalies in the perspective of predictive maintenance in Industry 4.0. And a

client-server architecture for Anomaly detection as-a-Service is a solution tailored to the business's needs.
Specifically, the purpose of the Anomaly detection analysis is to identify anomalous operating states of such machinery from a predictive maintenance perspective. Where the anomalous state of a machine means a phenomenon that continues over time and that differs from previous measurements.
The technological scenario includes a series of triaxial accelerometers positioned on applications for air treatment such as rotary electric motors or burners. The malfunction in these types of devices inside the plant is critical as it would send down the production.
The lack of data for which the machines first work correctly and then enter an anomalous state, lead to the choice of one-class classification algorithms; at the same time another solution is to generate synthetic data based on nominal data making it abnormal and thus applying Machine Learning algorithms, both binary and multi-class classification. A key point is the use of one model for each sensor, since sensors are not homogeneous with each other.

1.2. Thesis Structure

Chapter 2 describes the current state of art of Machine Learning as-a-Service and Anomaly detection with a glance in the perspective of predictive maintenance.
Chapter 3 introduces the fundamental notions about Anomaly detection including the different types of anomalies and the properties of data, along with a basic background on main Statistical, Machine Learning and Deep Learning methods for Anomaly detection. Moreover, the vibration signal theory is treated.
Chapter 4 is the core of the work and proposes an innovative client-server architecture suitable for the Anomaly detection as-a-Service via RESTful APIs task.
Chapter 5 goes into detail of the main operational phases of the architecture.
Chapter 6 describes the implementation of the architecture.
Chapter 7 explains the system of Anomaly detection as-a-Service running on AWS machines.
Chapter 8 presents the case study of Eisenmann Italia S.r.l. company.
Chapter 9 includes the experiments conducted on the case study with both supervised and unsupervised approaches, and the results obtained.
Conclusions are finally drawn in Chapter 10.

2. Related Literature

This Chapter collects studies and works with the goal of presenting Machine Learning as-a-Service solutions and different methods of Anomaly detection in the literature, in order to give a big picture of the state of the art.
In the literature of Machine Learning as-a-Service most of the efforts focused on images, video and textual information and the applications are far from Anomaly detection. Tay [2] applied deep learning algorithms within the area of wood identification, to create an "Artificial Intelligence as-a-Service" (AIaaS) product. Since their applied method, deep learning is part of ML, their product is also a MLaaS product. Their tool consists of individual cloud-based services for data collection, data verification, model training, internal testing, and inference. The authors trained their model with macroscopic wood anatomy images and information about type and origin of the wood. Their goal is to fight illegal logging by harming the black market for illegal wood. Instead, to tackle the cost of bug fixing within software engineering, modern debugging and testing methods moved from finding to predicting the bugs [3]. The field of bug fixing is another area that leverages ML predictions. Subbiah et al. [3] apply Microsoft's MLaaS platform Azure Machine Learning to leverage the easy model deployment via the cloud web service. By proposing bug prediction via MLaaS the authors provide Bug Prediction as-a-Service (BPaaS).
MLaaS can be leveraged in other areas as well. Within the field of Urban Modelling, the model can be used to enable another service model, Urban Modelling as-a-Service (UMaaS). Milton and Roumpani [4] propose the application of deep learning networks within an "as-a-Service" setting, to enable UMaaS. The system is providing urban planners with the capabilities to develop applications, and see results immediately, thanks to the increased speed by leveraging ML. The authors further propose a platform architecture, to enable sharing of modelling data, such as zone boundaries or trips and cost matrices. Curating the shared data provides another essential service by ensuring quality standards.
One of the challenges within MLaaS are attacks on network data. Rajagopal et al. [5] propose a solution within network intrusion detection applying MLaaS and by that increasing time-efficiency. The goal is to correctly classify an attack as such. The authors further analyzed the performance, training-, and prediction time of different algorithms on Microsoft's Azure Machine Learning Studio Platform (MAMLS) by applying all algorithms to the same dataset. The best accuracy and precision were accomplished by the decision forest

algorithm. Using Microsoft's MLaaS platform MAMLS enabled the authors to conduct this otherwise extremely time-consuming experiment.
Starting with statistical approaches of Anomaly detection, Histogram-based Outlier Score (HBOS) [6] generates an histogram for each independent feature of the given dataset. The values of the features of all the available data points are first used to build histograms; at a later stage, for each data point, the anomaly score is calculated as the multiplication of the inverse heights of the columns in which each of its features falls. In general histogram-based techniques are relatively simple to implement, but a key shortcoming of such techniques for multivariate data is that they are not able to capture the interactions between different attributes. An anomaly might have attribute values that are individually very frequent, but their combination is very rare, but an attribute-wise histogram based technique would not be able to detect such anomalies.
Sohn et al. [7] have presented time series analysis for fault source in mechanical systems. A two stage model combined with autoregression and exogenous input technique is used for damage location.
Outlier Detection using Indegree Number (ODIN) stems from Kth-Nearest Neighbor [8] and it examines the whole dataset to determine their feature distances to the given point. This allows isolating nearest neighbors(NN), creating the kNN graph. Differently from kNN, ODIN defines as anomalies the data points that have a low number of in-adjacent edges in the kNN graph.
Chen et al. [9] have presented a detection method that integrates one-class SVM with a pre-defined Autoregressive Integrated Moving Average (ARIMA) regression process. Meanwhile, an automatic cyclic-analysis method is also developed as a preprocessing to suppress temporal non-stationarity in condition signal before feeding it to the monitoring process. As such, a novel framework of continuous monitoring of condition signal is presented to inspect whether an unexpected running status change occurs or not during continuous machine operations. The proposed framework is applied to three representative condition monitoring applications: external loading condition monitoring, bearing health condition assessment, and rotational speed condition monitoring.
Vibration analysis is a widely used condition monitoring technique for high-speed rotating machinery. Using the information contained in the vibration signals, an automatic method for bearing fault detection and diagnosis is presented in the work of Fernández-Francos et al. [10]. Initially, a one-class v-SVM is used to discriminate between normal and faulty conditions. In order to build a model of normal operation regime, only data extracted under normal conditions is used.
A methodology was proposed by Sampaio et al. [11] to treat and convert the collected data of vibration measurements from a vibration system that simulated a motor and to build a dataset in order to train and test an ANN model capable of predicting the future condition of the equipment, predicting when a failure can happen. The methodology involves the use of frequency and amplitude data by classifying the dataset and defining a way of calculating the vibrating system's failure time.
Recorded failure data is often only valid for the specific circumstances and components for which it was collected. The work of Sobie et al. [12] directly addresses these challenges for roller bearings with race faults by generating training data using information gained from high resolution simulations of roller bearing dynamics, which is used to train ML algorithms that are then validated against four experimental dataset.
Zimnickas et al. [13] built a test workbench for collecting vibration signals from several different induction motors whose conditions can be listed as: bad bearings, loose mounting, rotor eccentricity, lost phase to motor and short circuit in stator winding, and they applied a hybrid method of Continuous Wavelet Transform (CWT) and CNN to 2D representations of their collected data.
Generative models aim to learn exact data distribution in order to generate new data points with some variations. The two most efficient generative approaches is Generative Adversarial Networks (GAN) [14]. A variant of GAN architecture known as Adversarial Autoencoders (AAE) that use adversarial training to impose an arbitrary prior on the latent code learned within hidden layers of autoencoder are also shown to learn the input distribution effectively. Leveraging this ability of learning input distributions, several Generative Adversarial Networks-based Anomaly detection (GAN-AD) frameworks [15] proposed are shown to be effective in identifying anomalies on high dimensional and complex dataset.

3. Background

This Chapter collects the main notions required to fully understands various aspects of this work. More specifically, Section 3.1 presents the concept of anomalies and the objective of Anomaly detection along with considerations on various approaches depending on the type of data. In Section 3.2 the statistical methods for Anomaly detection are shown. In Section 3.3 a brief summary on Machine Learning will be presented, along with various methods of ML for Anomaly detection; while in Section 3.4 a small consideration on Deep Learning is presented. Section 3.5 gives the basics of theory of signal. Lastly, Section 3.6 gives an overview on the current acceleration sensors solutions for vibration measurements.

3.1. Anomaly Detection

The most common definition of anomalies is the following: "Anomalies are patterns in data that do not conform to a well defined notion of normal behavior." Chandola et al. [16].
Anomalies can appear in different forms. Commonly, three different types of anomalies exist:

- **Point anomalies**: The majority of work in literature focuses on point anomalies. If a point deviates significantly from the rest of the data, it is considered a point anomaly (Figure 1). Hence, a point X_t is considered a point anomaly, if its value differs significantly from all the points in the interval $[X_t-k, X_t+k]$, $k \in R$ and k is sufficient large.

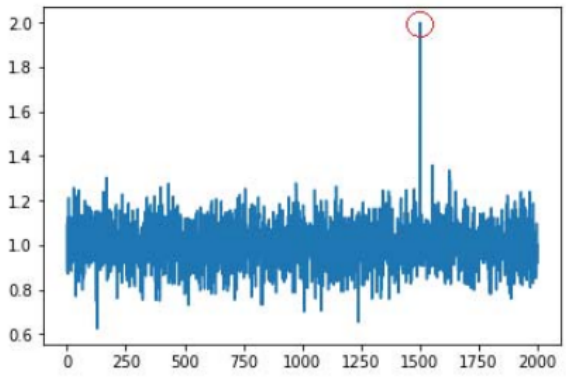

Figure 1: Point anomaly (circled in red) in random Gaussian noise

- **Collective anomalies**: If a collection of related data instances is anomalous with respect to the entire data set, it is termed as a collective anomaly. The individual data instances in a collective anomaly may not be anomalies by themselves, but their occurrence together as a collection is anomalous (Figure 2).

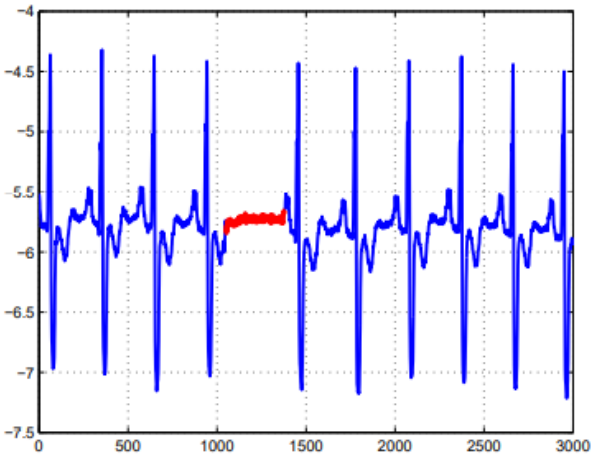

Figure 2: Collective anomaly in simulated ECG time series (marked in red)

- **Contextual anomalies**: Contextual anomalies are observations or sequences which deviate from the expected patterns within the time series, however, if taken in isolation they may be within the range of values expected for that signal. When taken in the context of the surrounding observations (Figure 3) a contextual anomaly is a deviation from the norm.

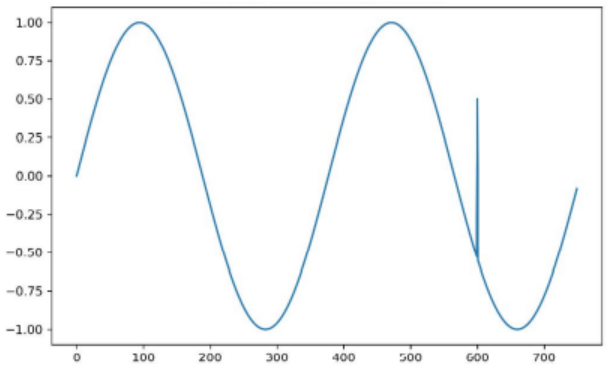

Figure 3: Contextual anomaly, the anomalous value at 600 is the same as a number of other observations however in the context this value is anomalous

Anomaly detection has the objective of detecting rare data points that deviate remarkably from the general distribution of the dataset. The amount of deviation is usually regarded as a measure of strength of the anomaly or probabilistically the likelihood of being an anomaly [17]. Thus formally, Anomaly detection can be defined as a function ϕ:

$$\phi : \mathbb{R}^n \to \mathbb{R}$$
$$\phi(x) \mapsto \gamma$$

where γ is the anomaly score, $x \in X \subseteq \mathbb{R}^n$, and X is the dataset.
To convert the continuous value γ into a binary label, a threshold $\delta \in \mathbb{R}$ is defined where all points with an anomaly score greater than δ are marked as an anomaly. Thus, let $\phi_{score} := \phi$, then the binary labeling Anomaly detection method ϕ_{binary} can be defined as:

$$\phi_{binary} : \mathbb{R}^n \to \{normal, anomaly\}$$

$$\phi_{binary}(x) \mapsto \begin{cases} \text{anomaly}, & \text{if } \phi_{score}(x) > \delta \\ \text{normal}, & \text{otherwise} \end{cases}$$

In order to achieve satisfying results in Anomaly detection, the proper Anomaly detection method has to be selected which is dependent on the properties of the inspected data. The following properties are important for selecting the appropriate approach:

1. Temporal vs Non-Temporal data: Non-Temporal data can be medical images, protein sequences etc. Temporal data include time-series, but also data with timestamps of unequal interval.
2. Univariate vs Multivariate data: Univariate data takes only one dimension, e.g. the stock price, while multivariate data contains multiple dimensions. Instances of multivariate data are images or time-series observed by several sensors.
3. Labeled or unlabeled data: A dataset is labeled if an annotation exists for each element in the dataset, which determines if it is a normal or anomalous data point. A labeled dataset with normal and anomalous points is the object of supervised Anomaly detection methods. It is possible that the dataset is completely labeled but only consists of normal points. Then, it can be analysed by semi-supervised methods. Finally, unlabeled data is the object of unsupervised Anomaly detection methods.

Anomaly detection methods can be divided in three main categories:

- Statistical methods.
- Classical Machine Learning methods.
- Methods using neural networks (Deep Learning).

3.2. Statistical Methods

These methods utilize historical data to model the expected behavior of a system. When a new observation is received it is compared against the current model for that system and if it does not fit within that model it is registered as an anomaly [18].

3.2.1. ARMA Model

Let X_t be the signal X at time t, it is assumed that X_t depends linearly on the previous values $X_{t-1}, \ldots, X_{t-p}$ where p is the order of auto-regression. Thus the Auto-Regressive (AR) model of order p, $AR(p)$ is defined as:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t$$

where $\phi_1, ..., \phi_p$ are auto-regression parameters that can be learned based on historical data and used to predict or find similar time-series, c is a constant and $\epsilon_t \sim N(0, \sigma^2)$ is a Gaussian noise.
The Moving Average (MA) model of order q regresses on the noise terms ϵ_t. Thus $MA(q)$ is defined as:

$$X_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

where $\theta_1, \ldots, \theta_q$ are learnable parameters of the model, μ is the expectation of X_t and $\epsilon_{t-i} \sim N(0, \sigma^2)$ are independent and identically distributed (iid) Gaussian noise terms.
It is possible to combine the two complimentary views of modeling X_t into a single equation giving rise to the Auto-Regressive Moving-Average model, $ARMA(p, q)$:

$$X_t = c + \epsilon_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

Using historical data, it is possible to select p and q in $ARMA(p, q)$ model and learn the model coefficients θ and ϕ in order to make a future prediction. A substantial deviation, from the prediction result in time-series anomaly, could be two standard deviation from a moving average of X. The model parameters θ and ϕ are learned via Maximum Likelihood Estimation (MLE).

3.2.2. ARIMA Model

One of the main problems with dataset is the fact that they can be non-stationary. Stationarity is a precondition for models like ARMA. The ARIMA model is a generalization of the ARMA model. In addition to the p and q parameter, it is also defined by a d parameter which defines the number of times the time-series is differenced. For $d = 1$, the time-series $x_0, ..., x_T$ is differenced as follows:

$$X_i^{'} = X_i - X_{i-1}, \forall i \in 1, ..., T$$

Differencing removes a trend in the time-series resulting in a constant mean. If the trend is non-linear, differencing must be done several times, thus, $d > 1$. Differencing is also used to remove seasons. The Seasonal Differencing is as follows:

$$X_t^{'} = X_t - X_{t-n}$$

where n is the duration of the season.
After fitting the ARIMA model, anomalies are detected by evaluating the deviation of the predicted point to the observed one.

3.3. Machine Learning methods

Machine Learning (ML) is a subset of Artificial Intelligence (AI). Artificial Intelligence is a generic term and refers to systems or machines that mimic human intelligence, although Machine Learning creates systems that learn or improve performance based on the data they use. In particular, ML is the study of computer algorithms which are able to improve automatically in solving a certain task. Machine Learning tasks are usually described in terms of how the ML system should process an example. An example is a collection of features that have been quantitatively measured from some object or event that we want the ML system to process [19]. Many kinds of tasks can be solved with ML algorithms. Some of the most common ML tasks include: classification, clustering, transcription, machine translation, Anomaly detection etc. ML algorithms use a mathematical model which could be optimized using sample data (training data) or not.
A model represents what was learned by a ML algorithm. The model is the "thing" that is saved after running a ML algorithm on training data and represents the rules, numbers, and any other algorithm-specific data structures required to make predictions.

There are at least three different paradigms used to build the model, based on the availability of the labels in the data. Supervised learning, Semi-Supervised Learning and Unsupervised Learning. Since the work is focused on the identification of anomalies the paradigms are presented in this perspective (Figure 4), through a clear categorization [20].

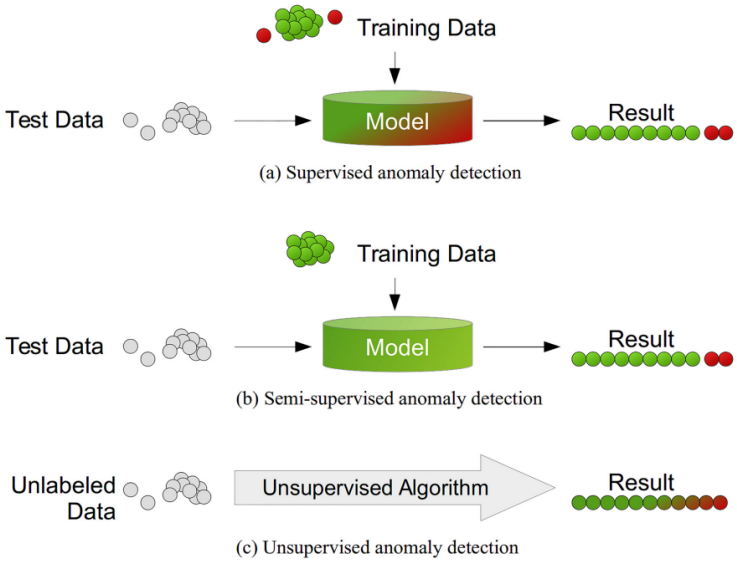

Figure 4: Different Anomaly detection modes depending on the availability of labels in the dataset

Supervised Anomaly detection. Techniques trained in supervised mode assume the availability of a training data set which has labeled instances for normal as well as anomaly class. Typical approach in such cases is to build a predictive model for normal vs. anomaly classes. Any unseen data instance is compared against the model to determine which class it belongs to. There are two major issues that arise in supervised Anomaly detection. First, the anomalous instances are far fewer compared to the normal instances in the training data (imbalanced class). Second, obtaining accurate and representative labels, especially for the anomaly class is usually challenging.
Semi-Supervised Anomaly detection. Techniques that operate in a semi-supervised mode, assume that the training data has labeled instances for only the normal class. Since they do not require labels for the anomaly class, they are more widely applicable than supervised techniques. The typical approach used in such techniques is to build a model for the class corresponding to normal behavior, and use the model to identify anomalies in the test data.
Unsupervised Anomaly detection. Techniques that operate in unsupervised mode do not require label training data and thus are most widely applicable. The techniques in this category make the implicit assumption that normal instances are far more frequent than anomalies in the test data. If this assumption is not true then such techniques suffer from high false alarm rate.

3.3.1. Clustering-Based

Clustering is an extensively studied data mining technique that groups data into multiple clusters with similar data instances ending up in the same cluster. Anomaly detection algorithms based on clustering usually take a two-step procedure: grouping the data with clustering algorithms and analyze the degree of deviation based on the clustering results. As pointed as by Aggarwal [21], there is a complementary relationship between clusters and outliers, which can be simplistically put as that a data point not belonging to any clusters is considered an outlier. Aside from the cluster membership (whether or not in a cluster), there are two other commonly used cluster-related quantities to construct an outlier score. The first is the distance to the cluster center, the assumption being that normal data points are close to the cluster centers, whereas the outliers are far from them. The second is the cardinality of a cluster, the assumption being that the cluster of normal data points is dense and large, whereas the cluster of outliers is sparse and small. In conclusion, those observations which present close and within dense clusters are indicated as normal observations while those which present further away from, or do not belong to, these clusters are reported as anomalous [16].
K-means Clustering (K-Means) is a popular clustering algorithm that groups data points into k clusters by their feature values [22]. First, the k cluster centroids are randomly initialized. Then, each data record is assigned to the cluster with the nearest centroid, and the centroids of the modified clusters are re-calculated. This process stops when the centroids are not changing anymore. Data points are put in the same cluster when they have

similar feature values according to a given distance function. Finally, scores of each data point inside a cluster are calculated as the distance to its centroid. Data points which are far from the centroid of their clusters are labeled as anomalies.
Local Density Cluster-based Outlier Factor (LDCOF) estimates the density of clusters generated by using the K-means clustering algorithm, which are separated into small and large groups [23]. For each cluster, the average distance of all its data points to the centroid is calculated, normalized by the average distance of the data points of this cluster to its centroid, and used as anomaly score. Therefore, expected data points result in smaller scores i.e., close to 1.0, because their densities are as big as the densities of their cluster neighbors. Instead, anomalies will result in larger scores, since their densities are smaller than the densities of their neighbors.
The advantages of clustering based techniques are that they can operate in an unsupervised mode. They can often be adapted to other complex data types by simply plugging in a clustering algorithm that can handle the particular data type and 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 that performance is highly dependent on the effectiveness of clustering algorithm in capturing the cluster structure of normal instances, some algorithms force every instance to be assigned to some cluster in fact 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. Another disadvantage is that some clustering based techniques are effective only when the anomalies do not form significant clusters among themselves.

3.3.2. Nearest Neighbor-Based

The concept of nearest neighbor analysis has been used in several anomaly detection techniques. Such techniques are based on the following key assumption: normal data instances occur in dense neighborhoods, while anomalies occur far from their closest neighbors.
Nearest neighbor based anomaly detection techniques require a distance or similarity measure defined between two data instances. Distance (or similarity) between two data instances can be computed in different ways. Nearest neighbor-based outlier detection approaches measure the degree of abnormality on the basis of a data point's relation to its nearest neighbors. There are two main ways to define the neighborhood: k nearest neighbors and a neighborhood within a pre-specified radius, centered by a data point. The underlying assumption is that normal data instances are closer to their neighbors, thus forming a dense neighborhood, whereas outliers are far from their neighbors, thus sparsely populated.
Kth-Nearest Neighbor (kNN) was primarily designed to identify outliers. For each data point, the whole set of data points is examined to extract the k items that have the most similar feature values: these are the k nearest neighbors (NN). Then, the data point is classified as anomalous if the majority of NN was previously classified as anomalous. Note that while the nearest neighbours are gathered in an unsupervised manner, the final classification needs some labels in the training set.
The advantages of nearest neighbor based techniques are that they are unsupervised in nature and do not make any assumptions regarding the generative distribution for the data; instead, they are purely data driven. Moreover, adapting these 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 that for unsupervised techniques, if the data has normal instances that do not have enough close neighbors or if the data has anomalies that have enough close neighbors, the technique fails to label them correctly, resulting in missed anomalies. Another disadvantage is that the computational complexity of the testing phase is also a significant challenge since it involves computing the distance of each test instance with all instances belonging to either the test data itself, or to the training data, to compute the nearest neighbors.

3.3.3. Classification-Based

Classification is used to learn a model (classifier) from a set of labeled data instances (training) and then, classify a test instance into one of the classes using the learnt model (testing). 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. Based on the labels available for training phase, classification based anomaly detection techniques can be grouped into two broad categories: multi-class and one-class anomaly detection techniques (Figure 5).

- Multi-class classification based anomaly detection techniques assume that the training data contains labeled instances belonging to multiple normal classes.
- One-class classification based anomaly detection techniques assume that all training instances have only

one class label. Such techniques learn a discriminative boundary around the normal instances using a one-class classification algorithm.

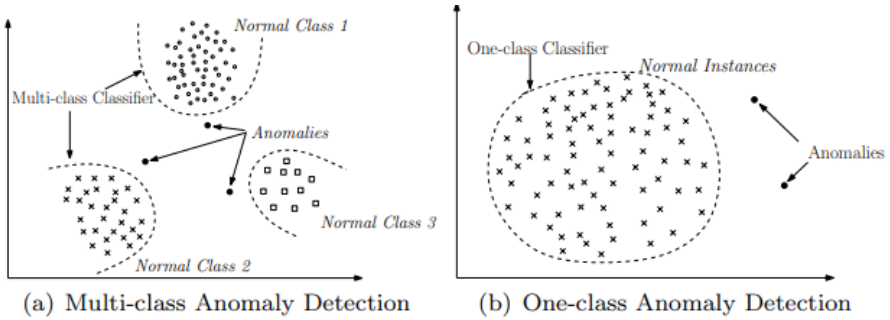

Figure 5: Multi-class and One-class classification for anomaly detection

One-class Support Vector Machine (one-class SVM) is an algorithm [24] that aims at learning a decision boundary to group the data points. Therefore it can be used for unsupervised anomaly detection, despite at first supervised support vector machines (SVMs) were used only for semi-supervised anomaly detection. In the unsupervised anomaly detection scenario, the one-class SVM is trained using the dataset and afterwards, each instance in the dataset is scored by a normalized distance to the determined decision boundary [24]. The parameter v needs to be set to a value lager than zero such that the contained anomalies are correctly handled by a soft-margin.
Isolation Forest (IF) structures data points as nodes of an isolation tree, assuming that anomalies are rare events with feature values that differ a lot from expected data points. Therefore, anomalies are more susceptible to isolation than the expected data points, since they are isolated closer to the root of the tree instead of the leaves. It follows that a data point can be isolated and then classified according to its distance from the root of the tree.
The computational complexity of classification based techniques depends on the classification algorithm being used. Generally, training decision trees tends to be faster while techniques that involve quadratic optimization, such as SVMs, are more expensive. The testing phase of classification techniques is usually very fast since the testing phase uses a learnt model for classification.

3.4. Deep Learning methods

Deep learning (DL) is part of a broader family of ML methods based on artificial neural networks (ANN). Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks (RNNs) and convolutional neural networks (CNNs) have been applied to fields including computer vision, speech recognition, natural language processing, machine translation etc., where they have produced results comparable to and in some cases surpassing human expert performance.
Autoencoders are the fundamental unsupervised deep architectures used in anomaly detection [25]. The deep unsupervised models proposed for anomaly detection rely on the assumption that unsupervised anomaly detection algorithm produces an outlier score of the data instances based on intrinsic properties of the data-set such as distances or densities. The hidden layers of deep neural network aim to capture these intrinsic properties within the dataset. The autoencoders are the most common architecture employed in outlier detection with quadratic cost, the computational complexity of training an autoencoder is much higher than traditional methods.
The advantage of unsupervised deep anomaly detection techniques is that they learn the inherent data characteristics to separate normal from an anomalous data point. This technique identifies commonalities within the data and facilitates outlier detection.
The significant disadvantages of unsupervised deep anomaly detection techniques are that often it is challenging to learn commonalities within data in a complex and high dimensional space and while using autoencoders the choice of right degree of compression, i.e., dimensionality reduction is often an hyper-parameter that requires tuning for optimal results.

3.5. Basics of Signals and Fourier Transform

A signal is a function that conveys information about the behavior of a system or attributes of some phenomenon [26].

Signals can be categorized in various ways. The most common distinction is between discrete and continuous spaces that the functions are defined over, for example, discrete and continuous-time domains.
Discrete-time signals are often referred to as time series in other fields. Continuous-time signals are often referred to as continuous signals.
Focusing on discrete time signals, the time-domain representation gives the amplitudes of the signal at the instants of time during which it was sampled. However, in many cases, it is interesting to know the frequency content of a signal rather than the amplitudes of the individual samples. In the frequency domain, one can separate conceptually the sine waves that add to form the complex time-domain signal. Figure 6 shows single-frequency components, which spread out in the time domain, as distinct impulses in the frequency domain. The amplitude of each frequency line is the amplitude of the time waveform for that frequency component.
The representation of a signal in terms of its individual frequency components is the frequency-domain representation of the signal. The frequency-domain representation might provide more insight into the signal and the system from which it was generated.
In conclusion, every signal in the real world is a time signal and is made up of many sinusoids of different frequencies. So, time domain signal can be converted into the frequency domain to view different frequency components. Fourier transform is a function that transforms a time domain signal into frequency domain. The function accepts a time signal as input and produces the frequency representation of the signal as an output. Fourier transform does not change the signal. It just provides a different view to analyze a time signal because some properties and features of the signal can be fully explored in the frequency domain.

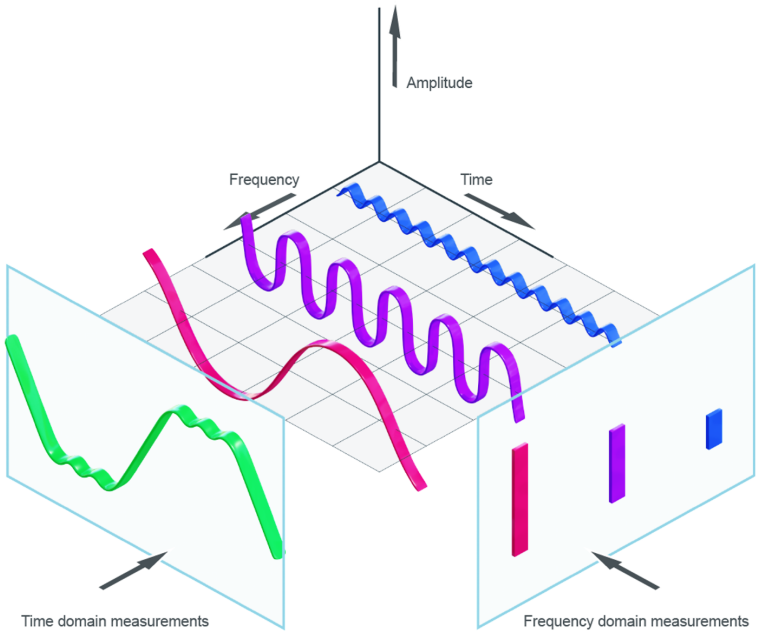

Figure 6: Time domain monitors signal amplitude over time. Frequency domain monitors signal amplitude over frequency

3.6. Vibration and Triaxial Accelerometer

Every rotating machine has its own vibration signature made up of the combined individual parts. The amplitude of that signature changes when the machine experiences unusual operation, such as unbalance, bearing wear, excessive looseness, and more. Vibration analysis is defined as a process for measuring the vibration levels and frequencies of machinery and then using that information to analyze how healthy the machines and their components are. Vibration measurements can be performed using a sensor that is placed near the element to be monitored. The most commonly used sensor for measuring vibrations is an accelerometer, or acceleration sensor. This name fits well with the sensor, because the sensor measures the forces that occur during acceleration and deceleration. During a vibration, in fact there is continuous acceleration and deceleration.
An accelerometer can measure vibration in up to three axis, often referred to as the X, Y, and Z axis.
A vibration can be detected as movement from side-to-side, forward/backward, and up/down. To be able to

see the full movement of the vibrations, triaxial accelerometer is the best choice. It gives the simultaneous measurement of vibrations in three directions.
Accelerometers can be of the piezoelectric type or of the MEMS (micro electro mechanical system) type. Piezoelectric accelerometers depend on changes in electric current due to motion. Moreover they are by far the most popular in industrial testing applications. Capacitive MEMS accelerometers have become very popular in embedded electronics with their low cost and small size, but piezoelectric still reigns supreme with mechanical engineers performing shock and vibration tests. Here are some of their benefits:

- Low noise output.
- Wide range, suitable for low amplitude vibration test as well as high amplitude shock.
- Excellent linearity over their dynamic range.
- Wide frequency range.
- Self-generating, no external power required.
- Compact yet highly sensitive.

Both piezoelectric and MEMS accelerometers can be used to monitor low and high frequency vibrations.

4. Architecture

The work aims to build a client-server architecture that allows to offer Anomaly detection as-a-Service via RESTful APIs. The system is based on REST (REpresentational State Transfer) paradigm, following the principles of:

- **Client-server**, the client-server design pattern enforces the principle of separation of concerns: separating the user interface concerns from the data storage concerns. Portability of the user interface is thus improved. Separation also simplifies the server components, improving scalability.
- **Statelessness**, there is no session information that is retained by the receiver, usually a server.
- **Cacheability**, clients and intermediaries can cache responses. Responses must, implicitly or explicitly, define themselves as either cacheable or non-cacheable to prevent clients from providing stale or inappropriate data in response to further requests.
- **Layered system**, a client cannot ordinarily tell whether it is connected directly to the end server or to an intermediary along the way. If a proxy or load balancer is placed between the client and server, it will not affect their communications, and there will not be a need to update the client or server code.
- **Uniform interface**, specific resources should be identified in the request. Usually, they are described by Uniform Resource Identifier (URI). Moreover, internal implementation is separate from resource representation.

REST works on top of the HTTP transport. It takes advantage of HTTP's native capabilities, such as GET, PUT, POST and DELETE. When a request is sent to a RESTful API, the response (the "representation" of the information "resource" being sought) returns in either the JSON, XML or HTML format. A RESTful API is defined by a web address or URI, which typically follows a naming convention.
The Anomaly detection as-a-Service solution allows the client to make requests to a server, in order to train ML models for the identification of anomalies from the data sent by the client and to make inference on a single data from the previously trained models. Since the identification of anomalies can refer to a wide range of applications and consequently data, in this work the focus is on data collected by accelerometers installed on industrial machines, without any loss of generality. Accelerometers will sample at a certain sampling rate and the data will be saved within Csv files.

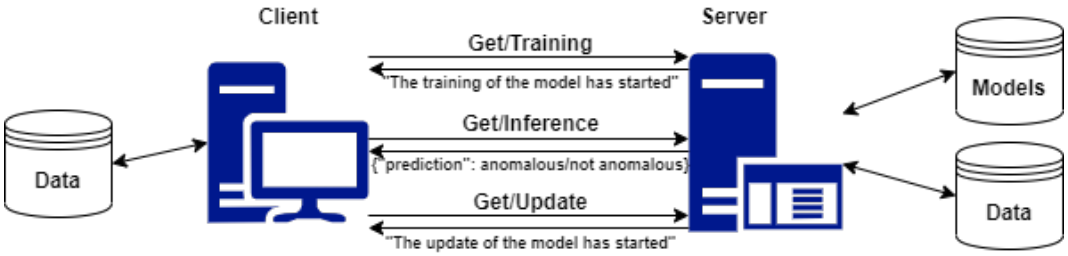

Figure 7: Client-Server architecture for Anomaly detection as-a-Service

The client-server architecture for Anomaly detection as-a-Service is shown in Figure 7 and it is designed to meet the following client requirements:

1. **Training** of a model.
2. **Inference** of a data sample.
3. **Update** of a model.

Client side, the Python script has been developed in order to make easy for the client the use of the different services. The script allows to take a single data or a series of data as input and serialize them in JSON format, to send the request and the JSON (in its body) at the appropriate server API.
Server side it is necessary to implement the APIs defined by appropriate URLs. They must allow the client to reach the server and send the above mentioned requests along with the data in JSON format. After defining the APIs, the server is ready to receive the client requests and execute the appropriate "service" code.
The three services are analyzed in detail for both client and server.

4.1. Training

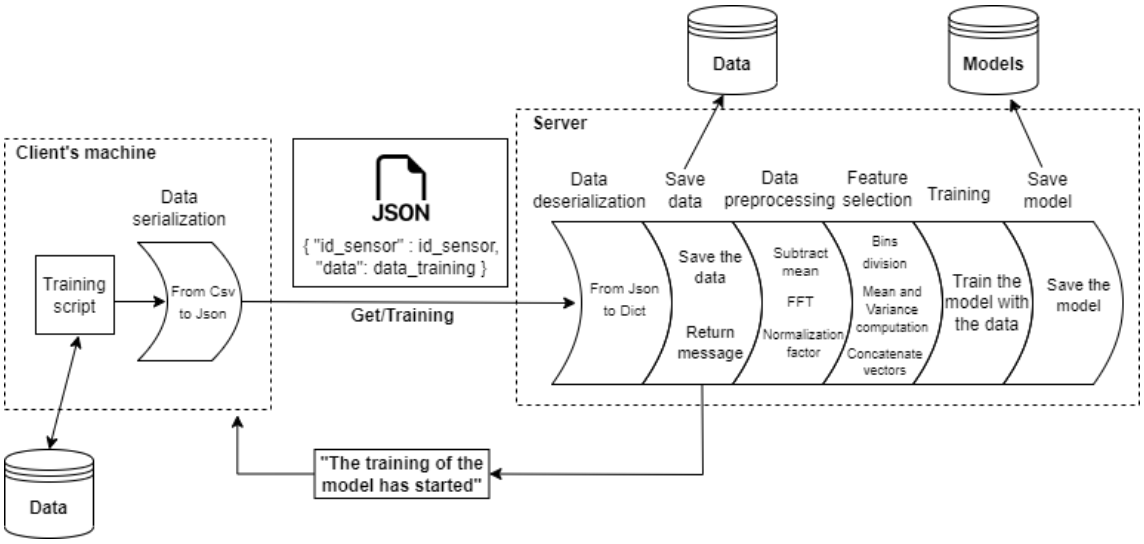

Figure 8: Operations of the model training

The client-side Python script:

- Takes as input the path of a folder containing the Csv files related to the data samples of a sensor, in order to carry out the training of the model.
- Transforms the Csv files into a single JSON with the structure displayed in the Figure 8, where id_sensor is the identification of a sensor.
- Makes a GET request containing the JSON to the appropriate API of the server.

The server listening:

- Receives the GET request from the client with the JSON in the body of the request.
- Saves the JSON in the DB Data with the appropriate nomenclature, that is the id_sensor (id_sensor.json).
- Sends the answer to the client notifying the start of the training of the model and ends the communication with client.
- Performs the preprocess of the data first and then the selection of the features.
- Performs the training of the ML model using the modified data.
- Finally saves the model in the DB Models with the appropriate nomenclature, that is the id_sensor (id_sensor.joblib).

4.2. Inference

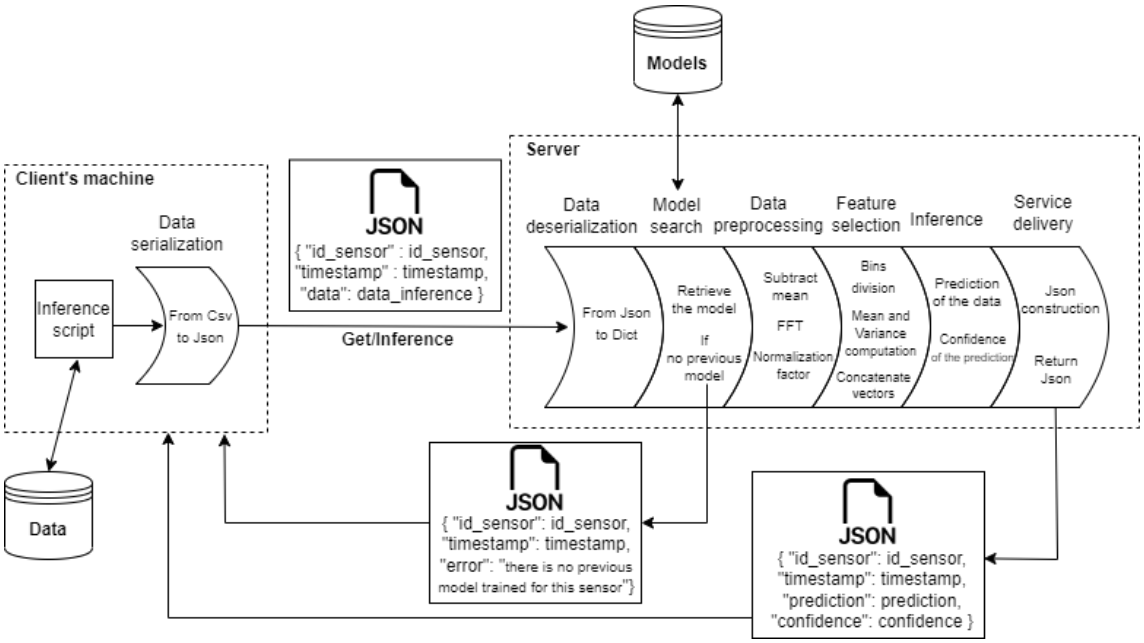

Figure 9: Operations of the inference of a sample

The client-side Python script:

- Takes as input the Csv file related to the data sample of a sensor, in order to carry out the inference of the sample.
- Transforms the Csv file into a JSON with the structure displayed in the Figure 9.
- Makes a GET request containing the JSON to the appropriate API of the server.
- After receiving the reply from the server containing the JSON, it saves the inference result in the folder "results" (if not present is created) as id_sensor.timestamp.json.

The server listening:

- Receives the GET request from the client with the JSON in the body of the request.
- Extracts the model inside the DB Models (if present) through the id_sensor, otherwise the server returns the JSON containing the error and it ends the requested service.
- Performs the preprocess of the data first and then the selection of the features.
- Uses the model to inference the modified data.
- Sends the response to the client containing the JSON with the structure shown in the Figure 9. In particular, the "prediction" key contains the value 1 if the model has evaluated the sample as nominal, otherwise it contains the value -1. Instead, Confidence expresses the ability of the model to provide a given prediction and it is a value between 0 and 1; it is an additional semantics to the single 1/-1 of the prediction, since it makes sense to look at how the measure evolves over time.

4.3. Update

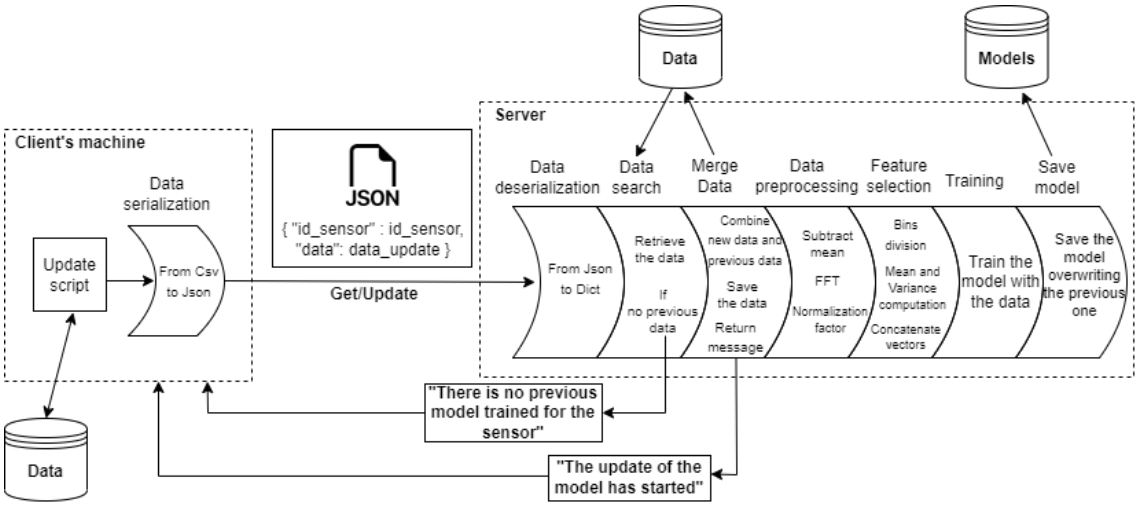

Figure 10: Operations of the model update

The client-side python script:

- Takes as input the path of a folder containing the Csv files related to the data samples of a sensor, in order to update the sensor model with new samples.
- Transforms the Csv files into a single JSON with the structure displayed in the Figure 10.
- Makes a GET request containing the JSON to the appropriate API of the server.

The server listening:

- Receives the GET request from the client with the JSON in the body of the request.
- Extracts the JSON containing the previous sensor data from the DB Data through the id_sensor (if present), otherwise the server returns the error message and it ends the requested service.
- Joins the JSON containing the previous sensor data with the JSON sent by the client, and saves it in the DB Data overwriting the previous JSON.
- Sends the answer to the client notifying the start of the training of the model with the integration of the new data and ends the communication with the client.
- Performs the preprocess of the data contained in the new built JSON first and then the selection of the features.
- Performs model training using the modified data.
- Saves the model in the DB Models overwriting the previous model(Figure 10).

5. Operational Steps

This chapter goes into detail of the various operational phases of the services: starting from data preproccesing, continuing towards feature selection and ending with the ML algorithms used for the Anomaly detection.
Data have been managed and manipulated through the use of Pandas and Numpy Python libraries. Pandas provide high performance, fast, easy to use data structures and data analysis tools for manipulating numeric data and time series [27]. Numpy provides high-performance multidimensional arrays and tools to deal with them [28].

5.1. Data Preprocessing

As noted above, the work focuses on data collected by accelerometers installed on machinery. This means that the data we deal with are signals in time domain. However signals in time domain do not accurately characterize the vibrations captured by the accelerometers.
Therefore the signal has to be transformed in the signal in frequency domain. Mathematically, the "transform" used to go from the time domain to the frequency domain and back is called the Fourier Transform. It is defined as follows:

$$X(f) = \int x(t) e^{-j2\pi ft} dt$$

For a signal $x(t)$ is it possible to get the frequency domain version, $X(f)$, using this formula. Note the t for time, and f for frequency. The j is simply the imaginary unit. To return to the time domain from frequency the procedure is almost the same, aside from a scaling factor and negative sign:

$$x(t) = 1/2\pi \int X(f) e^{j2\pi ft} df$$

The above equation for the Fourier Transform is the continuous form, which can be seen only in math problems. The discrete form is:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi}{N} kn}$$

Note that the main difference is that the integral is replaced with a summation. The index k goes from 0 to $N-1$.
Since this work was done using the Python language, instead of the equation for the discrete Fourier Transform it was used the Fast Fourier Transform function, in particular the fft() function of the SciPy library [29]. The Fast Fourier Transform (FFT) is simply an algorithm to compute the discrete Fourier Transform and converts a signal from time to frequency domain. The FFT function has one input: a vector of samples, and one output: the frequency domain version of that vector of samples (Figure 11).

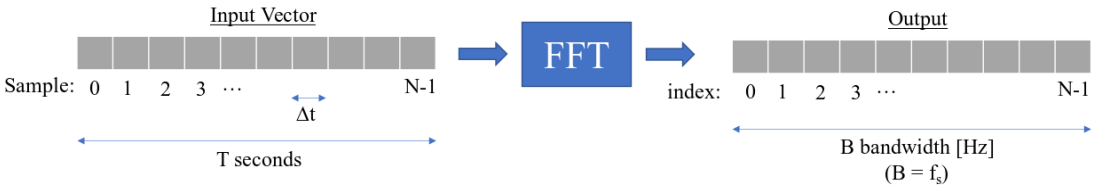

Figure 11: FFT input and output

Basically, if the signals are sampled at a rate f_s, then the FFT will return the frequency spectrum up to a frequency of $f_s/2$.
In particular, the function returns a vector of complex valued frequencies. Since they are complex valued, they will contain a real and an imaginary part. The real part of the complex value corresponds with the magnitude, and the imaginary part with the phase of the signal. In order to take the real part of the frequency spectrum, that is the interested one, the absolute value is applied.
The FFT of an input signal of N points, will return a vector of N points. The first half of this vector ($N/2$ points) contain the useful values of the frequency spectrum from 0 Hz up to the Nyquist frequency of $f_s/2$. The second half contains the complex conjugate and can be disregarded since it does not provide any useful information (Figure 12).

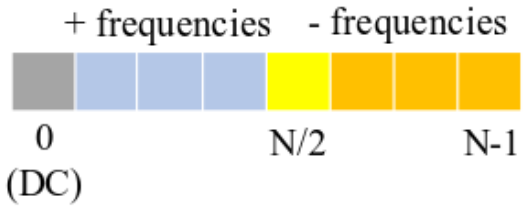

Figure 12: FFT output in detail

Applying the FFT on the signals of interest, one can see particular peaks at 0 Hz. Usually a peak at 0 Hz in the frequency domain indicates the presence of a DC (constant) bias in time series data. Since the data are extracted from accelerometers, it indicates the presence of a static acceleration field influencing the sensors (gravity). This kind of bias, especially for an accelerometer, is not desired. One way to fix this is to use a high pass (or band pass only in specific bands) filter, another way is to subtract the mean of the signal to the signal itself.
Thus the first operation is to subtract the average value from the signal in the time domain in order to remove the gravity acceleration. Then apply the FFT to the signal and take from the output of the FFT function the absolute value of the first $N/2$ points of the signal in the frequency domain. Finally the signal is multiplied by a normalization factor since it scales each element separately to the range 0-1, which is the range for floating-point values where there is the most precision.

5.2. Feature Selection

Feature selection, also called subset selection, aim to select a subset of features that can sufficiently represent the characteristic of the original features. In view of that, this will reduce the computational cost and may remove irrelevant and redundant features and consequently improve learning performance.
The individual signal in the frequency domain contains too many features and thus redundant information. This could cause loss of effectiveness in the predictions of ML algorithms and poor accuracy of the model. To be considered effective it was decided to divide the signal into bins of size of 50 Hz each (from 0 to 50 Hz, from 50 Hz to 100 Hz, etc.).
For each of these bins the mean and variance were calculated. The average takes into account possible peaks that distinguish important frequency bands. Variance, on the other hand, takes into account the variability of the assumed values within the bins and it is an appropriate index in the identification of fluctuations.
As a consequence, there are two vectors, one of the means and the other of the variances of equal size.
The two vectors are concatenated in order to obtain a single vector, which characterizes the spectrum of the single signal in a precise way.
After having defined and exposed the features to be given in input to the ML algorithms, the next step is to present the hybrid approach used in this work. Hybrid because the approach depends on the information contained in the data: if the data are accompanied by labels that allow to add details on the type of input data (nominal, anomalous, etc.), then it will be more correct to use a supervised approach. Conversely, if the data are not accompanied by labels, it will be more effective to use an unsupervised approach, in which all input data are considered nominal. In both cases trained models will be able to learn from these data.

5.3. Unsupervised Approach

It is relatively easy to gather training data of situations that are normal; it is just the standard production situation. But on the other side, collection example data of a faulty system state can be rather expensive, or just impossible. If a faulty system state could be simulated, there is no way to guarantee that all the faulty states are simulated and thus recognized in a traditional binary-class problem.
To cope with this problem, one-class classification solution is introduced. By just providing the normal training data, an algorithm creates a (representational) model of this data. If newly encountered data is too different, according to some measurement, from this model, it is labeled as out-of-class and so anomalous.
This thesis presents the application of Support Vector Machines to this one-class problem. Nominal data after the pre-processing and feature-selection phases are ready to be consumed by the svm.OneClassSVM module of the Scikit-learn library [30].
A fraction of data ν is decided and set as hyperparameter; it represents the upper bound of the number of anomalies present in data. The one-class SVM then tries to find a boundary that encloses regions of high data density excluding at most a fraction ν of data points. It can be seen that boundaries which are linear in the

problem variable space are too simple for most problems. Since SVM is a linear classifier by nature, it is needed to resort to kernel methods to build a flexible model with non-linear boundaries.
The one-class SVM tries to find a hyperplane separating the data from the origin. The idea is to map the data into the kernel feature space and to separate it from the origin with maximum margin using a linear classifier in the kernel feature space. This corresponds to using a non-linear boundary in the original problem space.
Kernels allow to fit simple models in very high dimensional feature spaces without explicitly calculating the high dimensional features. One of the most widely used kernels is the RBF kernel and is the one used to trained the model. SVM always fits a linear model in the kernel feature space even though the decision boundary looks non-linear in the original problem variable space.
Another parameter to tune for one-class SVM with RBF kernel is Gamma. Gamma is a parameter specific to the RBF Kernel and it controls the effect of neighboring points on the decision boundary. Large values of Gamma allow neighboring points to have larger influence on the decision boundary and smaller values of Gamma allow both neighboring and distant points to have an effect on the decision boundary. One way to select Gamma is to let sklearn choose Gamma and it is done by selecting gamma equals to "scale".
The inliers will be on one side of the decision boundary and all the outliers will be on the other side of the decision boundary (Figure 13).

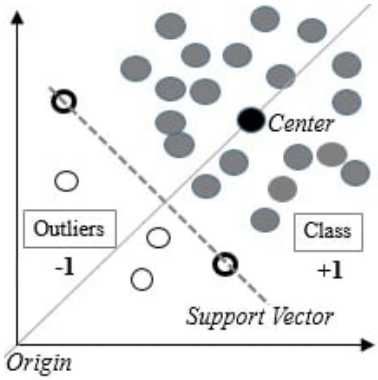

Figure 13: Support Vector Machine with one-class

In the end, the algorithm detects the soft boundary of the set of samples and then return the one-class model through the fit() function. The model is saved in .joblib format. Through the predict(X) function, instead it is possible to perform classification on samples in X, returning $+1$ or -1 according on whether the individual sample is classified as normal or anomalous. Moreover, score_samples(X) returns the (unshifted) scoring function of the samples. In order to obtain the confidence of a prediction it is needed to convert the score in terms of probability though a sigmoidal function (e.g. $\frac{1}{1+exp^{-score}}$).

5.4. Supervised Approach

When both nominal and anomalous data are available, accompanied by labels that defining them, it is possible to use supervised anomaly detection techniques including Neighbors-based classification. It is a type of instance-based learning or non-generalizing learning that does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.
Several nearest neighbors classifiers have been developed to address multi-class classification problems among which, this thesis makes use of neighbors.KNeighborsClassifier module of the Scikit-learn library [30]. It is a classifier implementing the k-nearest neighbors vote and the optimal choice of the value k is highly data-dependent: in general a larger k suppresses the effects of noise, but makes the classification boundaries less distinct.
In this work is used k equals to 3 and since data belong to the nominal class ($+1$), synthetic anomalous data has been created from normal data in order to have an anomalous class (-1). Synthetic data is information artificially manufactured rather than generated by real-world events.
So a supervised model can be trained with both nominal and anomalous data that went through the pre-processing and feature-selection phases, with the appropriate labels.

6. Implementation

This Chapter goes into detail of the implementation of the client-server architecture for Anomaly detection as-a-Service.

6.1. Client

The client can run the Python script passing the appropriate parameters via the terminal command line:

- **service** parameter that can be training or update or inference.
- **path** parameter that can be a path of a Csv file (inference) or a path of a folder containing a series of Csv files (training, update).

```
$ python3 .\client.py --service SERVICE --path PATH
```

This is an example of command line instruction:

```
$ python3 .\client.py --service training --path files/data/sensor
```

Moreover, a configuration file has been written in JSON format that contains the parameters "ip" and "port" relative to the server to make requests. When the server parameters changes, simply edit the configuration file. Python script takes the parameters and after the serialization of the data into JSON format, send the GET request containing the JSON in the body, to the server with the appropriate APIs:

```
r = requests.request('GET', IP:PORT + '/training', json = jsonTraining)

r = requests.request('GET', IP:PORT + '/inference', json = jsonInference)

r = requests.request('GET', IP:PORT + '/update', json = jsonUpdate)
```

Finally the client waits for a response from the server and the communication stops. In the case of training and update a message will be returned to the client. On the contrary the inference of a sample will return a JSON that will be saved in the folder results.

6.2. Server

The server is built with Uvicorn, that is an Asynchronous Server Gateway Interface (ASGI) web server implemented in Python. ASGI not only allows multiple incoming events and outgoing events for the application, but also allows for a background coroutine so the application can do other things. The server starts running via the terminal using the following command:

```
$ uvicorn server:app --host SERVER_IP --port PORT
```

The RESTful APIs, instead are developed in Python with FastAPI web framework. FastAPI fully supports asynchronous programming and can run with Uvicorn.
FastAPI is imported first and then an instance of the Python class FastAPI is created:

```
from fastapi import FastAPI
app = FastAPI()
```

This *app* is the same one referred by Uvicorn in the command.
After that it is necessary to create the various paths operation decorators:

```
@app.get("/training")

@app.get("/inference")

@app.get("/update")
```

The *@app.get("/")* tells FastAPI that the function right below is in charge of handling requests that go to the path "/" using a *get* operation.
Then it is necessary to define the paths operation functions:

```
@app.get("/training")
async def training(request: Request, background_tasks: BackgroundTasks):
    background_tasks.add_task(training_model, await request.json())
```

```
@app.get("/inference")
async def inference(request: Request):

@app.get("/update")
async def update(request: Request, background_tasks: BackgroundTasks):
    background_tasks.add_task(update_model, json_update, json_training)
```

These are Python functions and they will be called by FastAPI whenever they receive respectively a request to the URL "/training", "/inference", "/update" using a GET operation.
Note that, for the training and update functions it necessary to define background tasks to be run after returning a response. This is useful for operations that need to happen after a request, but that the client doesn't really have to be waiting for the operation to complete before receiving the response. In fact the training and update services have to return a message to the client and then continue the training/update of a model. The background tasks for training and update functions are respectively training_model and update_model functions, to which are passed the useful parameters.

7. Amazon EC2

The server code can be deployed into an Amazon Elastic Compute Cloud (Amazon EC2) in order to provide scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates the need to invest in hardware up front and deploy the application faster. One can use Amazon EC2 to launch as many or as few virtual servers as one need, configure security and networking, and manage storage. Moreover, Amazon EC2 enables to scale up or down to handle changes in requirements.
Amazon EC2 provides a wide selection of instance (virtual machine) types optimized to fit different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give the flexibility to choose the appropriate mix of resources for the applications.
In the case study addressed in Chapter 8 has been created an Amazon EC2 T3a instance. It provides a baseline level of CPU performance with the ability to burst CPU usage at any time for as long as required. T3a instance offers a balance of compute, memory, and network resources and is designed for applications with moderate CPU usage that experience temporary spikes in use. In particular, a T3a.large instance was used in which the specifications are the following: 2 vCPU (virtual Central Processing Unit) and 8 GiB RAM (Random-Access Memory).
After having created the virtual computing environment, the server is deployed as systemd service and the server code runs with the Uvicorn command written above. The client will previously provide its Ip address in order to add it on the machine instance to the access management policies. All the Ips belonging to the client's domain are whitelisted, in order to allow communication with the server from that domain.
In this way we have obtained a system of Anomaly detection as-a-Service that actually works on AWS machines and able to provide the services.

8. Case Study

The case study addressed in this thesis is that of Eisenmann Italia S.r.l. company. Eisenmann Italia S.r.l. company provides maintenance, plant optimization and spare parts supply services in the areas of process technology plant building, energy and the environment.
The aim of the Eisenmann Italia S.r.l. company is to build a monitoring system for the identification of anomalies in the perspective of predictive maintenance in Industry 4.0.
The applications of these solutions are machines for the treatment of the air inside of painting systems, that is fans of important dimensions, burners such as boilers that eliminate paint parts etc. The misalignment of these machines could cause the breakage of the load-bearing structures and the malfunction of these types of devices inside the plant is critical because it would lead to the suspension of production.
In addition to the painting plants, among the various applications there are the agricultural bio-gas plants, inside which there are pumping systems such as pumps and load augers. A preventive intervention on these machines would reduce the losses and the total breakage of the plant.
The sensors used are triaxial accelerometers. Having to deal with rotary organs it is extremely important to have the radial direction and axial direction data separately. The radial direction makes account of phenomena such as accumulation of dirt on the blades, problems with the bearings, something that interferes with the rotation of the system. On the other hand the axial rotation to the rotary motion realizes of possible problems that can arise when there are rotating organs that cannot be at first approximation assimilated to a rotating

disk, but that are approximated to two rotating planes. They require an extremely accurate balancing analysis because there is this kind of phenomenology that moves the load-bearing structure of the plant.
The sensors are directly mounted on the machines mentioned above. Figure 14 shows a practical scenario of a triaxial accelerometer positioned on a fan.

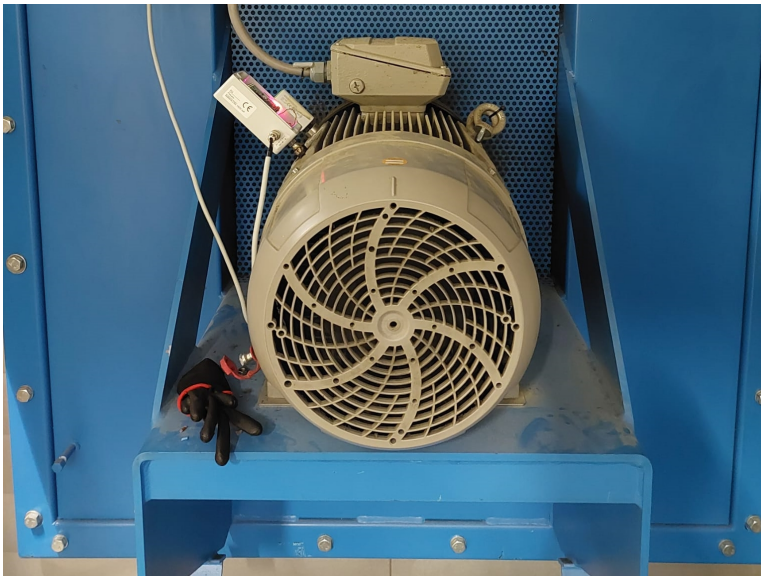

Figure 14: Sensor mounted on a fan

Sampling periods are carried out alternating with periods of inactivity of the sensors, in fact there is not a continuous stream of data on which to carry out a continuous analysis over time, because the amount of data would be high and the relative costs would be exorbitant.
The analysis focuses on all those anomalies that are "constant over time", i.e. failures in the perspective of predictive maintenance, anticipate a break as soon as possible. The type of phenomena on which the case study focuses have a constant pattern over time, where the dynamics of the phenomenon can change (more intense, less intense) but it is not a transient failure, which is therefore limited in time. Therefore it is expected that a machine will have a slow degradation with a certain continuity and not a transient failure.
The sensors have a sampling frequency of 5000 Hz, a resolution of 16 bits and a full scale of 4g. The duration of each sampling is equal to 30 seconds and the sampling rate shall be hourly. Each sensor is operated by a Raspberry Pi via Serial Peripheral Interface (SPI) bus. Communication is via WiFi in a star configuration with an industrial router in the centre, on which a SIM card is installed for data transmission. Via Virtual Private Network (VPN), data is sent to the company's server in Csv text format.
The data are then pre-processed on the server during the data injection into the database (DB). Checks are carried out on the operation of the machines, the presence of "clamping" is verified, that is, the achievement of the full scale and the consistency of the data are verified, since sometimes the sensors get stuck.
The data are then stored in a shared DB so that they can be accessed and each data sample is stored inside a Csv file. The metadata of the sample is contained in the file name, which is composed of the sensor name followed by the sampling start timestamp (date and time).
The Csv file consists of 4 columns:

- **TIME**, is a counter that is incremented in each row and is useful to sort the individual sampling.
- **X**, sensor measurement along the X axis.
- **Y**, sensor measurement along the Y axis.
- **Z**, sensor measurement along the Z axis.

The length of the Csv file is proportional to the sampling frequency and the duration of the sampling.
In this case being the sampling frequency equal to 5000 Hz and continuing to sample for 30 seconds, the length of each Csv file will be 150k rows.
Data collected daily by two triaxial accelerometers mounted on different physical systems are available. These data are obtained from steady-state measurements, they are therefore related to the nominal/correct operation of the systems. For simplicity the sensors will be called Sensor_1 and Sensor_2.
Eisenmann Italia S.r.l. company provided the Sensor_1 and Sensor_2 data, which are respectively composed of approximately 640 and 300 signals each. Moreover, the company have performed a data quality assessment in fact data have the same formats, the same dimensions and no missing values.

8.1. Data Exploration

In the Data Exploration phase the data sets are analyzed to summarize their main characteristics, using statistical graphics and other data visualization methods. All the plots have been created through the Plotly Graphing Library [29].
Figure 15 represents the time series of Sensor_1 and Sensor_2 sampled signals, separated for each component X,Y and Z. The signals in the time domain are very dense since the sampling interval is 0.0002 seconds and the sampling duration is 30 seconds, with a total of 150k points on each axis.

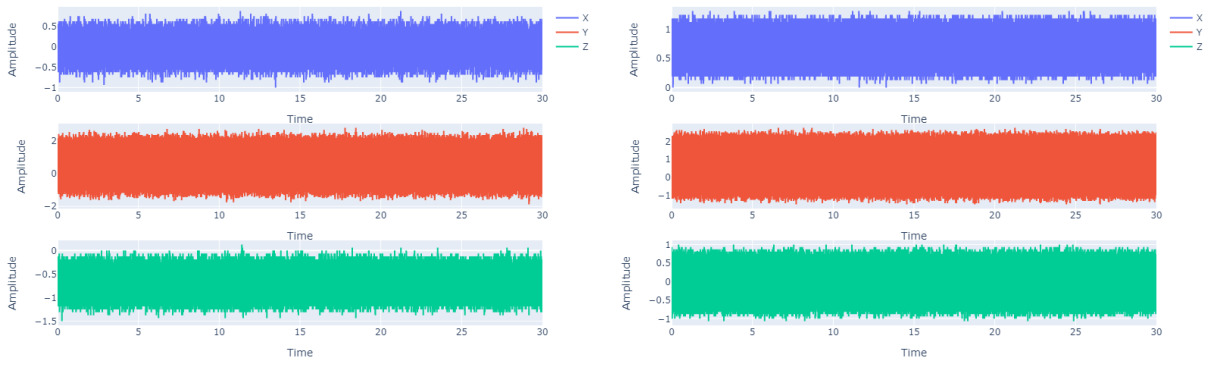

Figure 15: Time series plots of Sensor_1 and Sensor_2 sampled signals

Figure 16 shows the correlation matrix of Sensor_1 and Sensor_2 sampled signals. A correlation matrix is a table that shows the correlation coefficients between a set of variables. It is used to determine which pairs of variables are most closely related and to identify relationships between variables that may not be readily apparent. In this case variable Y might be the most correlated with the others, although it is important to take into account all 3 components X, Y and Z.

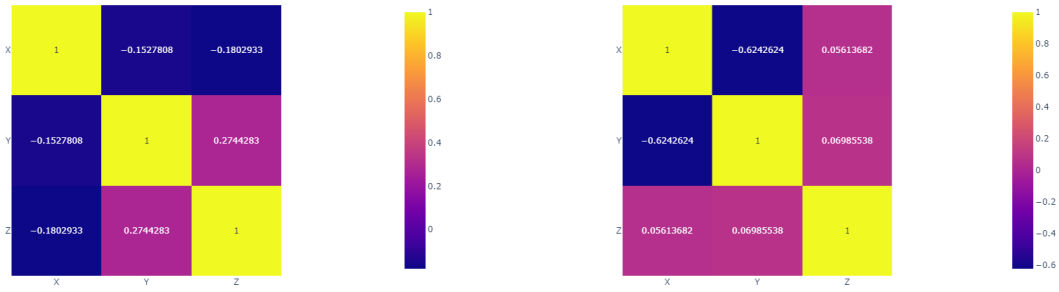

Figure 16: Correlation matrix of Sensor_1 and Sensor_2 sampled signals

Figure 17 shows the box plot of Sensor_1 and Sensor_2 sampled signals. Box plot is a statistical representation of the distribution of a variable through its quartiles. The ends of the box represent the lower and upper quartiles, while the median (second quartile) is marked by a line inside the box. The highlighted outliers would not seem to be errors in the data since they are within full scale (4g).

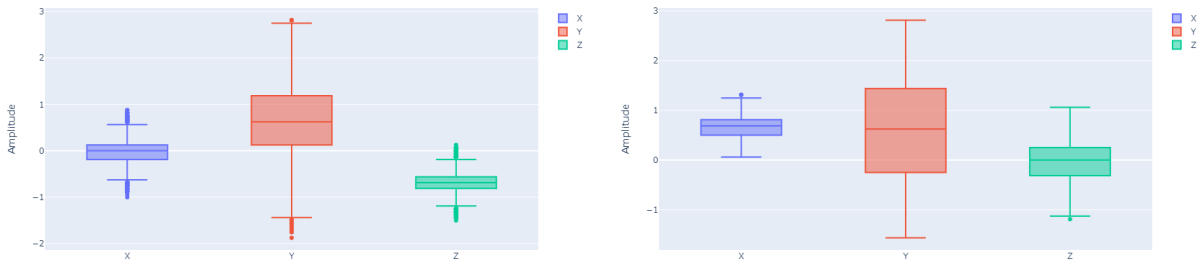

Figure 17: Box plot of Sensor_1 and Sensor_2 sampled signals

In statistics, a histogram is representation of the distribution of numerical data, where the data are binned and the count for each bin is represented, Figure 18 shows the histogram plot of Sensor_1 and Sensor_2 signals data. Almost all variable distributions appear to be symmetrical except one that is bimodal.

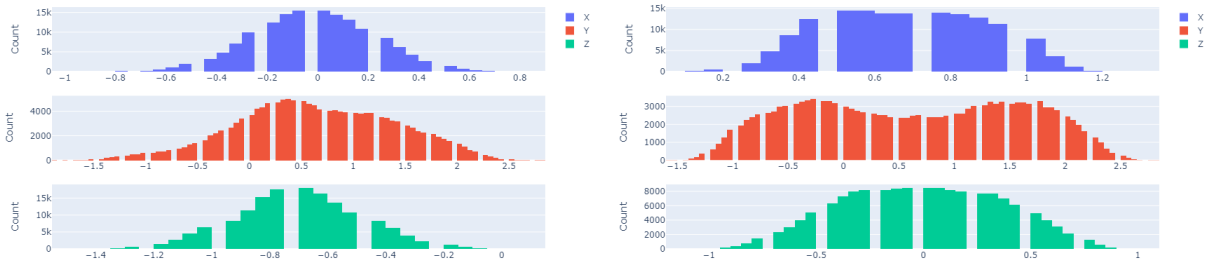

Figure 18: Histogram plot of Sensor_1 and Sensor_2 sampled signals

8.2. Data Preprocessing

Since the signals given are sampled at a rate f_s of 5000 Hz, the FFT will return the frequency spectrum up to a frequency of $f_s/2 = 2500$ Hz. Moreover, Sensor_1 and Sensor_2 signals in the frequency domain will have 75k points for each component (X,Y,Z), rather than 150k points in the time domain for each component.
Thus the first operation is to subtract the average value of each component from the signal in the time domain in order to remove the gravity acceleration. Then apply the FFT to each component of the signal and take from the output of the FFT function the absolute value of the first $N/2$ points of the signal in the frequency domain. Finally the signal is multiplied by a normalization factor since it scales each variable separately to the range 0-1, which is the range for floating-point values where there is the most precision.
The plots of Sensor_1 and Sensor_2 frequency spectrums are shown in Figure 19, for each axis.

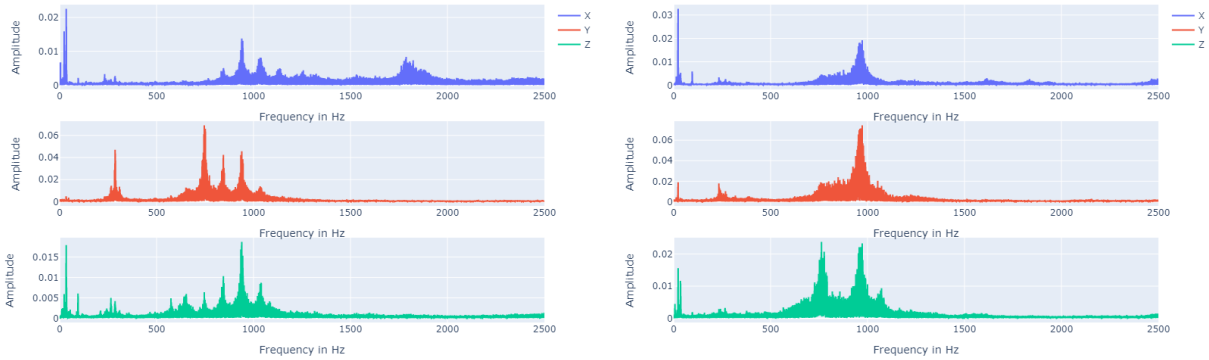

Figure 19: Plots of Sensor_1 and Sensor_2 sampled signals in the frequency domain

In addition Figure 20 shows the distributions of Sensor_1 and Sensor_2 signals in the frequency domain, highlighting the average ($\pm$ standard deviation) distribution.
Interestingly the distributions of both data sets bring out peaks in the frequency range between 600 Hz and 1200 Hz, with a certain correlation between Y e Z axis. This means that both applications on which the sensors are mounted have a similar rotational speed (RPM, revolutions per minute) but their spectrum remains different. It is a first indication that each sensor will have its own model, given their different spectral distribution.

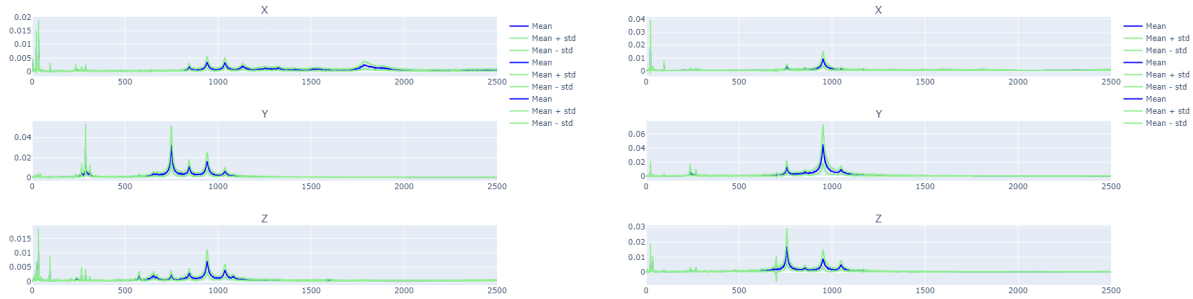

Figure 20: Plots of Sensor_1 and Sensor_2 sampled signals distribution in the frequency domain

Then the 3 components of the signal in the frequency domain are concatenated in order to compact the data (from multivariate to univariate) and the result is that a single sample will have 75k points for each axis with a total of 225k points.

8.3. Feature Selection

Passing through the feature selection, the signal is divided into bins of size of 50 Hz each (from 0 to 50 Hz, from 50 Hz to 100 Hz, etc.). This results in 50 bins for each component (150 in total) describing the signal spectrum exhaustively. For each of these bins the mean and variance were calculated. As a consequence there are two vectors, one of the means and the other of the variances of size equal to 150 each:

$$\begin{bmatrix} X_1 & X_2 \dots X_{50} & Y_1 & Y_2 \dots Y_{50} & Z_1 & Z_2 \dots Z_{50} \end{bmatrix}$$

The two vectors are concatenated in order to obtain a single vector of 300 values, which characterizes the spectrum of the single signal in a precise way.

$$\begin{bmatrix} X_1^m & X_2^m \dots X_{50}^m & Y_1^m & Y_2^m \dots Y_{50}^m & Z_1^m & Z_2^m \dots Z_{50}^m & X_1^v & X_2^v \dots X_{50}^v & Y_1^v & Y_2^v \dots Y_{50}^v & Z_1^v & Z_2^v \dots Z_{50}^v \end{bmatrix}$$

With the superscript of the vector's element that indicates the mean m or the variance v and the subscript that indicates the bin.

9. Experiments and Results

In order to know the anomalous states of the data and the way in which they present themselves, it is necessary to define their characteristics.
The machine on which a sensor is mounted may tilt. This would lead the sensor to record noise or bias that can be equivalent to:

- The addition of white noise tuned with feasible mean and standard deviation to nominal data.
  Figure 21 depicts a Gaussian white noise X with mean $\mu = 0$ and standard deviation $\sigma = 0.2$ in both time and frequency domain ($X \sim \mathcal{N}(\mu, \sigma^2)$).
  White noise in the frequency domain occupies the whole spectrum band and will therefore affect the signal in all frequencies.

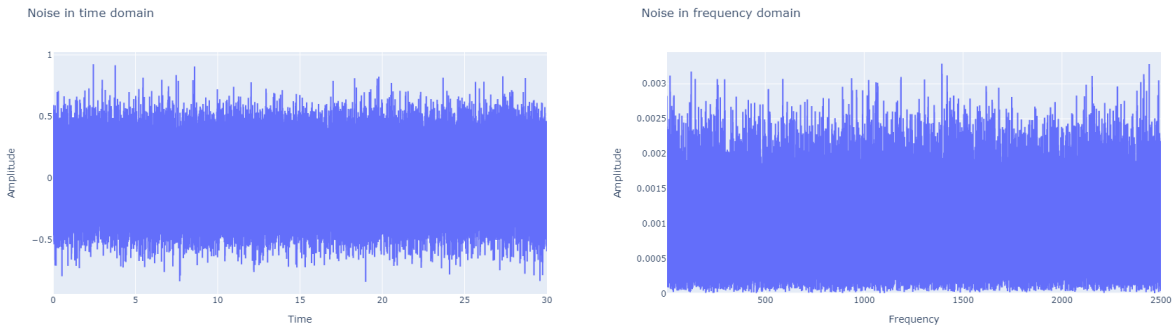

Figure 21: White noise in time and frequency domain

- The addition of sinusoids with appropriate magnitudes to nominal data.
  Figure 22 depicts a series of sinusoids in both time and frequency domain, in the context of sampling at 5000 Hz for 30 seconds. They have frequency between 1200 Hz and 1201 Hz, and magnitude of 0.66.
  The sinusoids in the frequency domain occupy only a limited band of the spectrum, which corresponds to their frequency and will therefore affect the signal with unusual peaks.

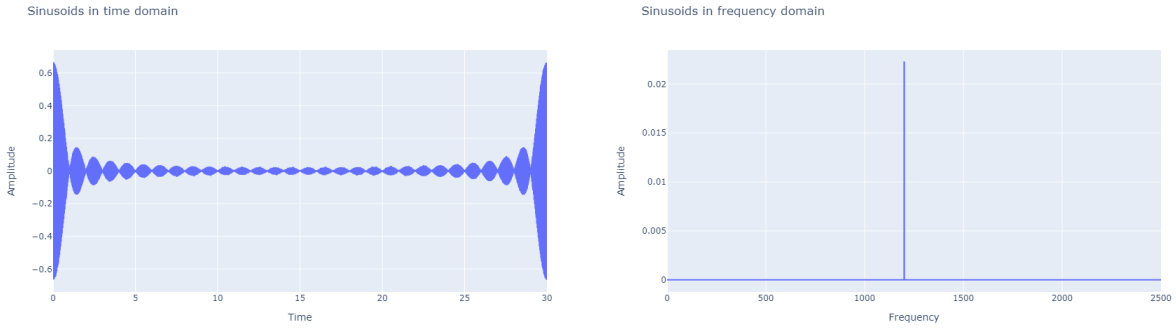

Figure 22: Sine waves in time and frequency domain

The production of the synthetic failures is based on the types of anomalies written before. These introductions applied to nominal data in the time domain on each axis (X,Y,Z) lead to undesired effects in the frequency domain.
With respect to the sensors data available, the unsupervised model was trained with a portion of Sensor_1's data. The model was then tested with all Sensor_2's data that were classified as anomalous. Thus sensors are not homogeneous between them and there is a need to have a model for each sensor.

9.1. Unsupervised model

The unsupervised one-class SVM model relative to Sensor_1 has been trained with 240 nominal samples of Sensor_1. The model has been tested on the remaining 400 samples (of Sensor_1) of which 200 were not modified and 200 were modified at each different configuration with the addition on each axis (X,Y,Z) of:

(a) White noise with mean $\mu = 0$ and standard deviation $\sigma = 0.16$, Figure 23.

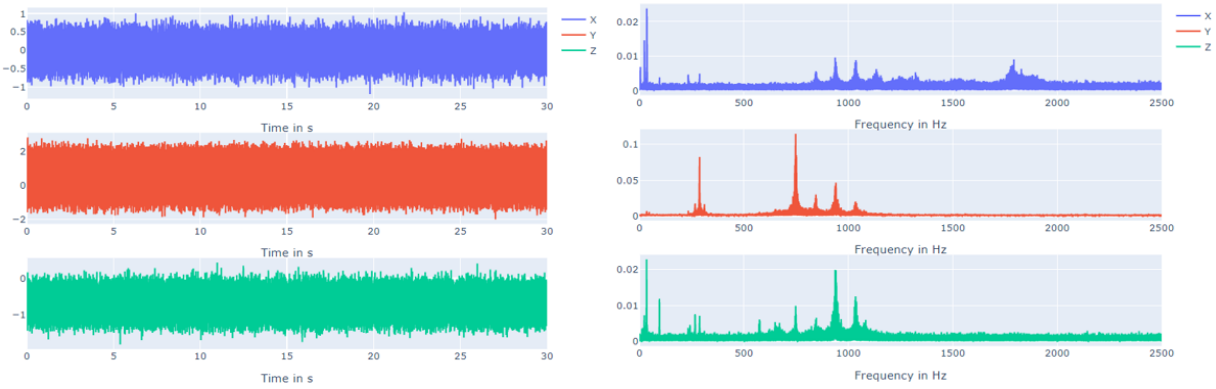

Figure 23: Sample data with white noise $X \sim \mathcal{N}(0,\, 0.0256)$ in time and frequency domain

(b) White noise with mean $\mu = 0$ and standard deviation $\sigma = 0.3$, Figure 24.

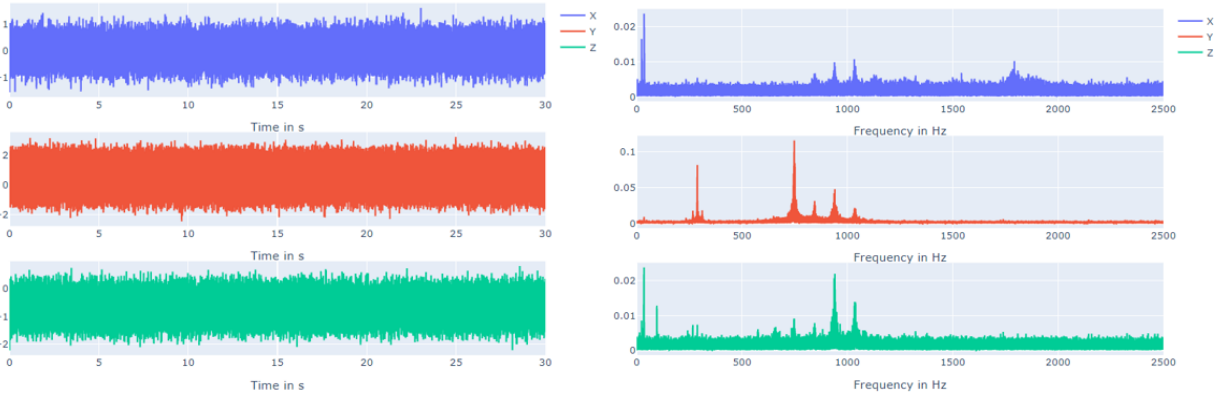

Figure 24: Sample data with white noise $X \sim \mathcal{N}(0,\, 0.09)$ in time and frequency domain

(c) Sinusoids with frequency $1200Hz \leq f \leq 1202Hz$ and magnitude equals to the max of each axis, Figure 25.

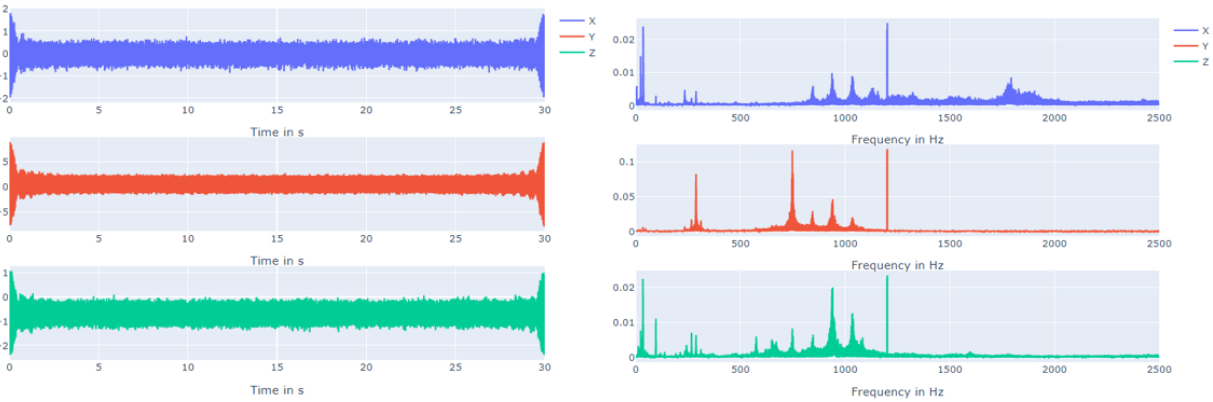

Figure 25: Sample data with sinusoids $(1200Hz \leq f \leq 1202Hz)$ in time and frequency domain

(d) Sinusoids with frequency $1200Hz \leq f \leq 1203Hz$ and magnitude equals to the max of each axis, Figure 26.

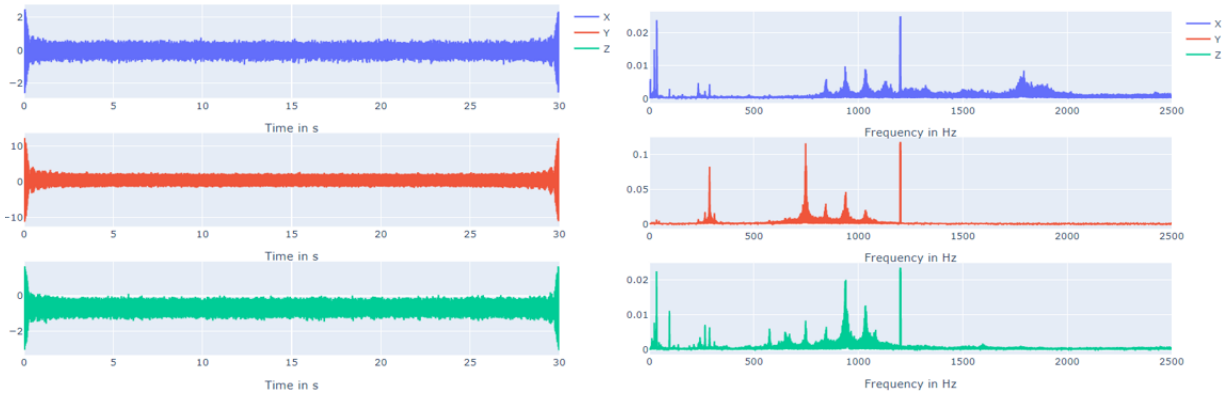

Figure 26: Sample data with sinusoids ($1200Hz \leq f \leq 1203Hz$) in time and frequency domain

(e) Sinusoids with frequency $600Hz \leq f \leq 601Hz$ and magnitude equals to the max of each axis, Figure 27.

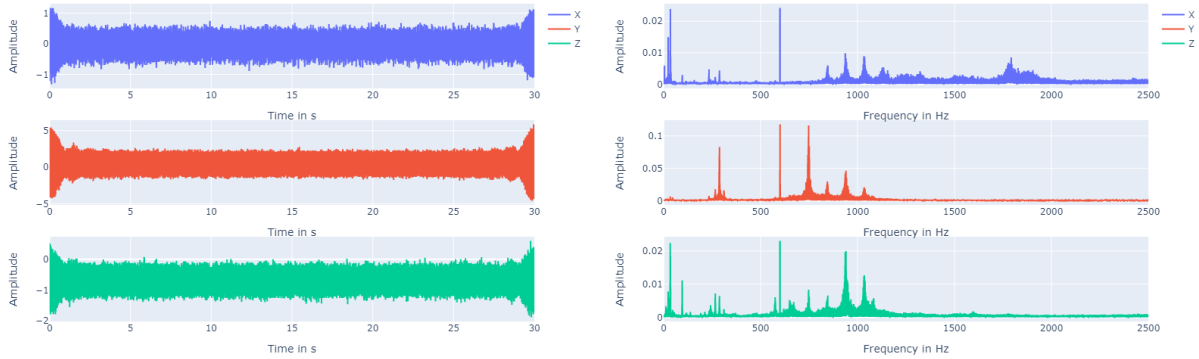

Figure 27: Sample data with sinusoids ($600Hz \leq f \leq 601Hz$) in time and frequency domain

The training phase and the test phase with all configurations, were repeated 5 times in order to have 5 different random partitions of the data. The accuracy results obtained by the unsupervised model are shown in the Table 1, divided by type of synthetic failure injected. Mean and variance refer to mean and variance of accuracies.

	Partition1	**Partition2**	**Partition3**	**Partition4**	**Partition5**	**Mean**	**Variance**
(a)	0.615	0.6	0.6075	0.975	0.9025	0.74	0.0268825
(b)	1.0	0.995	0.9825	0.9925	0.9925	0.9925	3.2499e-05
(c)	0.6	0.6075	0.575	0.5775	0.685	0.609	0.0016015
(d)	0.9475	0.94	0.975	0.965	0.9575	0.9570	0.000154
(e)	0.4925	0.495	0.505	0.5125	0.5175	0.5045	9.3499e-05

Table 1: Accuracies of unsupervised model

The results of the experiment show that the unsupervised model is able to correctly classify data affected by a relevant white noise. With less evident white noise (values extracted from a normal distribution with lower variance), the accuracy of the model is reduced depending on the partition.
As for the data with the addition of sinusoids: the model is able to correctly classify the sinusoids that have peaks in the frequency domain that occupy more than one frequency. On the other hand the peaks that occupy a single frequency are not classified as anomalous by the model. The model therefore does not assess the latter case with a significant weight. In fact the accuracy of the model is higher the more peaks occupy a large frequency band, denoting a unconventional intensity event.

9.2. Supervised model

In the context of supervised approach, the Sensor$_1$'s dataset (640 samples) was split in half randomly. One half (320 samples) left as it was. The other half have been added:

(a) White noise with mean $\mu = 0$ and standard deviation $\sigma = 0.08$, Figure 28.

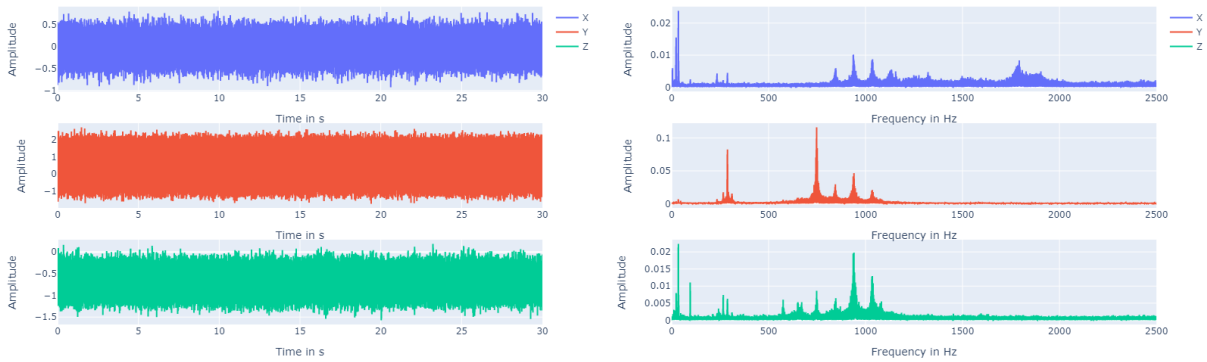

Figure 28: Sample data with white noise $X \sim \mathcal{N}(0, 0.0064)$ in time and frequency domain

(b) Sinusoids with frequency $1200Hz \leq f \leq 1202Hz$ and magnitude equals to the max of each axis, Figure 25.

(c) Mix of sinusoids with frequency $600Hz \leq f \leq 601Hz$ and magnitude equals to the max of each axis (Figure 27), and white noise with mean $\mu = 0$ and standard deviation $\sigma = 0.06$.

(d) Sinusoids with random frequencies and magnitude equals to the max of each axis.

The supervised KNN model relative to Sensor$_1$ has been trained with 240 samples of Sensor$_1$. The model has been tested on the remaining 400 samples.

The split of the dataset, the training phase and the test phase with all configurations, were repeated 5 times in order to have 5 different random partitions of the data. The accuracy results obtained by the supervised model are shown in the Table 2, divided by type of synthetic failure injected. Mean and variance refer to mean and variance of accuracies.

	Partition1	**Partition2**	**Partition3**	**Partition4**	**Partition5**	**Mean**	**Variance**
(a)	0.95	0.945	0.945	0.9475	0.9475	0.94699	3.50e-06
(b)	0.99	0.9975	0.995	0.9925	0.9575	0.9865	0.000216
(c)	0.8625	0.925	0.9175	0.9025	0.8775	0.89699	0.000561
(d)	0.4625	0.5125	0.5225	0.52	0.4925	0.502	0.0005009

Table 2: Accuracies of supervised model

The results of the experiment show that the supervised model is able to correctly classify data affected by both relevant and less evident white noise. And the accuracy of the model remains high.

As for the data with the addition of sinusoids: the model is able to correctly classify the sinusoids that have consecutive peaks in the frequency domain. On the other hand sinusoids with peaks that occupy random frequencies are not classified as anomalous by the model. The model therefore does not assess the latter case with a significant weight.

The supervised model was also trained and tested with three types of input data (configuration (c) with nominal data, anomalous sinusoids and anomalous white noise) and then a multi-class classification (not binary classification). The resulting accuracy of the model is still high.

In the scenario in which the case study is carried out, the supervised approach tends to be too limited. This is because the supervised model would not be able to correctly classify types of malfunctions that are different from those with which the model has been trained.

If real anomalous data had been available, such that a machine has some degradation over time, then it might be appropriate to use a supervised model trained on that data.

On the contrary, the unsupervised approach despite the results obtained during the experiments carried out, would seem to be a practical solution that adapts very well to the real need of the company.

10. Conclusions

The thesis project carried out aimed at the design and construction of a client-server architecture for the Anomaly detection as-a-Service via RESTful APIs. This solution is capable of supporting predictive maintenance activities of Eisenmann Italia S.r.l. in Industry 4.0 context. The developed architecture has proven to be able to provide the three services mentioned above. Moreover it classifies and detects anomalies, thus enabling diagnostic functions that reduce losses and total machines failure.
In addition, the solution is scalable across multiple sensors and consequently across multiple machines. In the event that the machine on which the sensor is mounted undergoes unnatural movements such as involuntary blows, it will be necessary to collect new data and then re-train the model. This is because otherwise the model would evaluate the current state of the machine as anomalous.
The results obtained by unsupervised model, based on the one-class SVM algorithm, are confident for use in industrial processes in the scenario in which Eisenmann Italia S.r.l. is located. On the contrary the use of supervised model, based on the KNN algorithm, does not guarantee reliability on uncharted anomalous states where the machines could be found.
In the end, the Anomaly detection as-a-Service is a promising and growing area of research. And the developed client-server architecture can start future improvements and applications in different areas of work.

References

[1] Peter Mell and Timothy Grance. The nist definition of cloud computing, 2011-09-28 2011.

[2] Yong Haur Tay. Xylorix: An ai-as-a-service platform for wood identification, 05 2019.

[3] Uma Subbiah, Muthu Ramachandran, and Zaigham Mahmood. Software engineering approach to bug prediction models using machine learning as a service (mlaas). In *ICSOFT*, 2018.

[4] Richard Milton and Flora Roumpani. Accelerating urban modelling algorithms with artificial intelligence. In *Proceedings of the 5th International Conference on Geographical Information Systems Theory, Applications and Management - Volume 1: GISTAM,*, pages 105–116. INSTICC, SciTePress, 2019. ISBN 978-989-758-371-1. doi: 10.5220/0007727201050116.

[5] Smitha Rajagopal, Katiganere Siddaramappa Hareesha, and Poornima Panduranga Kundapur. Performance analysis of binary and multiclass models using azure machine learning. *International Journal of Electrical and Computer Engineering*, 10(1):978–986, January 2020. ISSN 2088-8708. doi: 10.11591/ijece.v10i1.pp978-986.

[6] Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. In *Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm*, 09 2012.

[7] Hoon Sohn and Charles R Farrar. Damage diagnosis using time series analysis of vibration signals. *Smart Materials and Structures*, 10(3):446–451, jun 2001. doi: 10.1088/0964-1726/10/3/304. URL `https://doi.org/10.1088/0964-1726/10/3/304`.

[8] Ville Hautamaki, Ismo Karkkainen, and Pasi Franti. Outlier detection using k-nearest neighbour graph. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03*, ICPR '04, page 430–433, USA, 2004. IEEE Computer Society. ISBN 0769521282.

[9] Guangyuan Chen, Guoliang Lu, Jie Liu, and Peng Yan. An integrated framework for statistical change detection in running status of industrial machinery under transient conditions. *ISA Transactions*, 94: 294–306, 2019. ISSN 0019-0578. doi: https://doi.org/10.1016/j.isatra.2019.03.026. URL `https://www.sciencedirect.com/science/article/pii/S0019057819301569`.

[10] Diego Fernández-Francos, David Martínez-Rego, Oscar Fontenla-Romero, and Amparo Alonso-Betanzos. Automatic bearing fault diagnosis based on one-class ν-svm. *Computers I& Industrial Engineering*, 64 (1):357–365, 2013. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2012.10.013. URL `https://www.sciencedirect.com/science/article/pii/S036083521200277X`.

[11] Gustavo Scalabrini Sampaio, Arnaldo Rabello de Aguiar Vallim Filho, Leilton Santos da Silva, and Leandro Augusto da Silva. Prediction of motor failure time using an artificial neural network. *Sensors*, 19(19), 2019. ISSN 1424-8220. doi: 10.3390/s19194342. URL `https://www.mdpi.com/1424-8220/19/19/4342`.

[12] Cameron Sobie, Carina Freitas, and Mike Nicolai. Simulation-driven machine learning: Bearing fault classification. *Mechanical Systems and Signal Processing*, 99:403–419, 2018. ISSN 0888-3270. doi: https://doi.org/10.1016/j.ymssp.2017.06.025. URL `https://www.sciencedirect.com/science/article/pii/S0888327017303357`.

[13] Tomas Zimnickas, Jonas Vanagas, Karolis Dambrauskas, and Artūras Kalvaitis. A technique for frequency converter-fed asynchronous motor vibration monitoring and fault classification, applying continuous wavelet transform and convolutional neural networks. *Energies*, 13(14), 2020. ISSN 1996-1073. doi: 10.3390/en13143690. URL `https://www.mdpi.com/1996-1073/13/14/3690`.

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[15] Dan Li, Dacheng Chen, Jonathan Goh, and See kiong Ng. Anomaly detection with generative adversarial networks for multivariate time series, 2019.

[16] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), July 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL `https://doi.org/10.1145/1541880.1541882`.

[17] Mohammad Braei and Sebastian Wagner. Anomaly detection in univariate time-series: A survey on the state-of-the-art. *CoRR*, abs/2004.00433, 2020. URL https://arxiv.org/abs/2004.00433.

[18] Markos Markou and Sameer Singh. Novelty detection: A review—part 1: Statistical approaches. *Signal Process.*, 83(12):2481–2497, dec 2003. ISSN 0165-1684. doi: 10.1016/j.sigpro.2003.07.018. URL https://doi.org/10.1016/j.sigpro.2003.07.018.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[20] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLOS ONE*, 11(4):1–31, 04 2016. doi: 10.1371/journal.pone.0152173. URL https://doi.org/10.1371/journal.pone.0152173.

[21] Charu C. Aggarwal. *Outlier Analysis*. Springer Publishing Company, Incorporated, 2nd edition, 2016. ISBN 3319475770.

[22] Giuseppe Fenza, Mariacristina Gallo, and Vincenzo Loia. Drift-aware methodology for anomaly detection in smart grid. *IEEE Access*, 7:9645–9657, 2019.

[23] Mennatallah Amer and Markus Goldstein. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *Nearest-Neighbor and Clustering based Anomaly Detection Algorithms for RapidMiner*, 08 2012. doi: 10.5455/ijavms.141.

[24] Mennatallah Amer, Markus Goldstein, and Slim Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Enhancing one-class Support Vector Machines for unsupervised anomaly detection*, pages 8–15, 08 2013. doi: 10.1145/2500853.2500857.

[25] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 37–49, Bellevue, Washington, USA, 02 Jul 2012. PMLR. URL https://proceedings.mlr.press/v27/baldi12a.html.

[26] Roland Priemer. *SIGNALS AND SIGNAL PROCESSING*, pages 1–9. World Scientific Publishing Co Pte Ltd, 1990. doi: 10.1142/9789814434409_0001. URL https://www.worldscientific.com/doi/abs/10.1142/9789814434409_0001.

[27] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi: 10.25080/Majora-92bf1922-00a.

[28] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[29] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[30] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

Abstract in lingua italiana

Il Machine Learning as-a-Service (MLaaS) ha guadagnato grande popolarità negli ultimi anni. Si riferisce alla vasta gamma di strumenti di Machine Learning (ML) offerti come servizi da fornitori di cloud computing. Fondamentalmente, è l'uso della tecnologia per addestrare, testare e distribuire automaticamente i modelli di Machine Learning per un utente, in genere sfruttando una piattaforma AI con server farm offsite (il "cloud") per eseguirli per conto del client. L'obiettivo del MLaaS è quello di rendere più semplice e conveniente per le aziende l'utilizzo del ML, in modo che possano ottenere informazioni migliori dai loro dati più velocemente che mai. Il MLaaS viene utilizzato per una vasta gamma di casi d'uso come l'elaborazione del linguaggio naturale, la previsione, la regressione, il riconoscimento delle immagini ecc. Tuttavia, nella letteratura del Machine Learning as-a-Service è stata posta pochissima attenzione ai dati raccolti dai sensori in ambienti industriali. Da questo punto di vista, la progettazione e lo sviluppo di uno strumento di Machine Learning as-a-Service per l'identificazione delle anomalie nell'ambiente industriale è un'area di ricerca innovativa e promettente. Questa tesi mira a costruire un'innovativa architettura client-server che permetta di offrire Anomaly detection as-a-Service, basata su algoritmi di Machine Learning supervisionati e non-supervisionati, con tre diversi tipi di servizi: l'addestramento di un modello ML, l'aggiornamento di un modello ML e l'inferenza di dati attraverso un modello ML. In questa prospettiva viene trattato un vero e proprio caso studio in cui l'azienda Eisenmann Italia S.r.l. intende realizzare un sistema di monitoraggio per l'individuazione delle anomalie nel contesto della manutenzione predittiva. La manutenzione predittiva consente di anticipare il verificarsi di determinati eventi sul funzionamento dei macchinari industriali, migliorando la produttività, prolungando il ciclo di vita degli asset e riducendo i costi di riparazione e la complessità. Nello scenario industriale di Eisenmann Italia S.r.l., una serie di sensori triassiali montati su applicazioni per il trattamento dell'aria, bruciatori ecc., effettuano una raccolta periodica dei dati. Successivamente, questi dati sono stati manipolati e utilizzati per addestrare modelli ML supervisionati e non-supervisionati, consentendo di classificare e prevedere stati anomali di diverse macchine. Inoltre, poiché non erano disponibili dati in cui le macchine si trovavano in condizioni anomale, sono stati introdotti dati sintetici, costruiti a partire da dati nominali. I risultati ottenuti da entrambi gli approcci sono notevoli. In particolare, quelli ottenuti dall'approccio non supervisionato, basato sull'algoritmo One-class Support Vector Machine (One-class SVM) danno speranza per un uso pratico dell'intera architettura client-server per Anomaly detection as-a-Service nei processi industriali.

Parole chiave: Machine Learning as-a-Service, Machine Learning, Anomaly detection as-a-Service, Anomaly detection, Architettura Client-Server, Modello Non-Supervisionato