Dinis Costa Cabanas

BSc in Computer Science

Wind Turbine Fault Detection: an Unsupervised vs Semi-Supervised Approach

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Informatics Engineering

Adviser: Susana Nascimento, Assistant Professor, Universidade NOVA de Lisboa

Co-adviser: Tiago Silva, Assistant Professor, Universidade NOVA de Lisboa

Examination Committee

Chairperson: Professor Jorge Cruz

Raporteur: Professor Antonio J. Tallón Ballesteros

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Fevereiro, 2021

Wind Turbine Fault Detection: an Unsupervised vs Semi-Supervised Approach

This document was created using the (pdf)LaTeX processor, based in the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

In memory of Maria Antónia,
Joaquim Cabanas,
Maria Júlia,
and Joaquim Costa

Acknowledgements

My special acknowledgment goes to my advisor, Professor Susana Nascimento, for all the things she taught me, and the incredible perseverance, professionalism and patience demonstrated during this entire dissertation.

I would like to thank my family and friends, specially André Neves who shed me light when I had doubts.

Abstract

The need for renewable energy has been growing in recent years for the reasons we all know, wind power is no exception. Wind turbines are complex and expensive structures and the need for maintenance exists. Conditioning Monitoring Systems that make use of supervised machine learning techniques have been recently studied and the results are quite promising. Though, such systems still require the physical presence of professionals but with the advantage of gaining insight of the operating state of the machine in use, to decide upon maintenance interventions beforehand. The wind turbine failure is not an abrupt process but a gradual one.

The main goal of this dissertation is: to compare semi-supervised methods to attack the problem of automatic recognition of anomalies in wind turbines; to develop an approach combining the Mahalanobis Taguchi System (MTS) with two popular fuzzy partitional clustering algorithms like the fuzzy c-means and archetypal analysis, for the purpose of anomaly detection; and finally to develop an experimental protocol to comparatively study the two types of algorithms.

In this work, the algorithms Local Outlier Factor (LOF), Connectivity-based Outlier Factor (COF), Cluster-based Local Outlier Factor (CBLOF), Histogram-based Outlier Score (HBOS), k-nearest-neighbours (k-NN), Subspace Outlier Detection (SOD), Fuzzy c-means (FCM), Archetypal Analysis (AA) and Local Minimum Spanning Tree (LoMST) were explored.

The data used consisted of SCADA data sets regarding turbine sensorial data, 8 total, from a wind farm in the North of Portugal. Each data set comprises between 1070 and 1096 data cases and characterized by 5 features, for the years 2011, 2012 and 2013. The analysis of the results using 7 different validity measures show that, the CBLOF algorithm got the best results in the semi-supervised approach while LoMST won in the unsupervised scenario. The extension of both FCM and AA got promissing results.

Keywords: outlier detection; wind turbine fault detection; semi-supervised methods; fuzzy clustering; evaluation measures

Resumo

A necessidade de produzir energia renovável tem vindo a crescer nos últimos anos pelas razões que todos sabemos, a energia eólica não é excepção. As turbinas eólicas são estruturas complexas e caras e a necessidade de manutenção existe. Sistemas de Condição Monitorizada utilizando técnicas de aprendizagem supervisionada têm vindo a ser estudados recentemente e os resultados são bastante promissores. No entanto, estes sistemas ainda exigem a presença física de profissionais, mas com a vantagem de obter informações sobre o estado operacional da máquina em uso, para decidir sobre intervenções de manutenção antemão.

O principal objetivo desta dissertação é: comparar métodos semi-supervisionados para atacar o problema de reconhecimento automático de anomalias em turbinas eólicas; desenvolver um método que combina o Mahalanobis Taguchi System (MTS) com dois métodos de agrupamento difuso bem conhecidos como fuzzy *c*-means e archetypal analysis, no âmbito de deteção de anomalias; e finalmente desenvolver um protocolo experimental onde é possível o estudo comparativo entre os dois diferentes tipos de algoritmos.

Neste trabalho, os algoritmos Local Outlier Factor (LOF), Connectivity-based Outlier Factor (COF), Cluster-based Local Outlier Factor (CBLOF), Histogram-based Outlier Score (HBOS), *k*-nearest-neighbours (*k*-NN), Subspace Outlier Detection (SOD), Fuzzy *c*-means (FCM), Archetypal Analysis (AA) and Local Minimum Spanning Tree (LoMST) foram explorados.

Os conjuntos de dados utilizados provêm do sistema SCADA, referentes a dados sensoriais de turbinas, 8 no total, com origem num parque eólico no Norte de Portugal. Cada um está compreendendido entre 1070 e 1096 observações e caracterizados por 5 características, para os anos 2011, 2012 e 2013. A ánalise dos resultados através de 7 métricas de validação diferentes mostraram que, o algoritmo CBLOF obteve os melhores resultados na abordagem semi-supervisionada enquanto que o LoMST ganhou na abordagem não supervisionada. A extensão do FCM e do AA originou resultados promissores.

Palavras-chave: deteção de outliers; deteção de falhas em turbinas eólicas; métodos semi-supervisionados; agrupamento difuso; medidas de avaliação

Contents

List of Figures

List of Tables

Acronyms

k-NN	*k*-Nearest Neighbours.
AA	Archetypal Analysis.
AA-MMTS	Archetypal Analysis Modified Mahalanobis Taguchi System.
AC	alternating current.
ACA	Ant Clustering Algorithm.
AD	Anomaly Detection.
ADA	Archetypoid Analysis.
AO	Alternating Optimization.
AP	Anomalous Pattern.
APR	Average Precision.
AUC	Area Under Curve.
BMU	Best Match Unit.
BNEF	Bloomberg New Energy Finance.
CBLOF	Cluster-Based Local Outlier Factor.
CCOD	Cooperative Clustering Outlier Detection.
CMI	Conditional Mutual Information.
CMIM	Conditional Mutual Info Maximisation.
CMS	Condition Monitoring Systems.
CNN	convolutional neural network.
COF	Connectivity-Based Outlier Factor.
CP	contribution proportion.

CR	Classification Rate.
CS	Crisp Silhouette.
CVI	Clustering Validation Indices.
DISR	Double Input Symmetrical Relevance.
DR	Detection Rate.
DRAMA	Dimensionality Reduction Anomaly Meta-Algorithm.
EM	Expectation–Maximization.
EWEA	European Wind Energy Association.
F1	F1-score.
FAR	False Alarm Rate.
FCM	Fuzzy *c*-Means.
FCM-MMTS	Fuzzy *c*-Means Modified Mahalanobis Taguchi System.
FCPA	Functional Principal Components Analysis.
FCPM	Fuzzy Clustering via Proportional Membership.
FD	functional data.
FDA	Functional Data Analysis.
FF	FurthestFirst.
FindCBLOF	Find Cluster-Based Local Outlier Factor.
FN	False Negative.
FNR	False Negative Rate.
FOADA	Functional Outliers Archetypoid Analysis.
FP	False Positive.
FPR	False Positive Rate.
FS-AA	FurthestSum-Archetypal Analysis.
FSI	Fuzzy Silhouette Index.
GMM	Gaussian Mixture models.
GWEC	Global Wind Energy Council.

HBOS	Histogram-Based Outlier Detection.
HDFS	Hadoop Data File System.
ICAP	Interaction Capping.
IDS	Intrusion Detection System.
iForest	Isolation Forest.
IRENA	International Renewable Energy Agency.
JMI	Joint Mutual Information.
KCCA	Kernel canonical correlation coefficient.
KNN-NAD	*k*-NN Normalized Average Density.
KNN-LPE	*k*-NN Localized p-value Estimator.
LCOE	levelised cost of electricity.
LOF	Local Outlier Factor.
LOFN	LOF Normalized.
LoMST	Local Minimum Spanning Tree.
LRD	Local Reachability Density.
LSTM	long short-term memory.
MAE	mean absolute error.
MAP	Mean Average Precision.
MAPE	mean absolute percentage error.
MD	Mahalanobis Distance.
MDWD	multilevel discrete wavelet decomposition.
MIFS	Mutual Information Feature Selection.
ML	machine learning.
MMTS	Modified Mahalanobis Taguchi System.
MPC	Modified Partition Coefficient.

mRMR	Min-Redundancy Max-Relevance.
MST	Minimum Spanning Tree.
MTS	Mahalanobis Taguchi System.
NMF	Nonnegative Matrix Factorization.
NN	neural networks.
O&M	Operation & Maintenance.
ODSS	Outlier Detection DataSets.
PACE	Principal Components Analysis through Conditional Expectation.
PC	Partition Coefficient.
PCA	Principal Component Analysis.
PE	Partition Entropy.
PR	Precision-Recall.
QO	search-based quasi-optimal algorithm.
RBF-SVM	radial basis function support vector machines.
REN21	Renewable Energy Policy Network for the 21st Century.
ROC	Receiver Operating Characteristic.
RWS	Rank-Weighted Score.
SCADA	Supervisory Control and Data Acquisition.
SI	Separation Index.
SOD	Subspace Outlier Detection.
SOM	Self-organizing map.
SVM	support vector machines.
TN	True Negative.
TP	True Positive.

TPR True Positive Rate.

XB Xie-Beni.

Chapter 1

Introduction

1.1 Context and Motivation

Wind power technologies transform the kinetic energy of the wind into useful mechanical power. Wind turbines are generally categorised by whether they are horizontal axis or vertical axis, and whether they are located onshore or offshore. Power generation of wind turbines is determined by the capacity (active power) of the turbine (in kW or MW), the wind speed, the height of the turbine and the diameter of its rotors.

The first wind turbines typically had small capacities (10 kW to 30 kW) by today's standards but pioneered the development of the modern wind power industry that we see today. The wind power industry has experienced an average growth rate of 27% per year between 2000 and 2011, and wind power capacity has doubled on average every three years. The total wind power capacity at the end of 2011 was 20% higher than at the end of 2010 and reached 238 GW by the end of 2011 (Figure 1.1).

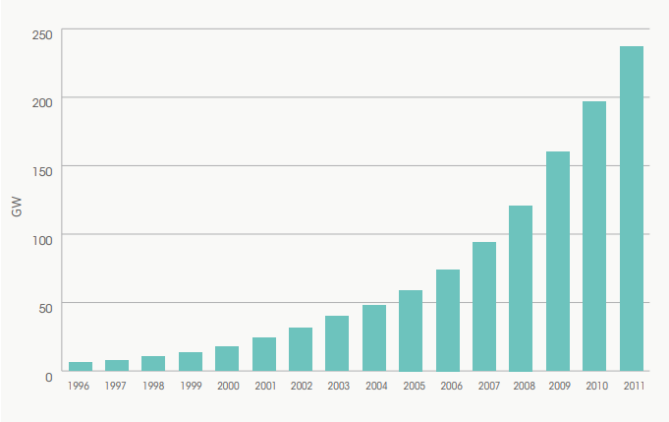

Figure 1.1: Global installed Wind Power capacity, 1996 to 2011.
Source: Global Wind Energy Council (GWEC) 2012, taken from[1]

Maintenance costs are an important point to take into account in renewable energies in general, and in wind electrical generation in particular (International Renewable Energy Agency (IRENA)[2], 2012). Current wind turbines are designed to work around 120 000 hours a year, over a 20-year lifetime. The capital costs of a wind power project can be broken down into the following major categories although we are only interested in the **first one**:

- The turbine cost: including blades, tower and transformer;
- Civil works: including construction costs for site preparation and the foundations for the towers;
- Other capital costs: these can include the construction of buildings, control systems, project consultancy costs, etc.

Such complex structures, like wind turbines, require good supervision and prevention of faults methods must be enforced in the wind farms they are inserted. These faults that occur are denoted as **anomalies**.

The Anomaly Detection (AD) field is important because anomalies in the data mean significant, and often critical,information in a wide variety of application domains such as, fraud detection for credit cards or insurances, intrusion detection for cyber-security, and military surveillance for enemy activities.

In [30] we can find a simple and concise definition of what an outlier is. An outlier/anomaly is a data object that deviates significantly from the rest of the objects, as if it was generated by a different mechanism. An outlier is not the same as noisy data. Noise is considered a random error or variance when we are measuring some variable. Noise is not desired in data analysis including outlier detection, therefore it should be removed before this process.

Due to their nature, anomalies impose many challenges, with special attention to the **fourth** one:

(i) The boundary between normal and anomalous behaviour is often not precise thus, an anomalous point that lies close to the boundary might be considered normal, and vice-versa;

(ii) In many domains normal behaviour might evolve in such a way that the notion of normal behaviour might not be sufficiently representative in the future;

(iii) In different application domains the notion of anomaly is not the same, that way, applying a technique developed in one domain to another is not straightforward;

[1] https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2012/RE_Technologies_Cost_Analysis-WIND_POWER.pdf

[2] https://www.irena.org/aboutirena

(iv) Labeled data for training/validation of detection models is sometimes scarse and that imposes a major issue;

(v) Many times the data contains noise which tends to be similar to the anomalies and the difficulty to distinguish these two groups is greater.

These make the task of AD not an easy problem to solve, therefore, the best approach is to use a specific detection model to tackle a specific formulation of the problem.

Labeling the data as accurate as possible is expensive since it is often done manually by a human expert which requires substantial effort. Getting labels for anomalous behaviour is more difficult than getting labels for normal behaviour [16]. This is the major reason for the widely adoption of semi-supervised and unsupervised techniques in AD. Semi-supervised techniques only assume the training data to be labelled for the normal behaviour and do not require labels for the anomalous case. Unsupervised techniques do not require training data, hence no labelling needed at all and thus are the most widely applicable.

1.2 Problem

Because of the harsh working environment and a complex structure, wind turbines are prone to relatively high failure rates, leading to undesired Operation & Maintenance (O&M) costs. One of the main tasks of the O&M process is to find out the possible causes of a fault manifested by a specific alarm or a set of alarms that may stop the wind turbine production.

When it comes to wind turbines there are two ways to detect faults: data not respecting the correct behaviour of the turbine (anomalous values), or miss calibration of the alarms that are part of the Condition Monitoring Systems (CMS) leading to data impossible to analyze (missing values). Not dealing with these anomalies can lead to serious problems in the turbines and hence tremendous financial costs.

In recent years machine learning (ML) techniques, like automatic input selection algorithms, convolutional neural networks and support vector machines, have been gaining popularity as a valid solution for solving the AD problem in the wind turbine context. ML can be described using its formal definition:

A machine is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P** if its performance at tasks in **T**, as measured by **P**, improves with experience **E** [46].

In simpler terms, ML provides systems the ability to automatically learn and improve from experience without being explicitly programmed. The basic process of ML is to give training data to a learning algorithm. The learning algorithm then generates a new set of rules, based on inferences from the data. This is, in essence, generating a new algorithm,

formally referred to as the ML model. Two types of ML models will be the main focus of this work, semi-supervised and unsupervised. Unsupervised learning is essentially a synonym for clustering where the learning process is unsupervised since the input examples are not class labeled. Typically, clustering is used to discover classes within the data [30].

These techniques support the O&M process and hence reduce the downtime or detect critical faults in earlier stages. Methodologies and tools that can support this type of process can benefit wind farm owners not only to increase availability and production but also to reduce costs [9].

1.3 Main Contributions

This dissertation presents some contributions such as:

(i) To compare semi-supervised methods for anomaly detection to attack the problem of automatic recognition of anomalies on wind turbines;

(ii) To develop an approach combining the Mahalanobis Taguchi System (MTS) with two popular fuzzy partitional clustering algorithms: the Fuzzy *c*-means (FCM) and Archetypal Analysis (AA), for the purpose of anomaly detection;

(iii) To develop an experimental protocol to comparatively study the two types of algorithms (i) and (ii) with data from a wind farm located in the North of Portugal, comprising eight wind turbines, for the years of 2011-2013.

1.4 Organization

The remaining of the dissertation is organized as follows, Chapter 2 is dedicated to present some brief theoretical details about the different components of a wind turbine. It also dedicated a Section to introduce the Supervisory Control and Data Acquisition (SCADA) system, a really important system used in many wind farms nowadays to monitor and gather sensorial data from the turbines. Later in this Chapter, a survey is done regarding AD in wind turbines.

In Chapter 3 theoretical details about the different types of anomalies is presented, as well as the categorization of AD algorithms and their respective advantages and disadvantages. The rest of this Chapter is about introducing and describing the algorithms studied in this dissertation, and also the novel approach.

Chapter 4 will focus on explaining the SCADA variables used for study, the exploratory data analysis performed, the preprocessing done, and finally the datasets used in the experiments.

Chapter 5 explains in a detailed manner the experimental protocol of this dissertation. It is also explained the normalization process of the datasets, the type of working conducted for the semi-supervised techniques and for the unsupervised techniques, and the assessment measures used to evaluate the performance of the algorithms.

Finally, Chapter 6 presents the conclusions and some proposals of future work.

CHAPTER 2

WIND TURBINE ANOMALY DETECTION

In this Chapter a brief contextual background of some wind turbines components is presented in Section 2.1. In Section 2.2 the SCADA system is introduced, an important core of wind turbines related topics. And finally in Section 2.3, a survey is done on works related to AD problems in wind turbines.

2.1 Wind Turbine Components

The principal components [8] of a wind turbine are: the rotor that includes the blades, the hub and the main shaft, the gearbox, the generator and the nacelle that contains all of these mechanical components. There is also the tower that holds and supports all the components and gives access to the nacelle. Figure 2.1 illustrates all the components that constitute a wind turbine.

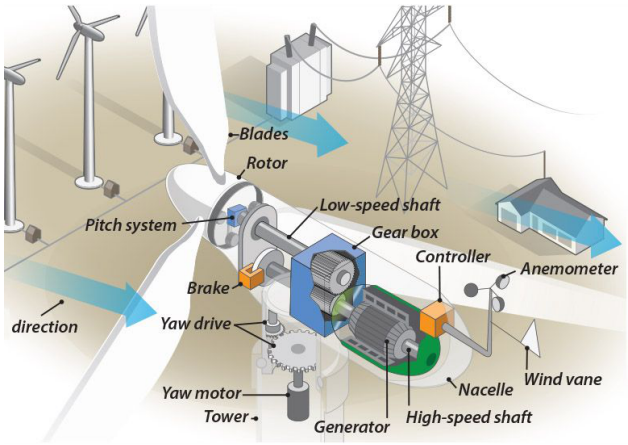

Figure 2.1: Components of a wind turbine
Source: taken from[1]

Rotor

The rotor transforms the wind energy in mechanical energy through the conversion of the wind aerodynamic energy in rotation. As mentioned above the rotor consists of, the blades responsible for harnessing the wind, the hub that supports the blades and the main shaft that connects the blades to the gearbox low-speed shaft.

Blades

The blades and the hub (which together constitute the rotor) are mounted on the nacelle through suitable bearings. The blades are the components that interact with the wind and are designed with an airfoil to maximize aerodynamic efficiency. It is important to design the side of the blade close to the center (hub), so there is good support and low air resistance.

Hub

The wind turbine hub is the component that connects the blades and the main shaft. It also includes the *Pitch* systems (pitch angle regulation systems). The hubs are usually made of molten iron and are protected externally by an oval component named (*Spinner*).

Gearbox

The majority of the wind turbines have gearboxes with one or more stages between the rotor, responsible for extracting kinetic energy of the wind and converting it in mechanical rotation energy, and the generator that converts the mechanical energy into available electrical energy. The gearbox main purpose is to increase the velocity of rotation received from the rotor so it can adapt the energy generated by the generator to the frequency of the power distribution grid. This component is a source of noise and vibration, and the one which requires the most maintenance.

Brakes

Almost every wind turbine makes use of mechanical brakes assembled in the transmission system, furthermore an aerodynamical brake. In many cases, mechanical brakes can stop the rotor during unfavourable weather conditions or when there is the necessity of some sort of maintenance intervention. Besides that, mechanical brakes can also be used to ensure that the rotor is stopped when the turbine is not working. Mechanical brakes can be located both next to the low-speed shaft or the high-speed shaft.

[1] https://www.energy.gov/eere/wind/inside-wind-turbine

Generator

Asynchronous generator

An induction generator or asynchronous generator is a type of electric current generator alternating current (AC) that uses the principles of induction motors to produce energy. The asynchronous generators operate at higher rotor speeds than synchronous speed generators. When a gust of wind hits a wind turbine equipped with an asynchronous generator rotor under short circuit, since the speed of rotation is constant, there is a sudden torque variation and the consequent rapid variation in output power.

Synchronous generator

In this type of generator, also called an alternator, the rotor consists of an electromagnet direct current or permanent magnets. The frequency of the induced voltage in the stator (and consequently the current generated) is directly proportional to the speed rotor rotation. Increase of rotation speed accumulates kinetic energy in the rotor itself and allows a constant power supply. On the other hand, when the wind decreases, the energy stored in the rotor is released while the rotor itself is slowing down.

Transformer

The output of electrical energy from generators is usually done at low voltage and should be transformed into medium voltage through a transformer, to reduce the transmission losses when connecting to the distribution network. The transformer is installed in the nacelle or at the base of the tower.

Regulation system of the nacelle (*yaw*)

The nacelle is built in such a way that it can rotate at the top of the tower through an active *yaw* system. This consists of electric actuators and gears so that the rotor is always aligned with the wind. Direction and velocity of the wind are continuously monitored by the sensors connected to the nacelle. The rotor is generally positioned according to the average wind direction, calculated over a 10-min period by the turbine control system.

2.2 Supervisory Control and Data Acquisition System

Nowadays all wind farms use the SCADA system to monitor and record their wind turbines behaviour in 10-min intervals [9]. The minimum data set typically includes 10 min-average values of wind speed, wind direction, active power, reactive power, ambient temperature, pitch angle and rotational speed (rotor and/or generator)[2]. In modern wind turbines, however, the SCADA data often comprises of hundreds of signals, including

[2] https://www.vgb.org/vgbmultimedia/383_Final+report-p-9786.pdf

temperature values from a variety of measurement positions in the turbine, pressure data, for example from the gearbox lubrication system, electrical quantities such as line currents and voltages or pitch-motor currents or tower vibration, amongst many others [29, 49, 57, 67]. Large-scale SCADA data sets often contain not only the 10-min or even 5-min averaged values, but also minimum, maximum and standard deviation values for each interval. Therefore, due to the large number of available variables and data, analyzing them can be a high time-consuming task [42] and when just well-known related variables are analyzed, hidden causes might not be found.

2.3 State of the art review

The authors in [43] made a thorough study of automatic input selection algorithms for wind turbine failure prediction. In this work, an exhaustive search-based quasi-optimal algorithm (QO), which has been used as a reference for the automatic algorithms, was proposed. This allowed to consider the whole set of variables of the subsystem and automatically select the smallest subset of relevant variables, which in turn will simplify the models and permit a graphical representation of their time evolution. The automatic algorithms used were: Mutual Information Feature Selection (MIFS), Conditional Mutual Information (CMI), Joint Mutual Information (JMI), Min-Redundancy Max-Relevance (mRMR), Double Input Symmetrical Relevance (DISR), Conditional Mutual Info Maximisation (CMIM) and Interaction Capping (ICAP). Using the Classification Rate (CR)[3] and F1-score (F1)[4], the authors concluded that the QO algorithm was the one that led the best CR and F1 for 6 features, while CMI automatic feature extraction algorithm lead to similar CR and F1 for 3 or 4 features. The case study was conducted for the following features: main bearing oil pressure, gearbox oil pressure, wind velocity, wind direction, active power and the temperature of the main gearbox bearing.

The authors in [70] proposed a method based on ML to predict long cycle maintenance time of wind turbines for efficient management in the power company. Long cycle maintenance time prediction makes the power company operate wind turbines as cost-effectively as possible to maximize the profit. To predict the long cycle maintenance time precisely the proposed method consists of a hybrid network which combines a two-layer convolutional neural network (CNN) and radial basis function support vector machines (RBF-SVM). CNNs are quite good at learning invariant features, but not always optimal for classification (most of the training data are in the middle layers). While RBF-SVM, with a fixed kernel function, cannot learn complicated invariances but can produce good decision surfaces when applied to well-behaved feature vectors. The *Apriori* algorithm [4] and linear regression were employed as preprocessing methods for feature selection. The authors concluded that the proposed method achieves better prediction results for the

[3] Calculated as the percentage of well-classified instances divided by the total number of instances
[4] Obtained as the harmonic mean of precision and recall

long cycle maintenance time of the wind turbine compared to the traditional CNN. The data understudy, collected by the SCADA system, corresponded to the following features: wind velocity, the power output of the wind turbines, the oil temperature of the wind turbine gearbox and the temperature of the high-speed bearing.

A two-stage methodology to predict failure within 1 to 2 months of occurrence was conducted in [66]. The study consisted of first using clustering techniques to produce subpopulation of data based on operating conditions. Secondly, classification is made on individual clusters as healthy or unhealthy from vibration-based CMS by applying order analysis[5] techniques to extract features. Two models are presented in this paper, both of which used classification algorithms to classify the bearing as healthy or unhealthy, with the first acting as a baseline model to compare the second two-stage cluster-classification model. The Fourier transform was adapted for a sliding time window, due to the generator shaft speed of the turbine can vary meaning that the signal is not stationary, by using the short-time Fourier transform, where a spectrotemporal representation of the signal is obtained. This is used for order analysis and allows the spectral values to be tracked in time. The authors chose the k-Means algorithm for the first stage of the process, and Decision Trees and support vector machines (SVM) with polynomial kernel for the second stage. Cross-validation was used to determine the overall accuracy of the algorithm, while the prediction process was evaluated by using a confusion matrix giving correct/incorrect classification and the likelihood of false positives/negatives. The authors concluded that the proposed model showed an overall high level of accuracy and that clustering is better than bin classification of the vibration samples, but too many clusters make data too sparse and accuracy falls.

In [21] the authors propose a method to detect anomalies in wind turbines, furthermore to find the sub-component responsible for such anomaly. This model is based on Self-organizing map (SOM) that aims to adopt the normal behaviour of a wind turbine by projecting high-dimensional SCADA data into a two-dimensional space. Afterwards, the Euclidean distance based indicator for system-level anomalies is defined and a filter is created to screen out suspicious data points based on quantile function. Parameter selection is a very important process for modelling the normal behaviour of a wind turbine. This process begins by using P-value[6] analysis for general relationship test and is followed by the Pearson correlation coefficient and Kernel canonical correlation coefficient (KCCA). The modelling of the normal behaviour of the turbine is done by analyzing the difference between the current status and normal behaviour, and it can be represented using the Euclidean distance between new input data and Best Match Unit (BMU)[7] in

[5] Order analysis is a resampling technique that can be effective when analysing nonstationary signals.

[6] Null Hypothesis refers to a general statement or default position that there is no relationship between two measured phenomena. After using P-value to filter unrelated parameters, correlation coefficient analysis is adopted for next step investigation.

[7] A BMU is a neuron whose weight vector has the smallest distance measure from the input data.

a two-dimensional space. The quantile function mentioned before is used to define a threshold to screen out the suspicious data. Whenever a data point falls outside this threshold, there's a high probability it is an anomaly. In order to go deeper and find the root component responsible for the anomaly a contribution proportion (CP) is added. The authors concluded that the proposed method is efficient at finding anomalies and the CP index is effective for figuring out which sub-component is responsible for the anomaly. The SCADA features used were: power output, wind's direction, power factor, nacelle temperature, outside temperature, reactive power and yaw.

The authors in [61] conducted quite an extensive review on ML models that have been used for condition monitoring in wind turbines. Most of the models use SCADA or simulated data, with almost two-thirds of methods using classification and the rest relying on regression. Neural networks (NN), SVM and Decision Trees are most commonly used. Although the training time of NNs can be potentially long, when it comes to actual classification or regression, the application of models is comparatively very fast. SVMs can be slow and training on large data sets remains a challenge. The time complexity is usually between $O(m^2n)$ and $O(m^3n)$, where m is the number of instances and n is the number of features. Feature selection and extraction is a very important process before building a ML model. Outlier identification, wrapper methods, embedded methods, filter methods, statistics[8], Hidden Markov were some of the methods covered by the authors. For validation, mean absolute error (MAE) and mean absolute percentage error (MAPE) were used and the authors concluded that deep NN are believed to achieve better performance in terms of accuracy and the model chosen depends on the task. Frameworks such as Hadoop Data File System (HDFS) are really good at handling and process huge data. The SCADA features used were: wind velocity, power output, generator speed, generator stator temperature.

In [71] a really specific problem in wind turbines is approached, blade icing detection. Ice accumulated on a blade will typically cause degradation for a turbine aerodynamic performance, and cause many other serious problems such as measurements errors. To solve this problem the authors propose a *WaveletAE*, a wavelet enhanced autoencoder model that contains a multilevel discrete wavelet decomposition (MDWD) model, a convolutional encoder, a multiple scale long short-term memory (LSTM) encoder-decoder, and a convolutional decoder. The proposed method can be briefly described in the following way: The input multivariate signals are first decomposed to multilevel wavelet detail coefficients. Then in each scale level, the original signals or wavelet detail coefficients go through a convolutional encoder and an LSTM encoder, the hidden states of the LSTM encoder will be concatenated to create a global code. In the decoding phase, fully connected layers will first map the global code to initial hidden states in each scale, then

[8]Mean, standard deviation, maximum, minimum, skewness, kurtosis, peak-to-peak, crest factor, wave factor, impulse factor, margin factor, root mean square [38]

an LSTM decoder and a convolutional decoder will reconstruct the original signals and the wavelet detail coefficients. The variables studied were, the running time (hours) of 3 wind turbines, a binary indicating whether the blades are frozen or not, range labels that indicate the segmentation is in an abnormal state. To validate the model the authors adopted the *accuracy*, *precision*, *recall* and F1 as validation metrics and concluded that, supervised AD may lead to biased results but the overall performance of *WaveletAE* has been verified for both supervised and semi-supervised learning. Simulated deployment case study with a shifting window demonstrates the robustness of the proposed method for real-time implementations.

Chapter 3

Anomaly Detection

In this Chapter it is presented in Section 3.1 the different types of anomalies, two classification modes used to classify AD techniques are explored in Section 3.2 as well as the advantages and disadvantages of the families of AD algorithms. In Sections 3.3 and 3.4 the semi-supervised and unsupervised algorithms explored and studied in this work are described, and in the latter it is also described the novel approach. At the end of this Section a review is done on AD works using clustering. Lastly in Section 3.5, are presented well established indices to validate partitions generated by fuzzy clustering approaches.

3.1 Types of Anomalies

The problem of AD has been the subject of many distinct works throughout the last decades. It is usually described as the problem of finding patterns in data that deviate from the expected behaviour. These non-conforming patterns are often called anomalies and can be categorized into three major groups [16]:

Point Anomalies

If an individual observation can be considered anomalous regarding the rest of the data. This is the simplest type of anomaly and the focus of the majority of research on AD. A real life example is the case of a credit card fraud detection, where the transaction for which the amount spent is very high compared to the normal range of expenses for a person, will be considered a point anomaly.

Contextual Anomalies

When a certain observation is anomalous in a specific context, where context is interpreted as the structure in the data set and needs to be specified as part of the problem formulation. Points in the data are defined using two sets of attributes: (1) contextual attributes that are used to determine the neighbourhood of the point; (2) behavioral attributes that define the non-contextual characteristics of a point.

In a certain context a data instance might be a contextual anomaly, but an identical data instance could be considered normal due to its behavioral attributes. Defining the context of a point is not always straightforward.

Collective Anomalies

In this type of anomalies, individual data instances may not be anomalies even if they belong to a collection of related observations that is considered anomalous as a whole. A good real life example for these type of anomalies is the following: let's considered the electrocardiogram of a patient, and for a large amount of time it shows a breaking rhythm of the patient's heart. The low value by itself is not an anomaly, but the entire selected region where that low value was detected denotes an anomaly.

3.2 Classification of Anomaly Detection Algorithms

3.2.1 Learning Mode *vs* Output

When it comes to classify AD algorithms according to the learning mode, there are three ways in which they can operate [16]:

- **Supervised:** In this mode of operation, the AD models make use of the labels available for training, using data sets with observations labelled as normal as well as anomalous. These techniques usually build a predictive model for normal *vs* anomaly classes, and any unseen data instance is compared against the model to determine which class it belongs to;
- **Semi-Supervised:** In some cases, is only available the labelling for the normal class. In this case, the typical approach is to train the model for the normal class and then use the model to identify anomalies in the test data;
- **Unsupervised:** These are the techniques most widely applicable since they do not require any training data. This only means that these techniques find anomalies in an unsupervised way, and not because the points don't have labels. Labels are only used for assessment of the model performance. In this mode of operation, AD techniques implicitly assume that anomalies are rare when compared to normal instances and this is important otherwise, such techniques suffer from high False Alarm Rate (FAR).

AD techniques can report the anomalies in one of the following two ways [16]:

- **Scores:** Each observation in the data is assigned an anomaly score depending on the degree of outlierness of that observation (if that observation is more or less anomalous). Therefore, the output is a ranked list of anomalies and one may choose to analyze the top N anomalies in that list;
- **Labels:** Here, each test instance is assigned a binary label usually *0* or *"n"* for the normal class, and *1* or *"o"* for the anomalous class.

Table 3.1 aggregates the information about the classification of the studied techniques in this dissertation, Local Outlier Factor (LOF), Connectivity-Based Outlier Factor (COF), Cluster-Based Local Outlier Factor (CBLOF), Histogram-Based Outlier Detection (HBOS), k-Nearest Neighbours (k-NN), Subspace Outlier Detection (SOD), Local Minimum Spanning Tree (LoMST), Fuzzy c-Means (FCM) and Archetypal Analysis (AA).

Algorithms	**Semi-Supervised**	**Unsupervised**	**Scores**
LOF	✓		✓
COF	✓		✓
CBLOF	✓		✓
HBOS	✓		✓
K-NN	✓		✓
SOD	✓		✓
FCM		✓	✓
AA		✓	✓
LoMST		✓	✓

Table 3.1: Classification of the studied techniques according to their learning mode *vs* output returned.

Classification based anomaly detection techniques operate, assuming that the distinction between normal and anomalous classes can be learnt in the given feature space [16]. Such techniques can be grouped into two categories: **multi-class** and **one-class** AD techniques.

In this dissertation, the semi-supervised techniques studied fall into the one-class category. One-class techniques learn a discriminative boundary around the training data points, hence any point in the test set that does not belong inside the learnt boundary is considered an anomaly. Figure 3.1 illustrates this process.

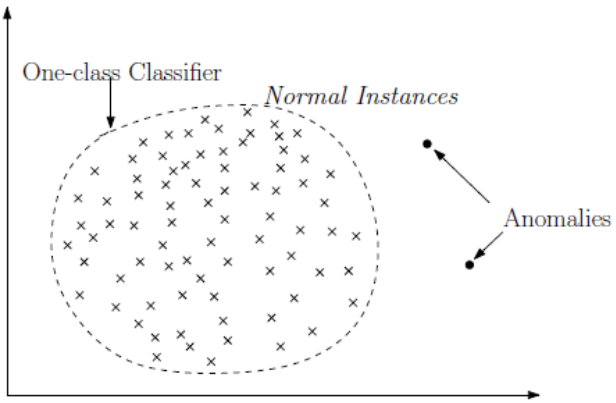

Figure 3.1: One-class Anomaly Detection
Source: taken from [16]

3.2.2 Advantages and Disadvantages

Nearest-neighbour-based techniques like *k*-NN, LOF and COF, present the following advantages and disadvantages [16].

a) Advantages:

1) They are naturally unsupervised and do not make any assumptions about the data, hence, they are purely data driven;
2) Semi-supervised techniques perform better than unsupervised ones regarding missed anomalies, since it is not likely for an anomaly to form a close neighbourhood in the training data;
3) Can be adapted easily to a different data type, only needing to define an appropriate distance measure for the given data.

b) Disadvantages:

1) For unsupervised techniques, if the normal data observations do not have enough close neighbours or if anomalies have close neighbours, many missed anomalies occur;
2) For semi-supervised techniques, if the normal observations in test data do not have enough similar normal observations in the training data, the False Positive Rate (FPR) is high;
3) The computational complexity of the testing phase is a significant challenge;
4) The performance is tightly related to the distance measure, which can be challenging to define when the data is complex.

The advantages and disadvantages of clustering-based techniques like CBLOF are the following [16].

a) Advantages:

1) Can operate in unsupervised mode;
2) Can be adapted to different data types easily, by simply choosing an appropriate clustering algorithm that can handle the particular data type;
3) Testing phase is fast since the number of cluster one must compare every observation's belongingness is small.

b) Disadvantages:

1) Performance is closely related to the effectiveness of the clustering algorithm in capturing the cluster structure of normal observations;
2) Many techniques are not optimized for AD;
3) The majority of clustering algorithms try to assign every instance to some cluster. Techniques that operate with the assumption that anomalies do not belong to any cluster, might considered anomalies as normal instances;
4) Several techniques are effective only when the anomalies do not for significant clusters;
5) When clustering algorithms run in $O(N^2 \cdot d)$, with N being the number of instances and d the number of dimensions, clustering the data is often a bottleneck.

Statistical techniques like HBOS present the following advantages and disadvantages [16].

a) Advantages:

1) If the data distribution remains true, these techniques are a reasonable solution for AD;
2) The anomaly score is associated with a confidence interval, which can help on deciding with additional information about any test instance;
3) If the distribution is robust to anomalies in data, these techniques can operate in unsupervised mode.

b) Disadvantages:

1) Rely on the assumption that the data is generated from a particular distribution and sometimes this does not happen, specially for high dimensional data sets;

2) They are not able to capture the interactions between different attributes. An anomaly can have attribute values that are individually frequent, but their combination is rare.

The SOD algorithm has the following advantages and disadvantages [39].

a) Advantages:

1) Really good at handling high dimensional data sets;
2) Strong at detecting local outliers.

b) Disadvantages:

1) Difficulty on how to find a good reference set;
2) Normalization of scores is oversimplistic.

The LoMST algorithm presents the following advantages [5].

a) Advantages:

1) Achieves the merit of subspace-based techniques without losing the benefits of local neighbourhood methods;
2) Computationally efficient.

The advantages and disadvantages of FCM are the following[62].

a) Advantages:

1) Is unsupervised;
2) Converges.

b) Disadvantages:

1) Long computational time for really big data sets;
2) Sensitivity to the initial guess (speed of convergence, local minima);
3) Sensitivity to noise and one expects low (or even no) membership degree for outliers (noisy points).

3.3 Semi-supervised Anomaly Detection Algorithms

3.3.1 Local Outlier Factor

The LOF algorithm was the pioneer to not label an outlier as a binary property but to quantify how outlying a point is. This outlying factor is local in the sense that only a certain neighbourhood of each point is considered [14].

The LOF can be calculated in three steps:

1) First, the k-NN of each observation need to be found and in case of a tie of the k^{th} neighbour, more neighbours can be used;

2) With the k-NN computed, N_{MinPts}, the local density for an object, p, is estimated by calculating the Local Reachability Density (LRD) in equation (3.1):

$$LRD_{MinPts}(p) = 1/\left(\frac{\sum_{o \in N_{MinPts}(p)} reach\text{-}dist_{MinPts}(p,o)}{|N_{MinPts}(p)|} \right) \tag{3.1}$$

   where $|N_{MinPts}(p)|$ is the cardinality of p neighbourhood;

3) Finally, the LRD is compared against the LRDs of the point k neighbours, and LOF is computed in equation (3.2):

$$LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{LRD_{MinPts}(o)}{LRD_{MinPts}(p)}}{|N_{MinPts}(p)|} \tag{3.2}$$

With $reach\text{-}dist_{MinPts}(p,o)$ denoted as the reachability distance of point p to o, LRD of p is the inverse of the average reachability distance based on the $MinPts(p)$. If there are duplicates in terms of spatial coordinates, LRD can actually be ∞. The LOF of p is the average of the ratio of the LRD of p and those of p nearest neighbours, and the degree to which p is quantified as an outlier. The lower the LRD of p is and the higher the LRDs of p nearest neighbours are, the higher will be the LOF value.

The LOF algorithm has a hyperparameter k, or *MinPts* as described by the authors, that is directly related to the approach, global or local when considering the neighbourhood of each point. A small value of k can better capture the complex structure of some real-world data sets and detect local outliers but is more erroneous when having much noise in the data. A large value of k takes a global view of the data set and because of that can miss local outliers but is faster when compared to a local approach where smaller neighbourhood regions are considered.

For objects inside a cluster, the LOF values of those objects are approximately 1, and the LOF value depends on the input parameter *MinPts*. Because of this, the authors propose to establish upper and lower bounds for *MinPts*, called *MinPtsUB* and *MinPtsLB* respectively. For *MinPtsLB* the authors state that its value should be the minimum number

of objects a cluster must have, to make other objects relative to that cluster, local outliers. *MinPtsUB* should be the maximum cardinality of a cluster so that all objects in that cluster can be local outliers.

In conclusion, if the density of a point is much smaller than the densities of its neighbours, a LOF greater than 1, the point is isolated from dense areas, and considered an outlier.

3.3.2 Connectivity-Based Outlier Factor

COF aims to differentiate "low density" from "isolativity" . While low density normally refers to the fact that the number of objects in the "close" neighbourhood of an object is (relatively) small, isolativity refers to the degree that an object is "connected" to other objects. In the general case, a low-density outlier results from deviating from a high-density pattern, and an isolated outlier results from deviating from a connected pattern [63].

COF improves the effectiveness of an existing LOF scheme in such a way that compensates the shortcoming of assuming that the data is distributed in a spherical way around the instance, and estimates the local density of the neighbourhood using a shortest-path approach, called the chaining distance. Mathematically, this chaining distance is the minimum of the sum of all distances connecting all k neighbours and the instance.

The COF of an observation p with respect to its k-neighbourhood is defined in equation (3.3):

$$COF_k(p) = \frac{|N_k(p)| \cdot ac\text{-}dist_{N_k(p)}(p)}{\sum_{o \in N_k(p)} ac\text{-}dist_{N_k(o)}(o)} \tag{3.3}$$

with $ac\text{-}dist$ being the average chaining distance, COF can be described as the ratio of the average chaining distance from p to $N_k(p)$ and the average of the average chaining distances from p k neighbours to their own k neighbours. This indicates how far a point shifts from a pattern and where COF and LOF differ. With this notion of "shifting", it is possible to conclude the following:

- Strongly shifted points have larger $ac\text{-}dist$ than weakly shifted ones. In the general case, the majority of the k neighbours of a strongly shifted point should have small $ac\text{-}dist$. This leads to higher COF values for strongly shifted points;
- Weakly shifted points, have a k-neighbourhood with comparable $ac\text{-}dist$, leading to smaller COF values for such weakly shifted points. These points are those that belong to the patter itself, hence, their COF is close to 1.

Like LOF, objects inside a cluster have a COF value close to 1, and for the k-neighbourhood of an object p, $COF_k(p)$ is greater or equal than $\frac{1}{1+\epsilon}$ and lower or equal than $1+\epsilon$, where ϵ is a small value. The authors followed the same approach as [14] to establish the lower and upper bounds for COF.

COF has a time complexity similar to LOF, linear time for low-dimensional data sets and quadratic for really high-dimensional data sets, and the same hyperparameter k.

3.3.3 Cluster-Based Local Outlier Factor

The CBLOF uses clustering to determine dense areas in the data and performs a density estimation for each cluster afterwards [34]. In theory, every clustering algorithm can be used to cluster the data in a first step, however k-Means is commonly used to take advantage of the low computational complexity. After clustering, CBLOF uses a heuristic to classify the resulting clusters into large and small clusters. Finally, an anomaly score is computed by the distance of each instance to its cluster centre multiplied by the instances belonging to its cluster.

The authors chose the *Squeezer* algorithm [33] as the clustering method background, since it provides the following novel features:

- High quality clustering results and scalability;
- The ability to handle high dimensional datasets effectively;
- It does not require the number of desired clusters, k, as input parameter.

This algorithm has a linear time complexity of $O(N)$ when compared to LOF and COF which is quadratic, where N is the number of observations in the data set.

Finding the CBLOF for each record is the second part of the Find Cluster-Based Local Outlier Factor (FindCBLOF) algorithm, and the first part is clustering the data. FindCBLOF has two major hyperparameters, α and β. If α is equal to 0.9 means that clusters containing 90% of the data will be denominated large clusters, and if β is equal to 5 means that any large cluster is at least 5 times of the size of any small cluster.

Summarizing, if an observation t belongs to a small cluster, its CBLOF is determined by the size of this cluster and the distance between the observation and its closest cluster. If the observation lies in a large cluster, the CBLOF value will be determined by the size of the cluster and the distance between the observation and the cluster it belongs to. The distance between the record and the cluster, can be the similarity measure used in the clustering algorithm. For simplicity, the CBLOF of a point t is calculated in equation (3.4):

$$CBLOF(t) = |C_i| \cdot min(distance(t, C_j)) \tag{3.4}$$

where C_i is a small cluster and C_j is a large cluster

$$CBLOF(t) = |C_i| \cdot distance(t, C_i) \tag{3.5}$$

where, in equation (3.5), C_i is a large cluster

3.3.4 Histogram-Based Outlier Detection

The HBOS algorithm assumes independence of the features making it incredibly faster, especially on large data sets, than multivariate approaches at the cost of less precision. It is reliable at detecting global anomalies but poor at detecting local ones [27].

This algorithm can be summarized in the following way: for every single feature (dimension), a univariate histogram is constructed first, where the height of every single bin represents a density estimation. The histograms are then normalized such that the maximum height is 1.0. This ensures an equal weight of each feature to the outlier score. Finally, the HBOS of every instance p is calculated, as shown in equation (3.6), using the corresponding height of the bins where the instance is located:

$$HBOS(p) = \sum_{i=0}^{d} \log\left(\frac{1}{hist_i(p)}\right) \tag{3.6}$$

The idea is very similar to the *Naive Bayes* algorithm, where all independent features probabilities are multiplied, but instead of multiplication the authors applied the sum of the logarithms. This makes HBOS a discrete *Naive Bayes* probability model, and the reason for this change is to make HBOS less sensitive to errors due to floating point precision in extremely unbalanced data causing very high scores.

The number of bins k, input parameter, needs to be set. An often-used rule of thumb is setting k to the square root of the number of instances N.

Important notes when it comes to the creation of histograms. If features are categorical, simple counting of each category is performed and the relative frequency (height of the histogram) is computed. For numerical features there are two different approaches:

1) **Static bin-width histograms.** This is the standard technique where k is equal to the width bins over the value range. The height of the bins is estimated through the frequency of samples falling into each bin;

2) **Dynamic bin-width histograms.** In this technique, the values are sorted first and a fixed amount of $\frac{N}{k}$ successive values are grouped into a single bin. This gives the same area (number of observations) for all bins and the width of the bins is determined by the first and last value, hence, bins covering a larger interval of values have less height (lower density).

The major reason for having these two approaches when creating histograms is due to the fact of having very different distributions of the feature values in real-world data sets. This happens in AD tasks where large gaps of value ranges exist, outliers being far away from normal data, therefore, the dynamic width approach is recommended by the authors.

When it comes to the complexity evaluation, HBOS is 7 times faster than nearest-neighbour-based methods and 5 times faster than clustering-based ones. HBOS works in

linear time $O(n)$ in case of fixed bin-width mode, and $O(n \cdot \log(n))$ in dynamic bin-width mode, with n being the number of points.

3.3.5 *k*-Nearest-Neighbours

The *k*-NN unsupervised anomaly detection algorithm was the first to bring the notion of an outlier as based on the distance of a point from its k^{th} nearest neighbour [53].

In *k*-NN the user no longer needs to specify the distance d to define the neighbourhood of a point p as an input parameter. Only needs to specify the number of outliers n to rank in the top n, and k-neighbourhood. The authors define an outlier the following way: *"Given a k and n, a point p is an outlier if no more than n-1 other points in the data set have a higher value for D^k than p"*, where $D^k(p)$ is the distance of the k^{th} nearest neighbour of p.

Each point it's ranked based on its distance to its k^{th} nearest neighbour and the top n points in this ranking are declared as outliers. If we denote the distance to a point's k^{th} neighbour as D then, points with larger values for D have more sparse neighbourhoods and are thus typically stronger outliers than points belonging to dense clusters which will tend to have lower values of D. *k*-NN is sensitive to its hyperparameter k that denotes the number of neighbours one wants to consider, and this value should be chosen accordingly.

A partition-based algorithm component makes *k*-NN not computationally expensive. Basically, the partition-based algorithm discards points with really small distances, that cannot possibly make it to the top n outliers, from their k^{th} nearest neighbours. By partitioning the data set, it is possible to determine is a point p belongs to the top n without actually computing the precise value of $D^k(p)$.

The authors concluded that to ensure a good performance of the partition component of *k*-NN is to choose the number of partitions in such a way the average of observations per partition is small, but not too small compared to k.

3.3.6 Subspace Outlier Detection

The SOD aims to analyze for each point, how well it fits the subspace that is spanned by a set of reference points [39]. This subspace hyperplane has a high variance of the reference points when compared to the variance of the reference points in the perpendicular subspace, which is low. The variance is simply the average squared distance of the reference points to the mean value. For each feature with a low variance for its reference points, the value of the corresponding subspace is 1 and for the remaining features is 0. A value near 0 indicates that the observation p fits very well to the hyperplane (not an outlier), where a high value indicates that p is an outlier. The Euclidean distance is used to naturally express the deviation of any observation to a subspace hyperplane, and therefore the measure of outlierness of any observation p.

The final SOD value can be defined in equation (3.7):

$$SOD_{R(p)}(p) = \frac{dist(o, H(R(p)))}{|v^{R(p)}|_1} \tag{3.7}$$

where *H(R(p))* is the hyperplane spanned by the reference set of p, *R(p)*, and $|v^{R(p)}|_1$ is the number of relevant dimensions.

The SOD algorithm relies on two input parameters. First, specifies the k number of nearest neighbours that are considered to compute the shared nearest neighbour similarity. Second, l specifies the size of the reference sets and this value should be smaller or equal than k. There is also a third parameter α that sets a threshold to decide whether an attribute is relevant or not. If the variance of the reference set along a certain feature is smaller than α times of the expected variance, the feature is considered relevant. The authors achieved consistently good results with $\alpha = 0.8$. By specifying for each outlier the features that are relevant for its outlierness, SOD has the property of being a quantitative outlier model.

The SOD algorithm runs in $O(d \cdot n^2)$ when compared to most existing AD algorithms, where d corresponds to the number of dimensions and n the number of observations.

The experiments conducted by the authors showed that SOD can find more interesting and meaningful outliers in high dimensional data, and that SOD is strong at detecting local outliers. The authors also noticed, even with a high number of dimensions and most of them being irrelevant attributes, SOD retrieves really few false positives.

3.4 Unsupervised Anomaly Detection Algorithms

3.4.1 Clustering to Anomaly Detection: a review

In [30] a large variety of outlier detection methods are described in detail. Clustering-based approaches assume that an outlier is an object that belongs to a small and remote cluster, or does not belong to any cluster. This philosophy leads to three general approaches: (i) if the object does not belong to any cluster then it is identified as an outlier; (ii) if there is a large distance between the object and the cluster to which it is closest, then it is an outlier, and (iii) if the object is part of a small or sparse cluster then all objects in that cluster are outliers.

The authors in [7] developed automated techniques for monitoring and recognizing activities in video, and for detecting threatening behaviour in maritime data. The work was focused on borderline/boundary cases of anomalous behaviour, which can be a challenging task due to the fact anomalies are rare to occur in the maritime domain. This issue was addressed by generating synthetic anomalous instances using a parametric statistical approach from real, non-anomalous data, which allowed to control the frequency if abnormal behaviours present in the generated data. The authors considered two types of algorithms, global[1] and local[2] ones. An Expectation–Maximization (EM) version of k-Means clustering and the k-NN Localized p-value Estimator (KNN-LPE)

[1] Methods that consider the information contained in an entire data set to identify anomalies.
[2] Methods that operate only on the information in the neighbourhood of the query.

for the global algorithms, and a variant of the LOF [14] and k-NN Normalized Average Density (KNN-NAD) for the local ones. The k-Means [36] is a popular distance-based clustering algorithm while KNN-LPE, LOF and KNN-NAD perform density-based AD. EM k-Means works by assuming that normal data instances lie close to the closest cluster centroid while anomalies are further away; normal data instances belong to large ad dense clusters, while anomalies belong to small or sparse clusters. KNN-LPE uses the entire training set to compute a score, which is an estimate of the probability that an instance is anomalous; if this score is above a threshold the test instance is anomalous. LOF Normalized (LOFN) like LOF is particularly suitable for outlier analysis in large multi-dimensional data sets, but in this study, it normalizes the data to $[0, 1]$. Finally, the KNN-NAD is a variant of LOFN with reduced run time; the score is computed the same way, with the difference residing in not computing the local reachability distances but the distance between a node and its k^{th} nearest neighbour. The authors' main goal was to compare the performance of local and global AD algorithms on a ground-based maritime AD task. The algorithms are supposed to differ only on borderline cases and not when the anomaly cases are distinct from normal instances. To assess this performance the area under the ROC curve was used. ROC curves are often used to compare the performance of binary classifiers [13] and plot the True Positive Rate (TPR)[3] against the FPR[4]. The authors concluded that the KNN-LPE, a global density approach, has the lowest runtime possible because it lends itself to catching of distance computations. KNN-LPE also has the highest TPR and is best suited for real-time data acquisition.

The authors in [59] bring up the problem of AD in high-dimensional datasets and propose a framework called Dimensionality Reduction Anomaly Meta-Algorithm (DRAMA)[5] to solve it. Significantly increasing the number of features leads to the problem of the, so-called, curse of dimensionality [3]: the performance of most machine learning algorithms deteriorates as the dimensionality of the feature space increases dramatically. The key reasons for the "curse" are that distance measures become less and less informative and feature space volume grows exponentially in higher dimensions. The proposed method can be described as follows: it performs dimensionality reduction (encoding) of data to a lower-dimensional space, followed by clustering to find the main prototypes in the data. After these steps, uplifting to the original space (decoding; optional) is performed and finally distance measurements between the test data and the prototypes (the main clustered components) to rank potential anomalies. An important note about DRAMA, agglomerative clustering was used for prototype detection because the authors performed experiments that showed it was superior to other methods like k-Means. The authors used 20 real-worlds datasets, only 20 due to computational resources, chosen at random from

[3]TPR = #anomalies correctly classified/Total number of anomalies

[4]FPR = #non-anomalies classified as anomalies/Total number of non-anomalies

[5]DRAMA is based on the popular *scikit-learn* [52] and *TensorFlow* [1] packages and comes with a Jupyter notebook interface for ease of use.

the Outlier Detection DataSets (ODSS)[6] database and many different kinds of anomalies were studied. DRAMA was compared against two popular general algorithms, LOF [14] and Isolation Forest (iForest) [41], and the means and best performances for two relevant metrics suited for AD, ROC AUC and Rank-Weighted Score (RWS) [55] were used. The RWS rewards the algorithms whose anomaly scores correlate well with the true probability of being an anomaly. The authors concluded that the true potential of DRAMA only shines in high-dimensional datasets ($n_f = 3000$), where n_f represent the number of features, and that the proposed method beat LOF and iForest on every simulated data challenge and on 17 out of 20 real-world challenges in terms of area under ROC curve. On the very inhomogeneous and fairly low-dimensional real-world datasets tested, DRAMA was highly competitive with LOF and iForest, showing that dimensionality reduction and clustering is a valuable approach to AD.

In [64] the authors' work consisted of a specific review on clustering techniques for AD and used network attacks as the background problem. Clustering is a popular unsupervised classification technique that separates the data space in regions based on similarity/dissimilarity metric, where similar elements are placed in the same cluster while dissimilar ones are placed in separate clusters. This method can find abnormal data having standard, or targeted automatically from the data set, in which it is not known what is normal, or what is abnormal. For this to work the unsupervised algorithm has two rules for the dataset: one is that the record number of normal activities must be bigger than the record number of intrusion events, the other is that there is an essential difference between normal and abnormal records. The unsupervised algorithms covered by the authors were, k-Means, FCM [12], Adaptive hierarchical based clustering [17], k-Means+ID3 [26] and Coclustering [56]. With the authors' work, the following can be outlined: k-Means has higher performance compared to others algorithms, FCM is virtually identical to k-Means except for a vector that expresses the percentage of belonging of a given point to each of the clusters. Therefore FCM is slower but shows better results for elongated clusters. FCM approach has proven to be very effective for solving many cluster analysis problems [32] and in practice converges quickly, usually in 30 iterations [31]; With filtering noise and updating profiles at any time, adaptive hierarchical clustering is quite effective; The combined application of the two algorithms overcomes some limitations of each algorithm when applied individually in k-Means+ID3; Finally, coclustering is not stated with enough proof to be a reliable and robust algorithm for AD.

Defining a normal region is a challenging problem as the exact notion of an anomaly is different for different application domains. In [44] the authors propose a method based on k-Means clustering for an initial partition of the dataset into k closest clusters, followed by C5.0 technique to built a Decision Tree for each closest cluster and the rules created

[6] http://odds.cs.stonybrook.edu/

by each decision tree to detect anomalies in the dataset. The reason for choosing the above algorithms was: k-Means has time complexity $O(n*k*m)$ where n is the number of clusters, k is the number of patterns and m is the number of iterations. k-Means space complexity is $O(n+k)$ and its scalability is order independent; The reason for choosing C5.0 [51] is due to the fact it is more efficient and its Decision Tree is smaller in comparison with C4.5 because unnecessary attributes are automatically removed by C5.0. The proposed algorithm can be briefly described in the following way: *(i)* initially k-Means is used for partitioning the dataset into k closest clusters using Euclidean distance [54] and *(ii)* C5.0 technique is then applied to build a Decision Tree for each cluster and classify each instance into normal or anomaly. Phases *(i)* and *(ii)* correspond to the selection phase and classification phase respectively. In this work temperature related anomalies were studied and the performance evaluation was done by splitting the data into 90% for train and 10% for test, followed by an analysis of the confusion matrix regarding the number of correct and incorrect classification. By doing this the authors concluded that the proposed algorithm gives impressive classification accuracy showing no errors in the training set but in the test set, 393 variables were correctly predicted and 19 incorrectly predicted.

The authors in [6] present a study on detecting and preventing attacks against computer networks. The systems used to protect against such attacks have a significant limitation since the signature database should be updated frequently. Ant Clustering Algorithm (ACA) is a popular unsupervised learning algorithm [65] that needs to be complemented with other algorithms for the classification process. In this work, the authors propose a hybrid method based on two phases: a training phase where ACA is used to determine clusters; and a second phase where a fuzzy approach is going to detect anomalies in new monitored data by combining two distance-based metrics. The reasons, confirmed by the authors through research and run experiments, for choosing ACA and a fuzzy approach to propose a novel hybrid Intrusion Detection System (IDS) scheme are related to the fact that ACA provides a higher detection rate and fuzzy logic can reduce the FAR[7] with higher reliability. The high detection rate of ACA is achieved since it provides minimum intra-cluster distance and maximum inter-cluster distance in order to present the inherent structures and knowledge from data patterns, where intra-cluster distance and inter-cluster distance mean better compactness and better separateness respectively. Fuzzy can reduce the FAR because it helps construct more abstract and flexible patterns for intrusion detection and thus greatly increases the robustness adaption ability of the detection system. To evaluate the performance of the proposed model the authors used the Detection Rate (DR)[8], FAR and False Negative Rate (FNR)[9] as performance metrics.

[7] Calculated by number of legitimate instances detected as attack instances divided by the total normal (legitimate) instances included in the data test.

[8] The ratio of the number of correctly detected attacks to the total number of attacks.

[9] Represents the number of attacks that were unable to be detected by the proposed method.

The goal of a IDS is always to provide high detection rates and low false alarm rates, such as the hybrid-IDS scheme proposed by the authors. The authors' conclusions were: ACA is a proper algorithm for high density and high dimensional data; it is very effective to detect both known and unknown attacks, however it still provides high FAR.

Sometimes some outliers are not detected by a single clustering-based outlier detection approach. To overcome this limitation, the authors in [37] propose a novel Cooperative Clustering Outlier Detection (CCOD) algorithm that involves multiple clustering techniques using the methodology of cooperation. The authors chose as their work methodology, to compare and analyze their model against a popular clustering-based outlier detection technique called FindCBLOF. FindCBLOF [34] assumes that outliers form very small-sized clusters, and the detection accuracy of FindCBLOF is mainly based on the quality of the adopted clustering technique. The proposed model, unlike FindCBLOF, provides efficient outlier detection and data clustering capabilities, and can be briefly described as follows: a first phase that obtains the intersection between the generated sub-clusters; the second phase represents each sub-cluster with a histogram representation of the pair-wise similarities between objects in the same sub-cluster; in the third phase the identification of a possible set of outliers is done by assigning a cooperative outlier factor to each object in each sub-cluster; finally the fourth phase returns the overall set of candidate outliers that affects the homogeneity of the merging process. The authors also stated that it has been experimentally proven that better clustering solutions reveal better detection of outliers using the notion of CBLOF. In the FindCBLOF algorithm, outliers are returned as objects with higher CBLOF values where the CBLOF is a measure of both the size of the cluster the object belongs to and the distance between the object and its closest cluster (if the object lies in a small cluster). The detection accuracy of outliers was evaluated comparing the proposed method against the LOF [14] algorithm using the *TopRatio* and *LocalTopRatio* metrics, that represent the number of detected outliers. The Separation Index (SI)[10] was used as the internal quality measure, which does not require prior knowledge about the data. The smaller the SI, the more separate the clusters. With these evaluation metrics, the authors concluded that the proposed method achieves better detection of outliers in the data due to the fact cooperative clustering outperforms non-cooperative; better results can be achieved in the clustering after removing the detected outliers.

The research work of the authors in [40] is yet another AD in network traffic using a IDS. For this purpose, FCM was used due to the fact soft clustering is more flexible than hard clustering and is practical for outlier detection because of the natural treatment of data. However, the authors propose a model that combines FCM clustering Gaussian

[10]The ratio of average within-cluster variance (cluster scatter) to the minimum pair-wise dissimilarity (measured by the cosine correlation measure) between clusters.

Mixture models (GMM) and feature transformation, with some modifications to the objective function and the distance function that reduce the computational complexity of FCM while keeping clustering accurate. In many cases, the measured features are not useful in producing a model, features may be irrelevant or redundant. Dimension reduction often leads to simpler models and fewer measured variables, with consequent benefits when measurements are expensive and visualization is important. Nonnegative Matrix Factorization (NMF) [68] and Principal Component Analysis (PCA) [2] are widely used techniques for dimension reduction, more specifically feature transformation. The authors used profiles data [69] extracted from raw Netflow data and chose 27 features [50] using k-Means and Fuzzy GMM to classify the traffic flow. With this research work the authors concluded: Fuzzy GMM is more robust than k-Means and for AD, outliers are a small part of the data; PCA can extract more effective features than NMF however, its computational overhead is higher than NMF which can be balanced by the reduction of feature numbers; the experimental results also showed that the selection of cluster number is crucial for classifying data and, re-running clustering techniques plus cross-validation [58] can help significantly correctly classifying the data; finally using diverse clustering techniques and different cluster numbers, it is possible to perform cross-validation and find more possible outliers.

3.4.2 A Fuzzy Clustering Approach to Anomaly Detection

3.4.2.1 Fuzzy c-Means

In many cases, real-world data sets show some sort of imprecision leading to a more difficult task when it comes to the assignment of structure to data. In such a case, assigning an entity to a cluster becomes non-trivial due to the fact, clusters having no sharp boundaries. This is where the concept of membership in fuzzy sets offers special advantages over crisp ones in a way that an entity is not only belonging or not belonging but has a certain degree of belongingness to a given set.

The core and the origin of an infinite family of fuzzy c-means approaches was the FCM [12]. FCM is considered a fuzzy analogue to c-means crisp clustering because it employs the same definition of clusters, via prototypes. The main difference of FCM compared to c-means relates to the fact that for FCM, the membership values, even though expressing the similarity between observations and prototypes, are not involved in the reconstruction of observations from the cluster prototypes.

The two main hyperparameters of FCM are, c that denotes the number of clusters one wishes to partition the data, and m that express the degree of fuzziness of the cluster partition. Parameter m must be chosen accordingly, values of $m \to \infty$ lead to entirely fuzzy partitions, and for $m \to 1$ FCM becomes a crisp algorithm originating more and more crisp partitions [47]. In most applications of FCM, the parameter m is chosen in the range $[1.1, 5.0]$ [47], with $m = 2$ the most common value [10]. Another input parameter exists, ϵ, that is responsible for controlling the duration of the iterative cycle as well as

the quality of terminal estimates. This parameter should be between 0.0001 and 0.01 range of values [47].

The main goal of FCM is to find an optimal c-partition and respective prototypes by minimizing the sum of squared errors, shown in equation (3.8), between observations in the data and cluster prototypes, whose weights correspond of a membership value. While prototypes are determined as weighted averages of the cluster points, membership functions are determined based on a distance function, expressing the proximity of points to the cluster centres. To evaluate a c-partition returned by FCM one needs to run FCM for different values of c, $c \geq 2$, and then for each c-partition can make use of validation functions [11] to assess the quality of such partitions.

$$\underset{C}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{j=1}^{c} (\mu_{ij})^m \cdot \|\varkappa_i - v_j\|^2 \tag{3.8}$$

where, C represents a list of c cluster centres, n is the number of data points, c represents the number of the cluster centre, μ_{ij} represents the the membership of i^{th} data to j^{th} cluster centre, m is the fuzziness index with $m \in [1, \infty[$ and $\|\varkappa_i - v_j\|^2$ is the Euclidean[11] distance between i^{th} data and j^{th} cluster centre.

3.4.2.2 FurthestSum-AA

The statistical technique AA aims to represent the data as a convex combination of pure or **extremal** types called archetypes [19]. While archetypes might not represent real points in the data, archetypoids are real observations. Archetypes are built as a convex combination of the observations. Important to point out that archetypes are **not** outliers.

In [48] the FurthestSum-Archetypal Analysis (FS-AA) algorithm is briefly described as an initialization strategy for AA, with the purpose of maximizing the speed of convergence of the algorithm and decreasing the chance of generating non-significant solutions. The idea behind the FS-AA algorithm is based on the FurthestFirst (FF) [35], a known method for initializing k-Means. The FF method takes as input the number of seeds that are supposed to be used and then: (i) it selects a random set of data point, the seeds; (ii) it begins to select the remaining data points such that they maximize the distance from already selected seeds. The points selected by FS-AA are guaranteed to belong to the minimal convex set of the unselected data points.

The authors point out some limitations inherent to the AA clustering approach: (1) the input number of clusters, as the number of archetypes increases the way in which they reflect the shape of the data will change for every single one; (2) validation method, the authors mention that none clustering validation index is adequate for validating the partitions generated by AA.

[11] Euclidean distance is the shortest distance between two points in an N-dimensional space also known as Euclidean space. It is used as a common metric to measure the similarity between two data points.

The AA technique can be extended to functional data (FD), to make it possible to be used in AD problems.

Functional Data Analysis (FDA) is a relatively new field that sees each curve as a distinct observation in itself. The objectives of FDA are essentially the same as those of any other branch of statistics.

A practical application of FDA is presented in [45], as a way to detect anomalous flows in urban water networks. Hydraulic variables are recorded in real-time hence they are FD. In this work the authors propose a FDA-based method for AD in the flows of urban water networks, presenting 3 main novelties: (1) treating real-time primary hydraulic variables as FD for the first time; (2) a FDA method to validate data and identify anomalies; (3) a new procedure for functional outlier detection based on [23] and comparison of its performance with other procedures. The data used consisted of 5-minute records of water inflows from three municipal water utilities on the eastern coast of Spain.

With this work, the authors concluded that the proposed procedure provided consistently very good results for all the types of outliers. Figures 3.2 and 3.3 aim to clarify the procedure and also give a general idea of how AA (in this specific example Archetypoid Analysis (ADA)) works. The x-axis represents the domain of the function, normalized, used by the authors to generate the points used in this example.

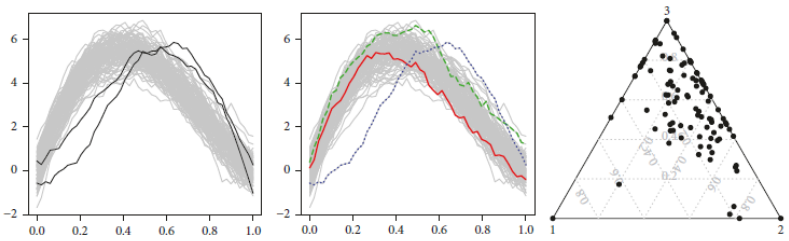

Figure 3.2: Urban water flow data with outliers. Left-hand panel: functions are generated from the main model (gray) and the contamination model (black). Central panel: archetypoids are represented with (red) solid, (green) dashed, and (blue) dotted lines, respectively. Right-hand panel: the vertices of the triangle represent each archetypoid.
Source: Taken from [45]

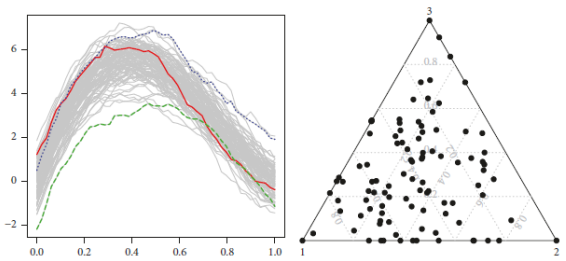

Figure 3.3: Urban water flow data with no outliers. Left-hand panel: archetypoids are represented with (red) solid, (green) dashed, and (blue) dotted lines, respectively. Right-hand panel: the vertices of the triangle represent each archetypoid.
Source: Taken from [45]

3.4.2.3 The Mahalanobis Taguchi System to Fuzzy Clustering

The work in this dissertation presents a novel approach for AD with FCM which is to use ROC curve to find the optimum threshold for the Mahalanobis Taguchi System (MTS), called Modified Mahalanobis Taguchi System (MMTS) [22]. This was also extended for AA and the process is identical. At the time of this dissertation, and up to our knowledge, this has never been done and can be described as follows:

1) Calculate the Mahalanobis Distance (MD)[12] of every point to each cluster centre returned by FCM;

2) Sort these distances in descending order;

3) Make use of the ROC curve to determine the threshold that best discriminates the classes (normal and anomalous), rigorously and systematically.

If the MD of an observation is greater or equal than the optimum threshold, then the observation is considered anomalous.

The ROC curve plots the FPR vs the TPR and these can be calculated as shown in equations (3.9) and (3.10):

$$TP_{rate}^{(\varkappa)} = \frac{TP^{(\varkappa)}}{N_p} \tag{3.9}$$

$$FP_{rate}^{(\varkappa)} = \frac{FP^{(\varkappa)}}{N_n} \tag{3.10}$$

where $TP^{(\varkappa)}$ is the total number of observations classified as positive from the pool of the positive observations (the positive observations whose $MD \geq \varkappa$), $FP^{(\varkappa)}$ is the total number of observations classified as positive from the pool of the negative observations (the negative observations whose $MD \geq \varkappa$), N_p is the total of positive observations and N_n is the total of negative observations.

It is possible to see that point A ($FP_{rate} = 0$, $TP_{rate} = 1$) represents the optimum theoretical solution (best performance) for any classifier. The curve drawn in the Figure 3.4 represents the MTS classifier performance for different threshold values. Changing the threshold will change the point location on the curve (points B, C, D, and E). Summarizing, the goal is to find the closest point that lies on the curve to point A.

This method served as inspiration since the authors concluded that proved itself to be a robust and rigorous process, to determine the threshold to discriminate between two classes in imbalanced data problems [22].

[12] Mahalanobis distance is an effective multivariate distance metric that measures the distance between a point and a distribution.

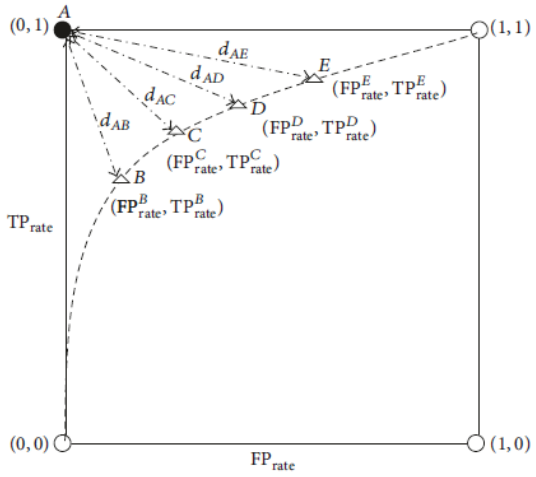

Figure 3.4: Using the ROC curve to determine the optimum threshold.
Source: taken from [22]

3.4.3 Local Minimum Spanning Tree Algorithm

If one considers data instances as vertices and the Euclidean distance between any pairs of data points as the edge weight, then a Minimum Spanning Tree (MST) can be constructed, with no cycles and with the minimum possible total edge weight, to connect all the nodes.

Although the distance between a pair of immediately connected nodes is still Euclidean, the distance between a general pair of nodes is not. The MST-based distance is the geodesic[13] distance between two data points which provides a better metric to differentiate them.

The LoMST [5] proceeds as follow:

1) Identify if anomalous clusters exist. For that, a global MST is built using all the data points and when this is done, a long edge is searched and treated as the connecting edge between the anomalous clusters and the rest of the MST. After this edge is removed two groups remain, the smaller is considered an anomalous cluster. The authors establish a robust way to flag long edges in the following way: plot the edge length distribution specific for each application and set the corresponding q in, $\mu + q \cdot \sigma$. The authors propose $q = 3$. This edge deletion procedure is iterated on the larger group to discard any less discriminative anomalous cluster;

2) Determine if an observation is an anomaly. For that the k-neighbourhood of each point is calculated, isolated and an MST is built in this neighbourhood. These MSTs are local, LoMST, and their total edge length is the LoMST score for the observation differentiating it as anomaly or normal. This score will be compared with the scores of the observations neighbours and this can be done in two ways:

[13] Geodesic distance is the minimum possible distance between two points in a curved surface like the surface of the earth.

compare the observation score with the mean of the neighbours scores, or with the mean-to-standard deviation ratio of the neighbours scores. The authors suggest the former when there are many anomalies and the latter when anomalies are rare;

3) Finally, after the comparisons are done, the LoMST scores will be sorted in decreasing order so the bigger scores imply a higher possibility of being an anomaly.

To choose the input parameter k, the authors propose to run k with a broad range of values between 1 and 100, and select k where the average LoMST scores are stable. The computational complexity of LoMST: (1) building the k-NN for each observation is, $O(pN \log N) + O(kp \log N)$, where the first component represents the time to build the tree structure and the second component represents the k-neighbourhood query time for a single observation; (2) building the LoMST takes $O(|V| \log |E|)$, where $|V|$ and $|E|$ are the number of vertices and edges respectively, and usually remain small.

As concluded by the authors, LoMST by using its distance metric to better detect local pointwise anomalies, stands out as superior and registers the best performance when compared with other anomaly detection algorithms such as LOF, SOD, COF, k-NN and others.

3.5 Cluster validity

"How well does a given partition represent the inherit clustering structure in the data?". This is the question many people try to answer in order to assess how accurately a set of partitions reflect the inherit clustering structure that is embedded in the data. In order for this problem to be adequately tackled there needs to be a measurement or set of measurements that are generated in a systematic manner, in order to help us derive the answer and by doing so removing from the answer any possible biases or inaccurate interpretations. This is all addressed in [48].

These measurements are known as Clustering Validation Indices (CVI) which are functions that given a partition a value, there are CVI with a wide variety of inputs. These values are calculated in such a way that they are comparable between them, enabling the comparison between partitions generated by different clustering algorithms and also by partitions generated by the same algorithms with different parameterization.

There are two types of CVI:

- **Internal Indices:** this set of indices give a evaluation without using the labels of the data. They use metrics that assess how compact are the clusters, the level of separation of the clusters or how different are the clusters between each other, and
- **External Indices:** this set of indices uses the labels of the data in their rating process.

In [48] the authors present 5 soft internal validity indices that have been extensively explored [60] and have been established as adequate to validate partitions generated by AA and Fuzzy Clustering via Proportional Membership (FCPM) algorithms [47]. These indices were subject of analysis in this dissertation and their implementation is available in a R language toolbox provided by [25]:

- **Partition Entropy (PE):** This index is a measurement of how much the clusters are separated, it is achieved by analysing the membership values in terms of their entropy, where the optimization criterion is minimization of PE;
- **Partition Coefficient (PC):** This index is a measurement of how much the clusters "overlap" between each other. This is achieved by averaging the square of the memberships, where the optimization criterion is maximization of PC;
- **Modified Partition Coefficient (MPC):** This is a normalization of the previously described PC, where the optimization criterion is maximization of MPC;
- **Xie-Beni (XB):** This index calculates the ratio between the weighted by membership sum of the squared within-cluster distances to the power m, and the minimum squared distance between every single pair of prototypes, multiplied by the number of points(N). The optimization criterion is minimization of XB;
- **Fuzzy Silhouette Index (FSI):** This index is the "fuzzyfication" of the Average Silhouette Width Criterion, or Crisp Silhouette (CS), developed for hard clustering partitions. The "fuzzyfication" of FSI from CS is achieved through the use of the weighted average instead of averaging its silhouette with the arithmetic mean. The optimization criterion is maximization of FSI.

Summary

The approaches used to address the AD problem depend on the nature of the data that is available for analysis. As an algorithmic framework for data analysis and interpretation, clustering has been widely used in understanding data, revealing fundamental phenomena, and visualizing major tendencies. Detecting anomalies through FDA is a good choice when data is being gathered at fixed frequency intervals. This way data can be expressed as a combination of basis functions and, FDA methods are computationally less demanding than the AD methods discussed previously.

Chapter 4

Data description and preprocessing

In this Chapter all the preliminary work done is highlighted in the first four Sections, in order to draw some initial possible conclusions. The first Section 4.1 refers to the statistical analysis background, Section 4.2 is all about describing in a detailed manner the SCADA system, Sections 4.3 and 4.4 explain the process behind the data analysis exploration and the analysis of the results respectively. The last Section 4.5 describes the preprocessing done, after the preliminary stage, and the datasets used.

4.1 Preliminary Data Analysis

Statistics presents a set of methods whose objective is to synthesize and represent understandably the information contained in the data. Statistical methods can help to understand and describe a variable. It can be understood as a classification or a measure, an amount that changes in each case or study unit. Variability is often found when dealing with engineering problems.

This analysis is meant to treat the raw data that is acquired by SCADA, as these may have registration errors affecting later the results of the descriptive analysis. The preliminary analysis can be carried out through graphics such as boxplots and histograms or measurements. Boxplots are a standardized way of displaying the distribution of data based on a five-number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum"):

- **median (Q2/50th Percentile):** the middle value of the dataset;
- **first quartile (Q1/25th Percentile):** the middle number between the smallest number (not the "minimum") and the median of the dataset;

- **third quartile (Q3/75th Percentile):** the middle value between the median and the highest value (not the "maximum") of the dataset;
- **interquartile range (IQR):** 25th to the 75th percentile;
- **whiskers (shown in blue)**;
- **outliers (shown as green circles)**;
- **"maximum": Q3 + 1.5*IQR**;
- **"minimum": Q1 -1.5*IQR**

The data was visualized using boxplots in two ways: original and normalized. Original as the name implies suffered no modification and, normalized suffered a pretty common transformation called standardization.

Standardization is often called **Z-score normalization** and aims to rescale the features so that they have the properties of a standard normal distribution with, $\mu = 0$ and $\sigma = 1$, where μ is the mean (average) and σ is the standard deviation from the mean; standard scores (also called **z** scores) of the samples are calculated as shown in equation (4.1), where $\varkappa$ is the original data:

$$z = \frac{\varkappa - \mu}{\sigma} \tag{4.1}$$

Standardizing the features is not only important if we are comparing measurements that have different units, but it is also a general requirement for many machine learning algorithms. Figure 4.1 helps to visualize boxplots used in normal distributions.

During this preliminary stage, and to ease the reader when looking at the tables in Section 5.3, it was possible to see that turbines fall into three different categories according to their percentage of outliers (low, medium and high).

- WT8, WT3, WT7 belong to the **high** outlier percentage group with 53.0%, 42.06% and 38.05% respectively;
- WT5, WT6 belong to the **medium** outlier percentage group with 20.57% and 14.6% respectively;
- WT2, WT1, WT4 belong to **low** outlier percentage group with 13.27%, 11.95% and 11.75% respectively

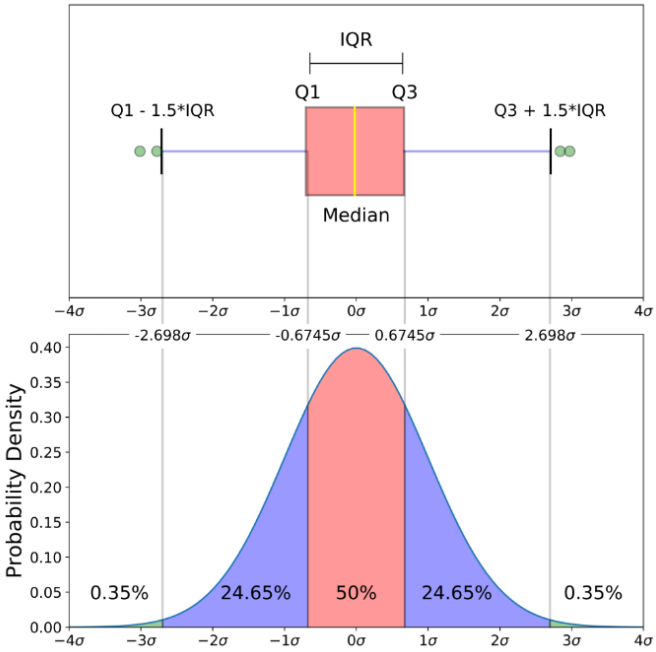

Figure 4.1: Comparison of a boxplot of a nearly normal distribution and a probability density function for a normal distribution.
Source: Taken from[1]

4.2 Description of the SCADA data

The present study consists of the analysis of data registered with the SCADA system of a wind farm[2] for the years 2011, 2012 and 2013 with 52560 rows x 73 columns each, where (9x8 turbines + 1 column for timestamp). Even though the three years are available for this work, the exploratory analysis in Section 4.3 was conducted, at random choice, for the year 2012. The data in question has an observation frequency of 10 minutes and these refer to different variables:

- Active Power (kW);
- Wind speed (m/s);
- Wind direction (º degrees);
- Rotor speed (rpm);
- Outside temperature (T_{out});
- Nacelle Temperature (T_{nac});
- Main Bearing Temperature (T_{MB});

[1] https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51
[2] https://www.iberwind.pt/pt/parques/freita-i/

- Gearbox Main Bearing Temperature (T_{GB});
- Gearbox Oil Sump Temperature (T_{GO})

4.3 Exploratory Data Analysis Background

This section will explain with further detail the visualizations, mentioned in Section 4.1, carried out to analyze the data and the tasks responsible for preprocessing the data.

The exploratory process began by creating a virtual *Windows XP*® environment with *Microsoft SQL Server*® 2005 to access the *.mdf* database file containing the data, using the *VMware Workstation*® software. This was necessary due to the extension of the file containing the *Nordex* database being too old for modern *Microsoft*® *SQL Server Management* 2017. After that, *SQL* scripts were written to extract only the desired features, and *.excel* files were created so that data could be easily accessed. Having the *.excel* files with the data ready to be analyzed, all later tasks were done in a *Python*® 3.7 environment. The first column of the datasets that corresponds to the timestamps of each reading is discarded. All the boxplots shown in this work were done using the *Python*® *matplotlib*[3] package.

Figures 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9 and 4.10 show each SCADA feature measured by the different turbines before data preprocessing (raw data).

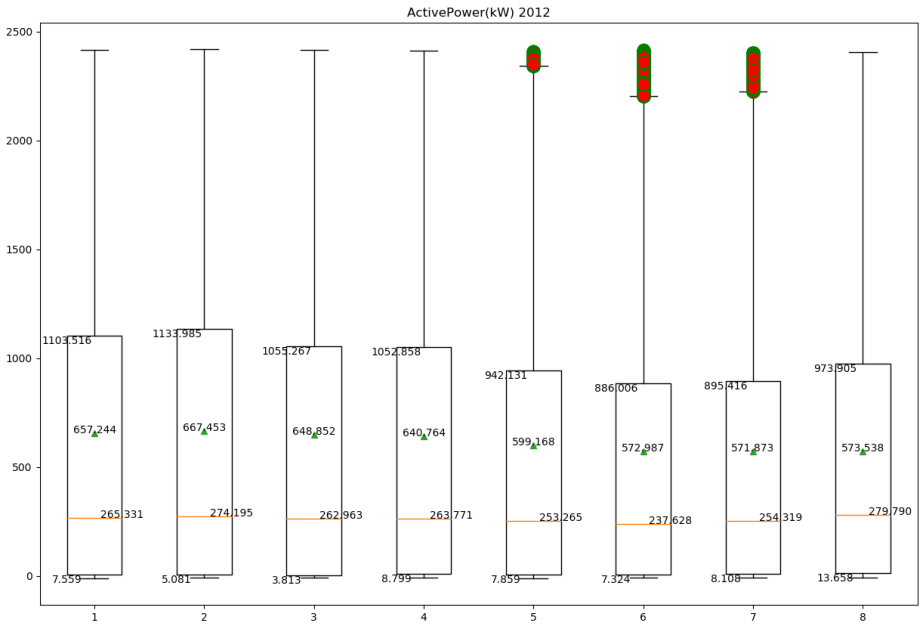

Figure 4.2: Active Power in kW for each turbine.

[3] https://matplotlib.org/

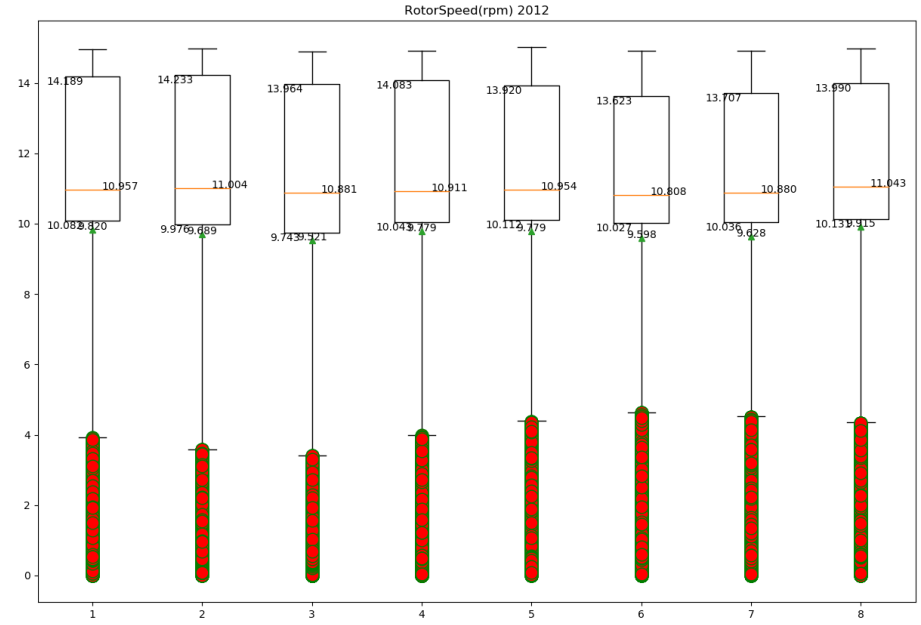

Figure 4.3: Rotor Speed in rpm for each turbine.

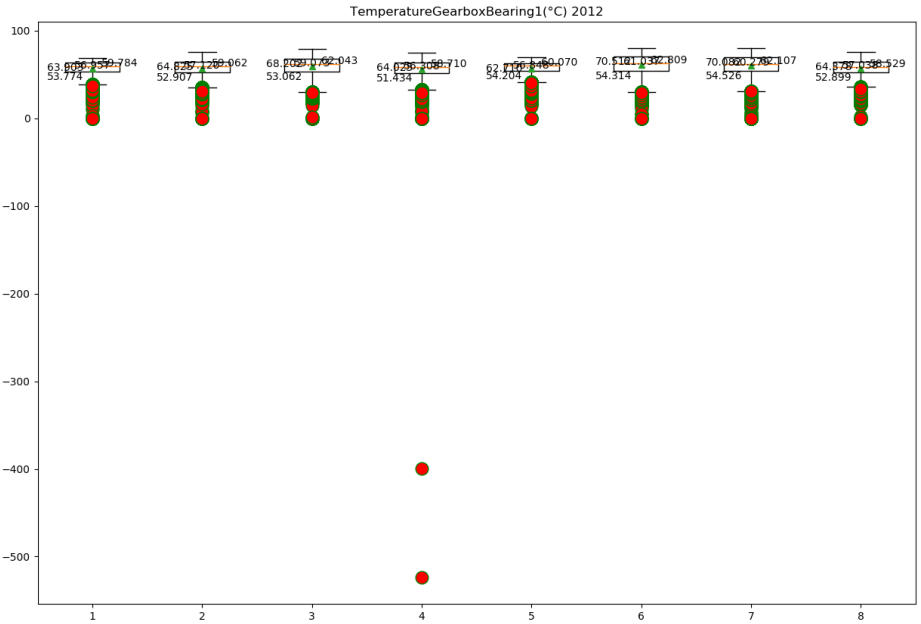

Figure 4.4: Gearbox Bearing Temperature in ºC for each turbine.

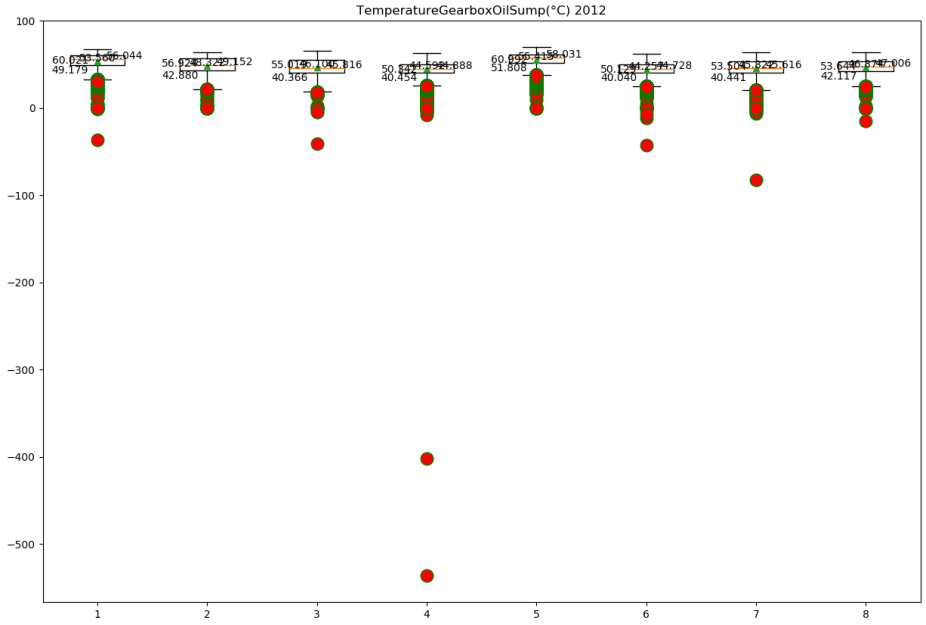

Figure 4.5: Gearbox Oil Sump Temperature in ºC for each turbine.

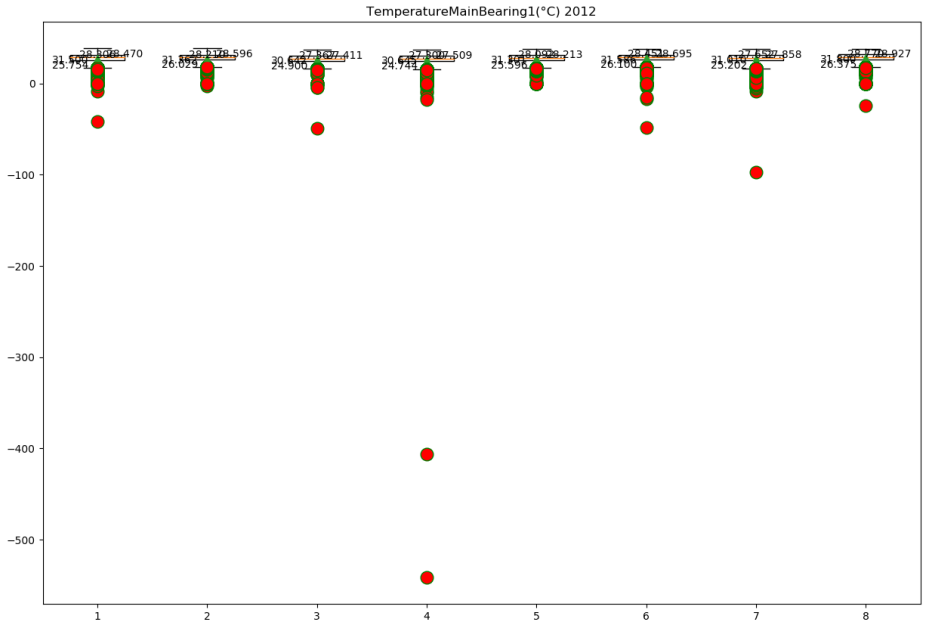

Figure 4.6: Main Bearing Temperature in ºC for each turbine.

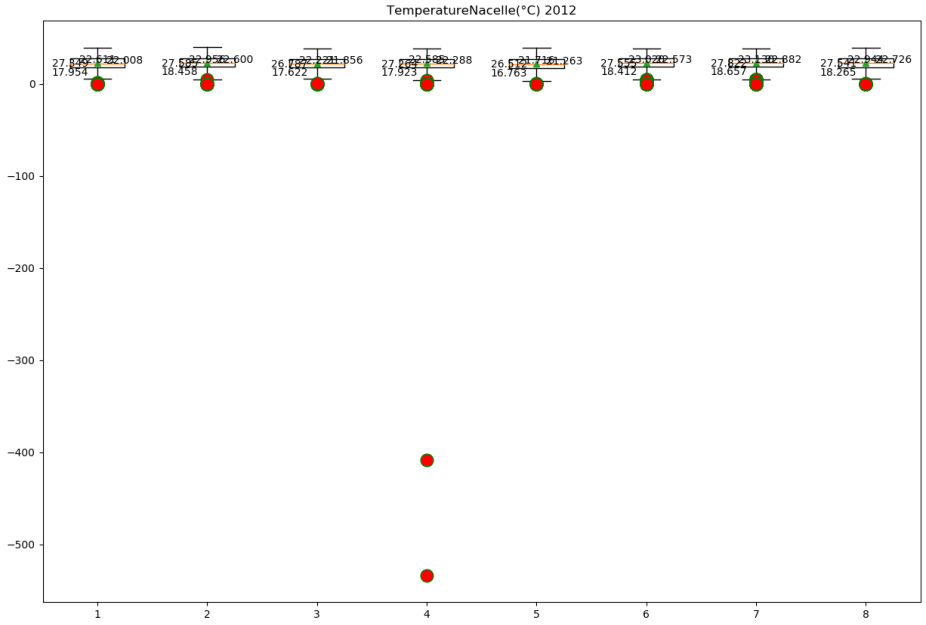

Figure 4.7: Nacelle Temperature in ºC for each turbine.

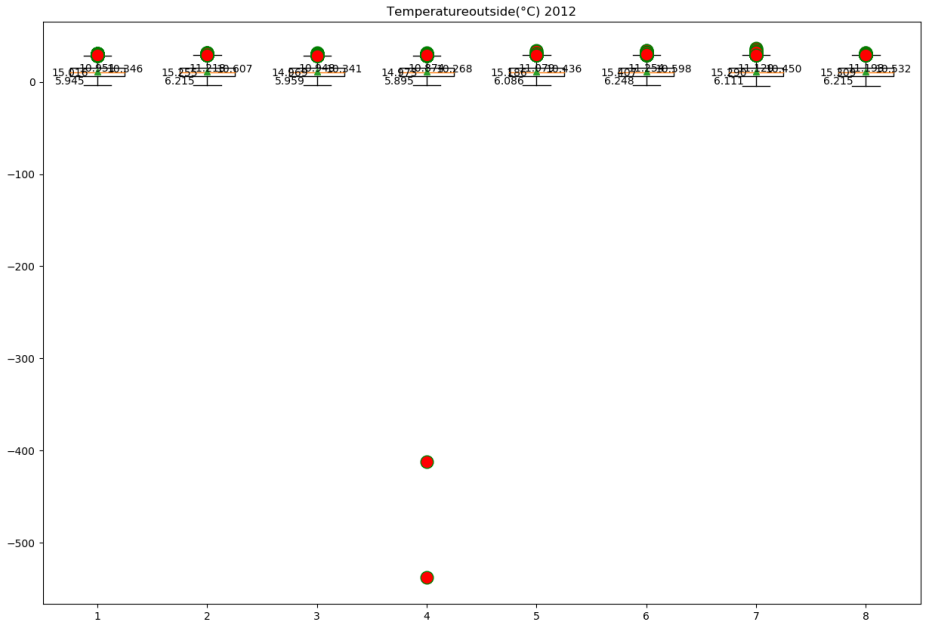

Figure 4.8: Outside Temperature in ºC for each turbine.

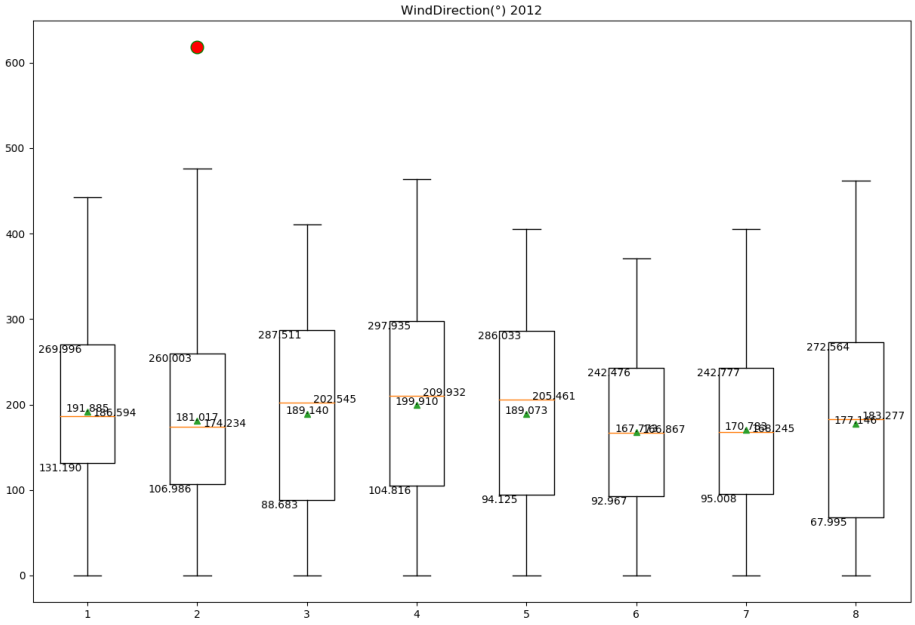

Figure 4.9: Wind Direction in º degrees for each turbine.

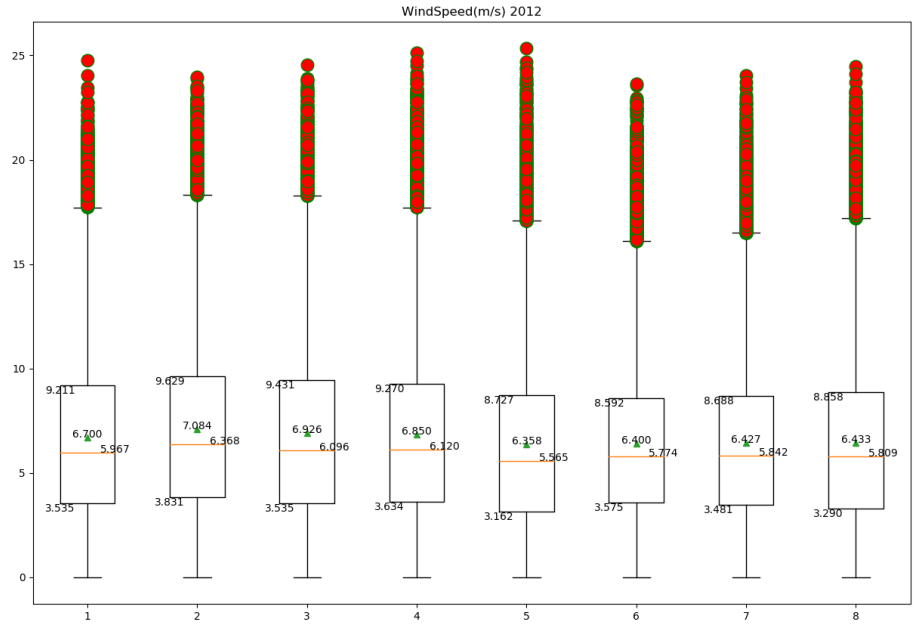

Figure 4.10: Wind Speed in m/s for each turbine.

The author in [8] carried out an extensive analysis to define acceptable intervals for the features and it can also be seen from the previous boxplots that some values do not make sense, especially the boxplots concerning the temperatures. These values are justified by the authors as measurement errors hence considered as outliers.

So in this work, outliers were removed according to the acceptable intervals, using *regex* expressions and filters from the *Python*® *pandas*[4] package. It is also important to mention that data imputation[5] was performed to handle the missing values present in the data. The method chosen was the *k*-NN since it can be much more accurate than the mean, median (depending on the data set), with the drawback of being computationally expensive due to the fact it stores the whole training data set in memory. The new boxplots can be seen in Figures 4.11, 4.12, 4.13, 4.14, 4.15, 4.16 and 4.17.

Special acceptable intervals were defined for the *RotorSpeed* and *WindSpeed* SCADA variables. The *RotorSpeed* values are established by the manufacturers of the turbines[6] and the *WindSpeed* values are defined according Betz's Law[7].

4.4 Analysis of the Results

Now that the data is treated, its analysis can be done more accurately. This is shown in greater detail in this Section.

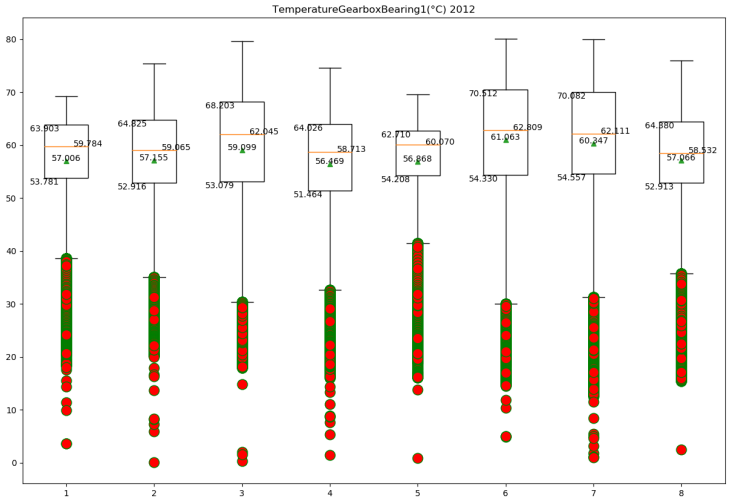

Figure 4.11: Gearbox Bearing Temperature in ºC for each turbine.

[4] https://pandas.pydata.org/about/index.html

[5] https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779

[6] https://www.iberwind.pt/pt/parques/freita-i/

[7] http://www.energiasrenovaveis.com/DetalheConceitos.asp?ID_conteudo=60&ID_area=3&ID_sub_area=6

As can be seen, there is a fluctuation of the Gearbox Bearing Temperature across all turbines. The temperature tends to be between 53-65 ºC except for turbines 3, 6 and 7. Despite this fluctuation, the median of the temperature is very similar for all turbines.

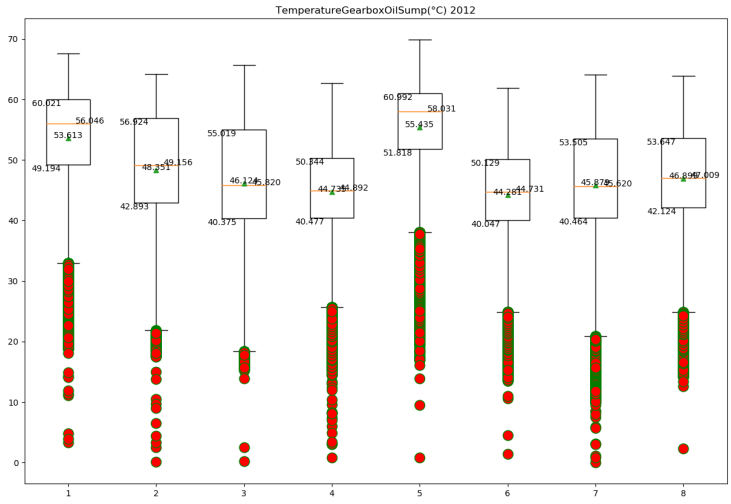

Figure 4.12: Gearbox Oil Sump Temperature in ºC for each turbine.

Turbine 5 stands out from the rest due to the fact its temperature is higher than the majority of the turbines. The rest of the turbines represent slight fluctuations regarding the temperature of the Gearbox Oil Sump, with turbines 4 and 6 having almost the same temperature interval, 7 and 8 and 3 and 7 having similar upper and lower temperature intervals respectively.

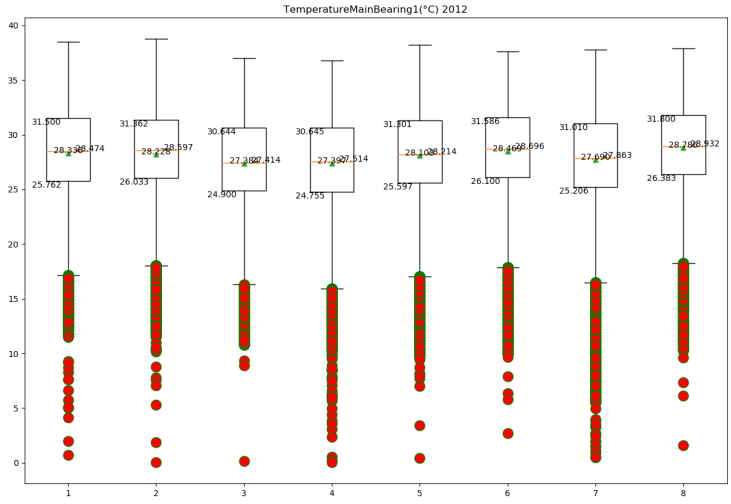

Figure 4.13: Main Bearing Temperature in ºC for each turbine.

The temperature of the Main Bearing seems to be almost identical for all the turbines with only minor fluctuations in the lower and upper intervals and the median. The temperature tends to be between 25-32 ºC.

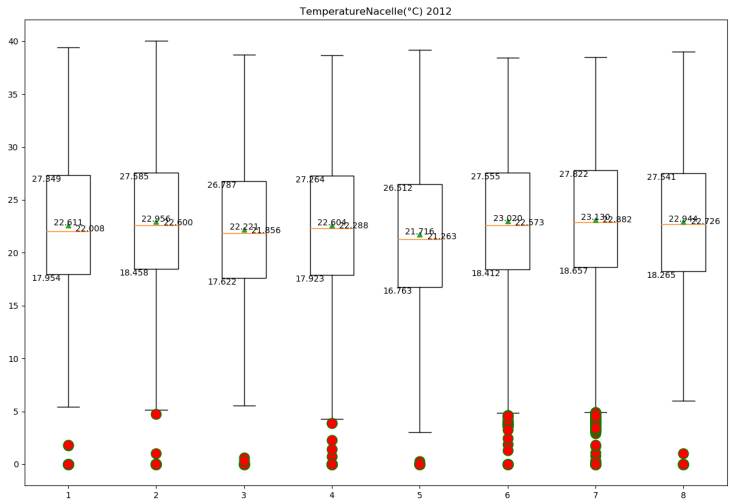

Figure 4.14: Nacelle Temperature in ºC for each turbine.

As can be seen, the temperature in the nacelle is also stable amongst the turbines fluctuating for the same measures as in Figure 4.13. The temperature in the nacelle tends to be between 17-27 ºC.

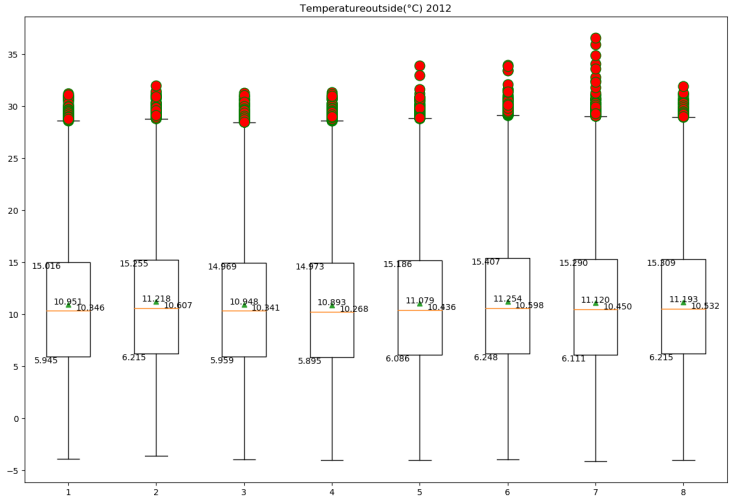

Figure 4.15: Outside Temperature in ºC for each turbine.

The temperature outside is the most stable of the temperatures showing almost the same lower and upper intervals and median for all the turbines. This is to be expected

since the temperature outside has nothing to do with the turbines and they are all on the same farm. The temperature outside is between 6-15 ºC.

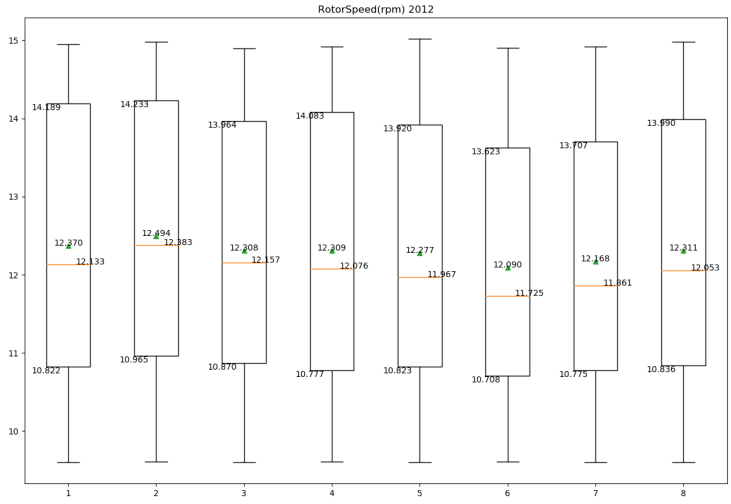

Figure 4.16: Rotor Speed in rpm for each turbine.

The turbines work on an average of 12 rpm and the rotor speed values tend to be between 11-14 rpm. There is only a little fluctuation in the median for turbines 2, 6 and 7. Overall the rotor speed is similar amongst all turbines.

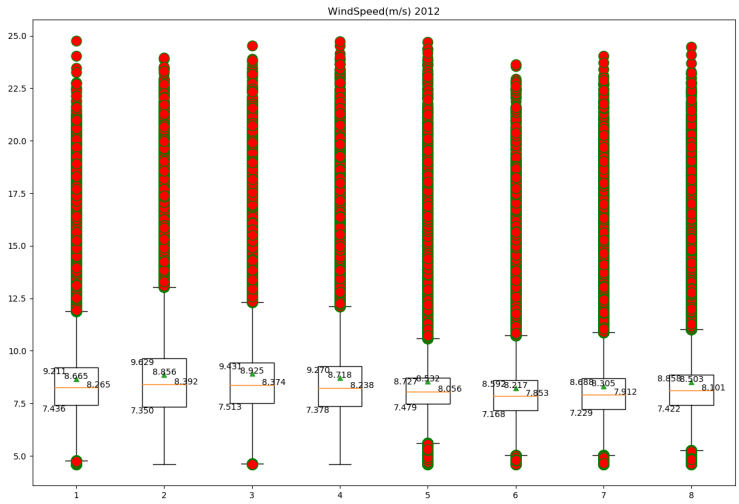

Figure 4.17: Wind Speed in m/s for each turbine.

Wind speed is very similar for all turbines which is natural since the turbines are all in the same wind farm. The median is practically the same and the wind speed tends to be between 7,5-9,6 m/s.

Summary

By doing this preliminary exploratory analysis it was possible to conclude: the turbines tend to follow the same behaviour regarding the temperature of the main bearing, nacelle and outside. The only concerning temperatures are the gearbox bearing and gearbox oil sump, especially in turbines 3, 6 and 7 for the first temperature and turbines 1 and 5 for the second one. These might indicate a slightly malfunctioning of these turbines. It is also important to mention that data referring to anomalous behaviour might not allow to observe its impact in the different variables mentioned above, given its relative dimension.

4.5 Preprocessing

The datasets used in the experimental study, after the refinement done in the preliminary stage were of two different types: (1) a dataset regarding the data of each turbine; (2) a dataset containing the data of all turbines.

Table 4.1 describes the datasets used in the unsupervised experiments. The datasets are sorted according the different groups mentioned in 4.1 because it would be interesting to see if the algorithms get affected by the amount of outliers present in the data. The datasets used for the semi-supervised algorithms will be described in Section 5.2.

The preprocessing performed to the data consisted in two tasks: (1) choosing the most relevant features of the turbines; (2) reduce/eliminate noisy data.

The path chosen to select the most relevant features was to calculate the Pearson correlation coefficient of the Active Power attribute against all others. Pearson correlation can be interpreted in the following manner: it is a coefficient with a range of values between -1 and 1, and a correlation value of 0.7 between two variables would indicate that a significant and positive relationship exists between the two. This means if a feature value goes up the other feature value will also go up and *vice-versa* for the negative case. After this process the most relevant features used in the experiments were: Active Power (kW), Wind speed (m/s), Rotor speed (rpm), Gearbox Main Bearing Temperature (T_{GB}) and Gearbox Oil Sump Temperature (T_{GO}).

To reduce noisy data, the median of the observations corresponding to a day of a turbine functioning time was calculated. This led to a drastic decrease in the number of observations in the data sets.

DataSet	N	d	ϕ(%)
WT8	1096	5	53.38
WT3	1096	5	42.06
WT7	1096	5	38.05
WT5	1089	5	20.57
WT6	1096	5	14.6
WT2	1070	5	13.27
WT1	1096	5	11.95
WT4	1081	5	11.75

DataSet	N	d	ϕ(%)
All	8720	5	26.0

Table 4.1: Description of the datasets used for the unsupervised algorithms. N is the number of instances, d is the number of attributes and ϕ is the contamination (percentage of anomalies).

Chapter 5

Experimental Study

In this Chapter, the entire experimental protocol is explained and the novel approach presented.

Section 5.2 describes the preprocessing done to the datasets used in the experiments, and previous works are used to serve as theoretical background to help the reader understand how AD algorithms performances and results are evaluated. Popular methods for evaluating classifier performance like, *Precision*, *Recall*, *Accuracy*, *AUC* and *F-measure*, are carefully explained in Section 5.2.2.

The last two Sections, 5.2.3 and 5.3, concern the methodologies adopted for the experiments of this dissertation, and the comparison and analysis of the different approaches results, respectively.

5.1 Main Goals of the Study

The main objective of this dissertation is to explore and adapt the application of unsupervised approaches, such as fuzzy clustering and archetypal analysis, as well as semi-supervised approaches on the automatic detection of wind turbine faults. Since archetypes are extremal points in the data, it was interesting to study how these kind of algorithms adapt to AD problems. At the end we intend to see what approaches are better at detecting anomalies, unsupervised or semi-supervised ones.

5.2 Setting of Experiments

The experiments conducted in this dissertation used as its backbone the algorithms studied and described in Sections 3.3 and 3.4. The algorithms are available in a public toolbox[1] called *PyOD*. *PyOD* [72] is a comprehensive and scalable *Python*® toolbox for AD with more than 30 detection algorithms and it is also well acknowledged by the machine learning community.

As mentioned in Section 4.5 two other groups of datasets were used in the semi-supervised experiments, one dataset per turbine and one dataset containing the data of all turbines, both split in train-test. The technique used to split the datasets into train-test sets was a stratified k-fold cross-validation technique, due to the fact the data being imbalanced. With this technique, it is possible to guarantee balanced proportions for the training and testing sets. For the unsupervised algorithms, no split technique was used.

Five folds were used in the splitting technique, hence the instances and outlier percentage values in the Table 5.1, are averages of the folds.

Data Set	N	d	ϕ(%)
WT8	876	5	50.0
WT3	876	5	42.01
WT7	876	5	38.01
WT5	871	5	20.55
WT6	876	5	14.61
WT2	855	5	13.22
WT1	876	5	11.87
WT4	864	5	11.69

Data Set	N	d	ϕ(%)
All	6970	5	25.25

Table 5.1: Description of the datasets after train-test split used for the semi-supervised algorithms. N is the number of instances, d is the number of attributes and ϕ is the contamination (percentage of anomalies).

When conducting experiments with semi-supervised algorithms, the classifier is trained only with normal data and the model is tested against normal and anomalous data. The number of instances present in the Table 5.1 above concern the training data.

The experiments were all executed in a machine with a *Intel*® *Core*™ i7-9700K CPU (8 cores) @ 3.60GHz-4.90GHz processor and 16GB of RAM. The entire code of this dissertation is written in *Python*® version 3.7 with the exception of the LoMST that is implemented in *R*® version 3.6.3.

[1] https://github.com/yzhao062/pyod

5.2.1 Data Normalization

The datasets suffered the same normalization process prior the execution of every algorithm. For every observation $\varkappa_i$ of the original dataset with d features, $d = 1, 2, \ldots, k$, the *mean* of each feature is subtracted to $\varkappa_i$, and divided by the difference of that feature maximum and minimum values. The final normalization process can be formalized as show in equation (5.1):

$$\varkappa_i^{'} = \frac{\varkappa_i - \mu(d_k)}{max(d_k) - min(d_k)} \tag{5.1}$$

This was the same normalization chosen as in [48].

5.2.2 Evaluation Measures

This Section presents some popular measures for assessing how "accurate" a classifier is at predicting the class label of observations [30]. Table 5.2 groups the majority of the evaluation measures, except AUC, and their respective formulas all together to serve as a consultation for the reader.

Measure	Formula
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{P}$
Accuracy	$\frac{TP+TN}{P+N}$
F-measure	$\frac{2 \cdot precision \cdot recall}{precision+recall}$

Table 5.2: Evaluation measures, where TP, TN, FP, FN, P, N correspond to the number of true positive, true negative, false positive, false negative, number of positive observations and number of negative observations respectively.

To make it easier to grasp the meaning of the evaluation measures, some nomenclature must be presented first:

- True Positive (TP): The number of positive observations that were correctly labelled as positive;
- True Negative (TN): The number of negative observations that were correctly labelled as negative;
- False Positive (FP): The number of negative observations incorrectly labelled as positive;
- False Negative (FN): The number of positive observations incorrectly labelled as negative.

An useful tool for analyzing when the classifier is labelling the observations correctly and labelling the observations incorrectly, is a confusion matrix presented in the Table 5.3 bellow. Ideally, one wants the classifier to have most of the entries in the left diagonal and the rest of the entries zero or close to zero, to have a good accuracy. Despite the confusion matrix shown represents a binary classification problem, it can be easily adapted for multiple classes.

		Predicted class		
		Positive	Negative	Total
Actual class	Positive	TP	FN	P
	Negative	FP	TN	N
	Total	P'	N'	$P+N$

Table 5.3: Generic confusion matrix.

The *accuracy* can be described as the percentage of correctly classified observations. By looking at a confusion matrix it is easy to see if a certain classifier is mislabelling between two classes. *Accuracy* is better when applied to balanced data. Analogously, one can think of the *error rate* of a given classifier as 1-*accuracy*, and can be calculated in equation (5.2):

$$error\ rate = \frac{FP+FN}{P+N} \tag{5.2}$$

When a certain problem presents imbalanced data, that is, the positive class represents the majority of observations while the negative class the minority, the classifier can correctly be labelling only the observations belonging to the negative class, and mislabelling all observations belonging to the positive class. In such problems makes sense to talk about two particular methods, *sensitivity* and *specificity*, shown in equations (5.3) and (5.4) respectively:

$$sensitivity = \frac{TP}{P} \tag{5.3}$$

$$specificity = \frac{TN}{N} \tag{5.4}$$

where P and N are the total number of positive and negative observations respectively.

The *sensitivity*, also known as *recall*, corresponds to the proportion of positive observations that are correctly classified, whereas *specificity* corresponds to the proportion of negative observations that are correctly classified. *Accuracy* can also be seen as a function of both *sensitivity* and *specificity*, expressed in equation (5.5):

$$accuracy = sensitivity \cdot \frac{P}{P+N} + specificity \cdot \frac{N}{P+N} \tag{5.5}$$

The *precision* and *recall* measures are also widely used to evaluate a classifier. *Precision* is the percentage of positive observations labelled as such, and *recall* is the percentage of actual positive observations labelled as such.

A perfect *precision* score of 1.0 for a certain class means that every observation labelled as belonging to that class, is correct. However, it is not possible to infer how many the remainder observations were mislabelled as belonging to that class. Usually, there is an inverse relationship between *precision* and *recall*, where it is possible the increase the value of one while the value of the other decreases. For this reason, *precision* values are often compared for a fixed value of *recall*, and vice-versa.

An alternative way to use *precision* and *recall* together is to combine both into a single measure called *F-measure*, also called F_1*-score*. The *F-measure* can be interpreted as the harmonic mean of *precision* and *recall*. There's a variant called the F_β measure defined in equation (5.6):

$$F_\beta = \frac{(1+\beta^2)\cdot precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{5.6}$$

where, β is a non-negative real number. While *F-measure* assigns equal weight to precision and recall, the F_β assigns β times as much weight to recall as to precision. F_2 means that recall is weighted twice as much as precision, and $F_{0.5}$ means that precision is weighted twice as much as recall.

For binary classification problems, a ROC cures helps to analyze the trade-off, of a certain classifier, between the rate of correctly classified positive observations *vs* the rate of negative observations being wrongly classified as positives for different portions of the data. The area under the ROC curve, AUC, is a measure of the accuracy of a classifier. To better understand the AUC measure, first, the ROC curve must be briefly explained.

The classification returned by the model comprises a list of labels, and that list is sorted in a such a way that an observation most likely to belong to the positive class appears at the top of the list, and an observation least likely to belong to the positive class will be at the bottom at the list.

The vertical axis of a ROC curve represents the TPR and the horizontal axis the FPR. Starting from the bottom left corner, where both TPR and FPR are equal to 0, the observation actual class label is checked at the top of the previous list. If it is a TP, TPR increases and this corresponds to move up and plot a point on the graph. If instead the observation is wrongly classified as positive, leading to an increase of both FP and FPR, in this case moving right and plot a point (Figure 5.1). This is an iterative process for the entire list.

In summary, the *accuracy* measure works best when the data classes are balanced, while *sensitivity* (or *recall*), *specificity*, *precision*, F, and F_β, are better suited to the class imbalance problem, where the main class of interest (positive) is rare. To assess the accuracy of a classifier, one can use the AUC measure. The closer the ROC curve is to the diagonal, AUC equal to 0.5, the less accurate the classifier is and the closer it is to top left corner, TPR equal to 1, the more accurate the classifier is. A classifier with perfect accuracy has an AUC of 1.0.

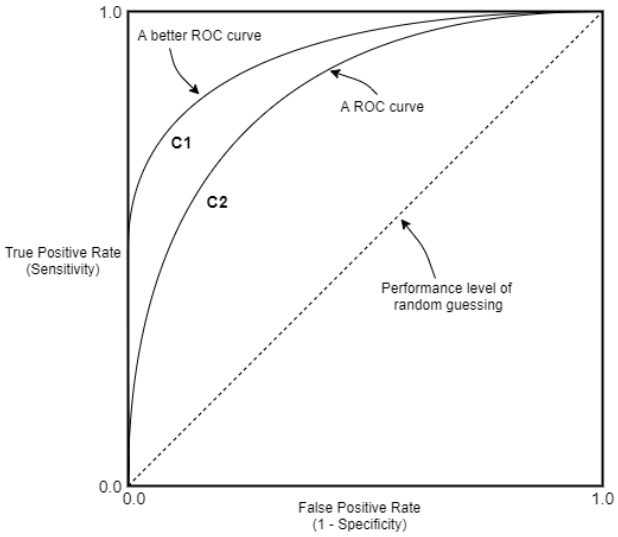

Figure 5.1: The ROC curves for two classifiers, C_1 and C_2, where the diagonal line represents the equal probability of a classifier to label an observation as positive or negative.

5.2.3 Experimental Protocol

Comparing the ROC and Precision-Recall (PR) curves are robust ways to evaluate the performance of AD algorithms [15, 20, 28].

When it comes to fine-tuning the algorithms, the authors in [20, 28] calculated the *mean* and *standard deviation* of both metrics, AUC and APR, of each algorithm for each data set and decided that the best hyperparameters were the ones which increased the Mean Average Precision (MAP). Since real-world data sets when it comes to AD are heavily imbalanced and the positive class (anomalies) are more interesting than the negative class, the fine-tuning was performed based on maximizing the APR of each algorithm for each data set.

Besides using the two well known state of the art AUC and APR, the authors in [15] also used an interesting evaluation metric called precision at n (denominated $P@n$), defined as the proportion of correct results in the top n ranks [18]. This top n ranks is retrieved from the **full ranking** of objects returned by each method, scoring every object based on its outlierness. $P@n$ can be formalized in equation (5.7), where O is the ground truth of outliers in the data set with N instances:

$$P@n = \frac{|\{o \in O \mid rank(o) \leq n\}|}{n} \tag{5.7}$$

When using $P@n$ it is uncertain how to fairly choose the parameter n. The most common case is to set n equal to the number of outliers in the ground truth, $n = |O|$.

The authors stated that the values for $P@n$ tend to be considerably lower for data sets with smaller proportions of outliers, and *vice-versa*. The $P@n$ measure can be helpful in discriminating between methods that perform more or less equally well in terms of AUC and APR.

The work conducted in this dissertation followed the approaches briefly mentioned above since they seem adequate and fit our problem well.

5.2.3.1 Semi-supervised Techniques

Each semi-supervised algorithm was fine-tuned for its major hyperparameter using a *GridSearch*[2] approach. This approach is really expensive in terms of computation power and time, but it is the most efficient, since each combination of hyperparameter value is tried. The objective was to find the value for the hyperparameters that maximized the APR of each algorithm for each dataset (turbine).

The algorithms LOF, COF, k-NN and SOD got their major hyperparameter "*n_neighbours*" (k) varied between 10 and 50, while CBLOF got its hyperparameter "*n_clusters*" (c) varied between 10 and 30 and HBOS got its hyperparameter "*n_bins*" (b) varied between 20 and 40. These values highly depend on the data so they must be tested empirically to gain sensitivity. Figures 5.2, 5.3 and 5.4, 5.5 show the average of 10 runs for the values of AUC and APR for the dataset with the lowest and highest percentage of outliers respectively.

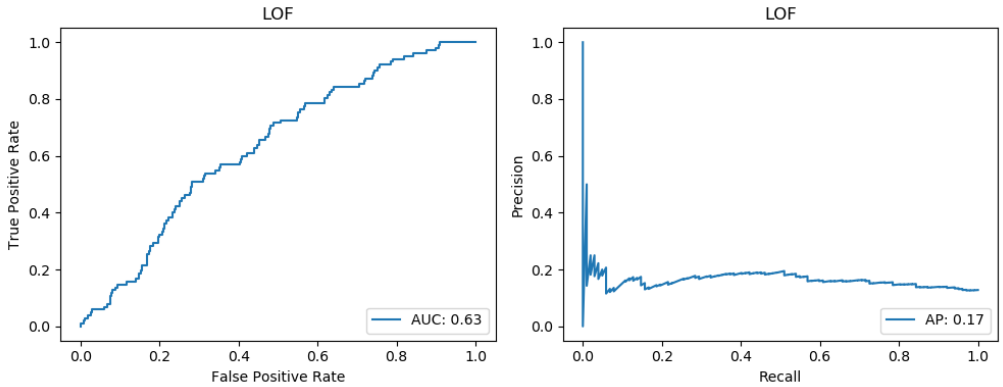

Figure 5.2: Average of AUC (left) and APR (right) values for turbine 4 in the training set.

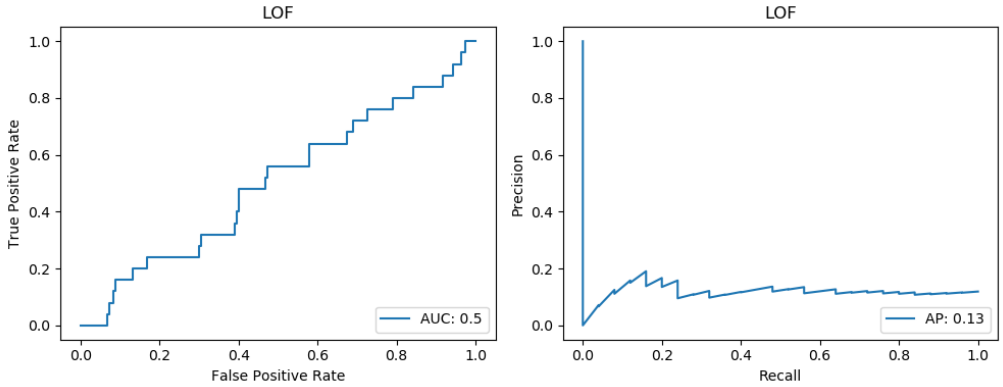

Figure 5.3: Average of AUC (left) and APR (right) values for turbine 4 in the test set.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

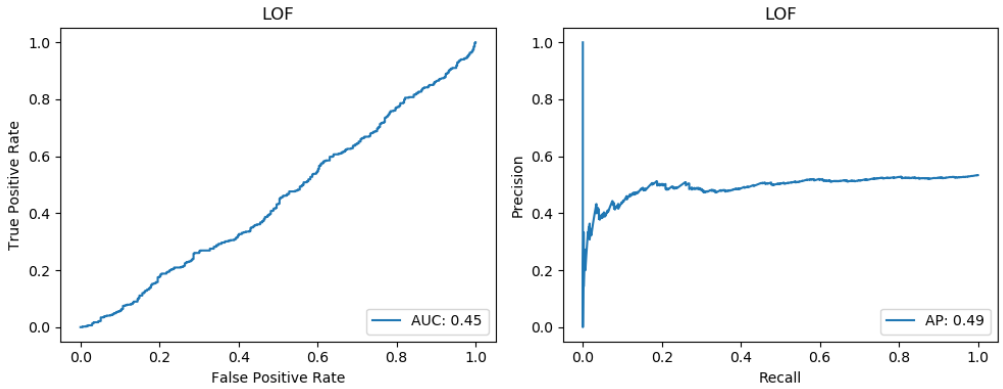

Figure 5.4: Average of AUC (left) and APR (right) values for turbine 8 in the training set.

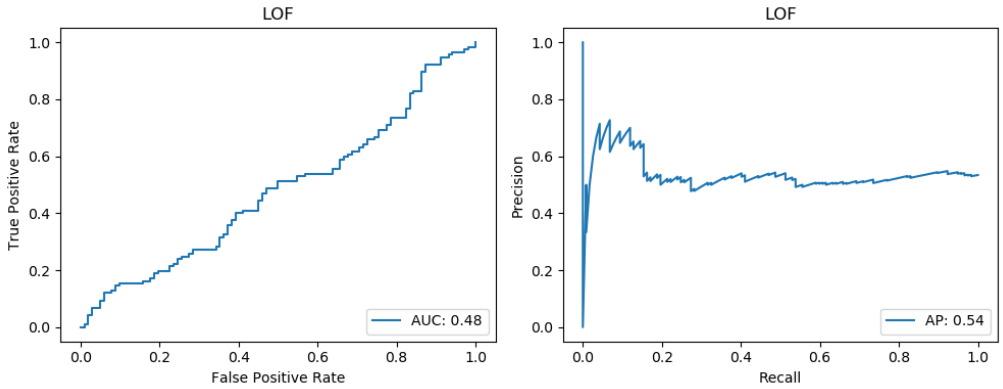

Figure 5.5: Average of AUC (left) and APR (right) values for turbine 8 in the test set.

5.2.3.2 Unsupervised Techniques

The FCM and AA algorithms were fine-tuned for their major hyperparameter, c and *arch* that denote the number of clusters and archetypes respectively. Finally, LoMST was fine-tuned for its major hyperparameter k that represents the neighbourhood to consider. The hyperparameter tuning for FCM, AA and LoMST was different due to the fact these algorithms were not included in the toolbox mentioned previously, therefore *GridSearch* was not possible. For FCM the fine-tuning consisted in varying the number of clusters c between 2 and 50, for AA *arch* was varied between 2 and 5, and for LoMST a range between 1 and 100 was used for parameter k as recommended and done by the authors [5]. These number of archetypes are highly dependent of the data so they must be tested empirically to gain sensitivity. Figures 5.6 and 5.7 show the average of 10 runs for the values of AUC and APR for the dataset with the lowest and highest percentage of outliers respectively.

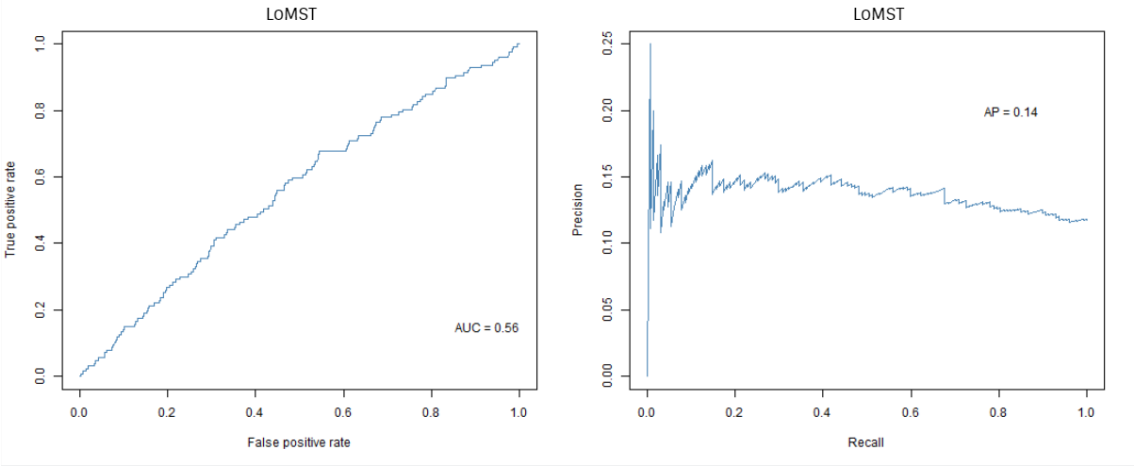

Figure 5.6: Average of AUC (left) and APR (right) values for turbine 4.

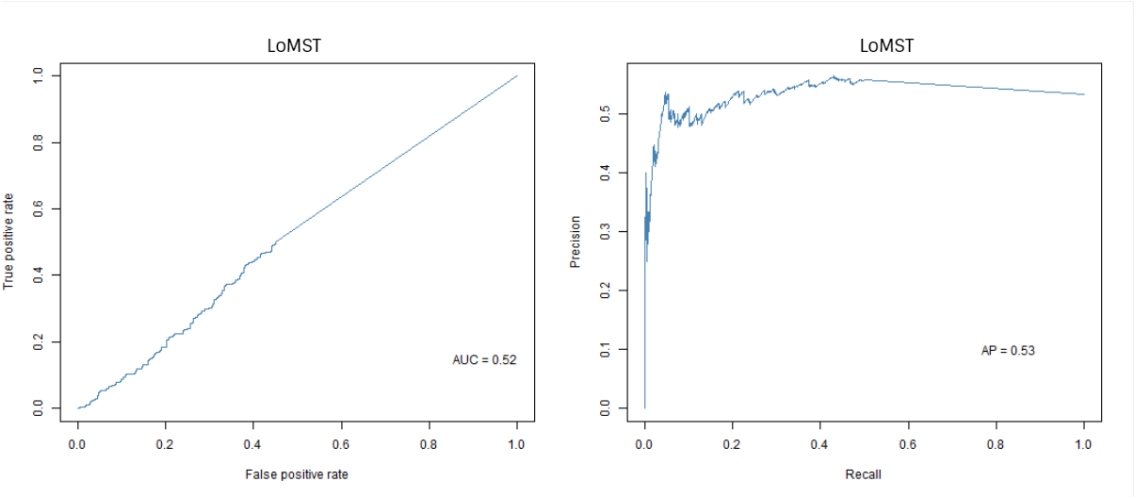

Figure 5.7: Average of AUC (left) and APR (right) values for turbine 8.

5.3 Results and Discussion

Now that the entire experimental work has been thoroughly described, in this Section the results of the experiments are presented and further analysed. The results comprise the values for the following evaluation measures: *execution time* (to measure the *efficiency*), *ROC AUC*, *P@n*, *Precision*, *Recall*, *Accuracy* and *F-measure* (to measure the *effectiveness*). For the sake of table representation, Fuzzy *c*-Means Modified Mahalanobis Taguchi System (FCM-MMTS) will simply be denoted as FCM, and Archetypal Analysis Modified Mahalanobis Taguchi System (AA-MMTS) as AA.

When running the experiments, it was possible to conclude that the algorithms didn't benefit, in the majority, from the datasets being all gathered as one. For this reason, the results correspond to the experiments when running with datasets regarding each turbine

separately. It is also presented in Table 5.4, the best parameters of each algorithm for each turbine, as a reference for future works.

Data	LOF	COF	KNN	AvgKNN	MedKNN	CBLOF	HBOS	SOD	AA	FCM	LoMST
WT8	k=50	k=50	k=50	k=50	k=50	c=10	b=22	k=16	a=5	c=40	k=1
WT3	k=50	k=50	k=49	k=50	k=47	c=19	b=20	k=38	a=5	c=38	k=1
WT7	k=18	k=48	k=36	k=50	k=50	c=24	b=35	k=11	a=2	c=50	k=2
WT5	k=10	k=10	k=10	k=12	k=13	c=22	b=22	k=13	a=5	c=50	k=12
WT6	k=50	k=50	k=25	k=41	k=44	c=23	b=21	k=33	a=4	c=50	k=27
WT2	k=29	k=10	k=23	k=23	k=13	c=18	b=23	k=47	a=4	c=50	k=19
WT1	k=50	k=16	k=32	k=50	k=49	c=19	b=31	k=14	a=2	c=7	k=64
WT4	k=15	k=23	k=41	k=50	k=39	c=12	b=20	k=30	a=5	c=50	k=9

Table 5.4: Best parameterization, where k is the number of neighbours, c the number of clusters, b the number of bins and a the number of archetypes.

5.3.1 Semi-supervised

Results for the semi-supervised experiments are presented in Tables 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11 and 5.12. For these experiments each metric was calculated for both the train and test sets, and the values presented are the average of the 5 folds.

The cells with "*" represent the best value of that metric in the train set for that turbine, and the cells with "**" represent the best value of that metric in the test set for that turbine. For these experiments all the algorithms got really low values, and most of the time 0, for the *standard deviation* hence, it is not presented to not make the tables too dense and difficult to read.

Execution Time

Data	N	ϕ	LOF	COF	KNN	AvgKNN	MedKNN	CBLOF	HBOS	SOD	Best(↓)
WT8	876	53	0.02	6.62	0.11	0.14	0.23	0.04	0.01*	1.17	0.01
WT3	876	42.01	0.02	6.85	0.12	0.14	0.23	0.07	0.01*	1.09	0.01
WT7	876	38.01	0.01*	6.76	0.11	0.14	0.23	0.08	0.01*	0.93	0.01
WT5	871	20.55	0.01*	3.04	0.1	0.13	0.21	0.07	0.01*	0.99	0.01
WT6	876	14.61	0.02	7.6	0.11	0.14	0.23	0.08	0.01*	1.23	0.01
WT2	855	13.22	0.02	3.07	0.11	0.13	0.21	0.07	0.01*	1.18	0.01
WT1	876	11.87	0.02*	3.9	0.11	0.14	0.23	0.1	0.02*	1.12	0.02
WT4	864	11.69	0.01*	4.48	0.11	0.14	0.22	0.06	0.01*	1.2	0.01

Table 5.5: Average execution times (seconds) for 10 runs.

When it comes to assess the *efficiency* we can clearly see that LOF and HBOS are the most efficient with the lowest times of execution. The COF algorithm is the least efficient, presenting the highest values of execution, and is the only algorithm that seems to benefit from datasets with low percentage of outliers.

Area Under Curve

Data	N	ϕ	LOF	COF	KNN	AvgKNN	MedKNN	CBLOF	HBOS	SOD	Best(↑)
WT8	876	53	0.46/0.48**	0.42/0.43	0.44/0.44	0.43/0.43	0.43/0.43	0.47*/0.45	0.45/0.43	0.45/0.44	0.47/0.48
WT3	876	42.01	0.51/0.51**	0.47/0.51**	0.52/0.51**	0.52/0.5	0.51/0.49	0.53*/0.49	0.52/0.51**	0.5/0.49	0.53/0.51
WT7	876	38.01	0.53/0.55**	0.49/0.54	0.52/0.54	0.53/0.55**	0.53/0.55**	0.54*/0.54	0.54*/0.55**	0.51/0.54	0.54/0.55
WT5	871	20.55	0.56*/0.53**	0.52/0.5	0.47/0.49	0.5/0.5	0.49/0.5	0.5/0.49	0.46/0.5	0.5/0.46	0.56/0.53
WT6	876	14.61	0.52*/0.57**	0.5/0.48	0.48/0.55	0.49/0.55	0.48/0.55	0.51/0.56	0.47/0.54	0.49/0.53	0.52/0.57
WT2	855	13.22	0.58/0.56	0.55/0.57**	0.57/0.54	0.59*/0.55	0.59*/0.55	0.58/0.56	0.53/0.54	0.55/0.53	0.59/0.57
WT1	876	11.87	0.61*/0.63	0.49/0.68**	0.56/0.58	0.57/0.58	0.57/0.58	0.58/0.6	0.46/0.5	0.53/0.63	061/0.68
WT4	864	11.69	0.62*/0.51	0.54/0.6**	0.54/0.54	0.56/0.54	0.55/0.54	0.55/0.52	0.54/0.53	0.53/0.49	0.62/0.6

Table 5.6: Average of AUC values for 10 runs.

The AUC can be interpreted in the AD domain as the probability of an AD algorithm to assign a random normal observation a lower score than a random anomalous observation. We can see that the nearest-based algorithms show better results for AUC when they "become" more local (small values of k) with LOF achieving the best results. In a general way, the values seem better for datasets with low percentage of outliers.

Average Precision

Data	N	ϕ	LOF	COF	KNN	AvgKNN	MedKNN	CBLOF	HBOS	SOD	Best(↑)
WT8	876	53	0.49/0.53**	0.47/0.5	0.52*/0.51	0.5/0.5	0.49/0.5	0.5/0.51	0.49/0.5	0.5/0.5	0.52/0.53
WT3	876	42.01	0.44/0.45	0.4/0.45	0.44/0.45	0.44/0.44	0.43/0.44	0.45*/0.45	0.45*/0.46**	0.42/0.43	0.45/0.46
WT7	876	38.01	0.41/0.44	0.38/0.45**	0.4/0.42	0.41/0.42	0.4/0.42	0.42*/0.42	0.42*/0.43	0.4/0.43	0.42/0.45
WT5	871	20.55	0.23*/0.26**	0.22/0.22	0.2/0.23	0.21/0.22	0.21/0.22	0.21/0.23	0.21/0.23	0.21/0.21	0.23/0.26
WT6	876	14.61	0.18*/0.28**	0.16/0.17	0.15/0.2	0.16/0.2	0.15/0.21	0.16/0.21	0.14/0.17	0.14/0.17	0.18/0.28
WT2	855	13.22	0.21/0.24	0.18/0.24	0.21/0.24	0.22/0.24	0.23*/0.25**	0.22/0.25**	0.2/0.2	0.17/0.19	0.23/0.25
WT1	876	11.87	0.17/0.2	0.12/0.29**	0.17/0.2	0.18*/0.2	0.18*/0.21	0.18*/0.21	0.11/0.16	0.14/0.25	0.18/0.29
WT4	864	11.69	0.16*/0.12	0.13/0.19**	0.15/0.17	0.15/0.17	0.15/0.17	0.15/0.16	0.14/0.16	0.13/0.17	0.16/0.19

Table 5.7: Average of APR values for 10 runs.

One of the drawbacks of AUC might be that it is not the best for imbalanced datasets, and APR can possibly better emphasize small detection performance changes. With APR being the area under the PR curve, it shows how *precision* and *recall* trade against one another. In an ideal world one would want the AD algorithms to identify *all* and *only* anomalies. It is clear that the best values of APR are in the datasets with more percentage of outliers and where the nearest-based algorithms are more global. All algorithms got their best value for this metric in the top most outliers datasets.

Precision at N

Data	N	ϕ	LOF	COF	KNN	AvgKNN	MedKNN	CBLOF	HBOS	SOD	Best(↑)
WT8	876	53	0.51*/0.52**	0.49/0.47	0.48/0.48	0.47/0.48	0.48/0.47	0.51*/0.49	0.5/0.48	0.5/0.49	0.51/0.52
WT3	876	42.01	0.42/0.43**	0.39/0.42	0.45*/0.42	0.44/0.41	0.43/0.4	0.45*/0.41	0.43/0.43**	0.44/0.41	0.45/0.43
WT7	876	38.01	0.41/0.45**	0.36/0.41	0.41/0.41	0.42*/0.41	0.41/0.41	0.42*/0.4	0.41/0.4	0.39/0.42	0.42/0.45
WT5	871	20.55	0.24*/0.25**	0.24*/0.24	0.19/0.2	0.22/0.21	0.2/0.21	0.19/0.2	0.19/0.18	0.21/0.18	0.24/0.25
WT6	876	14.61	0.17*/0.2**	0.14/0.14	0.15/0.18	0.16/0.17	0.15/0.18	0.16/0.2**	0.15/0.16	0.12/0.13	0.17/0.2
WT2	855	13.22	0.2/0.19	0.19/0.19	0.19/0.2	0.22*/0.19	0.22*/0.2	0.22*/0.21**	0.2/0.18	0.17/0.16	0.22/0.21
WT1	876	11.87	0.15/0.15	0.1/0.29**	0.23*/0.18	0.22/0.18	0.23*/0.19	0.2/0.19	0.07/0.16	0.13/0.27	0.23/0.29
WT4	864	11.69	0.16*/0.09	0.12/0.19**	0.14/0.17	0.16*/0.16	0.14/0.16	0.16*/0.14	0.15/0.12	0.15/0.17	0.16/0.19

Table 5.8: Average of *P@n* values for 10 runs.

Since the top n rank is equal to the set of all anomalies, *P@n* behaves in a similar way to *recall*. The values of this metric are really low for the datasets with few outliers and high for the datasets with high number of outliers as expected. The LOF is the winner for this metric but CBLOF has similar results.

Precision

Data	N	ϕ	LOF	COF	KNN	AvgKNN	MedKNN	CBLOF	HBOS	SOD	Best(↑)
WT8	876	53	0.51/0.53**	0.49/0.49	0.48/0.49	0.47/0.49	0.48/0.48	0.51/0.51	0.52*/0.5	0.48/0.5	0.52/0.53
WT3	876	42.01	0.42/0.42	0.39/0.42	0.44*/0.43**	0.44*/0.4	0.43/0.4	0.44*/0.41	0.43/0.4	0.44*/0.42	0.44/0.43
WT7	876	38.01	0.41/0.39	0.37/0.41	0.41/0.4	0.42*/0.4	0.41/0.41	0.42*/0.39	0.41/0.4	0.4/0.42**	0.42/0.42
WT5	871	20.55	0.25*/0.21	0.24/0.2	0.19/0.22**	0.21/0.22**	0.2/0.21	0.19/0.21	0.19/0.22**	0.2/0.19	0.25/0.22
WT6	876	14.61	0.17*/0.17**	0.15/0.14	0.15/0.16	0.16/0.16	0.15/0.17**	0.16/0.16	0.15/0.17**	0.12/0.15	0.17/0.17
WT2	855	13.22	0.2/0.16	0.19/0.17**	0.19/0.17**	0.22/0.14	0.23*/0.13	0.21/0.14	0.2/0.15	0.17/0.16	0.23/0.17
WT1	876	11.87	0.15/0.16	0.1/0.26**	0.23*/0.19	0.22/0.18	0.23*/0.18	0.19/0.18	0.07/0.12	0.13/0.22	0.23/0.26
WT4	864	11.69	0.15/0.12	0.11/0.17**	0.14/0.12	0.16*/0.11	0.15/0.11	0.16*/0.12	0.15/0.12	0.14/0.13	0.16/0.17

Table 5.9: Average of *precision* values for 10 runs.

Precision is the percentage of correctly identified anomalies that are actually anomalies. We want AD algorithms with good values of *precision* because this means our models will have a low FAR. For this metric the best values go to LOF, CBLOF and HBOS. For the top most outliers datasets the values of *precision* are better.

Recall

Data	N	ϕ	LOF	COF	KNN	AvgKNN	MedKNN	CBLOF	HBOS	SOD	Best(↑)
WT8	876	53	0.48/0.61	0.5/0.54	0.44/0.53	0.45/0.6	0.45/0.59	0.55/0.7	0.64*/0.73**	0.31/0.56	0.64/0.73
WT3	876	42.01	0.41/0.58	0.39/0.42	0.45/0.58	0.44/0.62	0.42/0.63	0.5*/0.73**	0.46/0.67	0.33/0.57	0.5/0.73
WT7	876	38.01	0.4/0.69	0.37/0.49	0.4/0.57	0.4/0.6	0.39/0.6	0.46*/0.74**	0.44/0.61	0.32/0.58	0.46/0.74
WT5	871	20.55	0.23*/0.75**	0.23*/0.19	0.17/0.48	0.18/0.56	0.18/0.54	0.2/0.53	0.18/0.45	0.18/0.34	0.23/0.75
WT6	876	14.61	0.15/0.39	0.14/0.21	0.14/0.39	0.15/0.42	0.14/0.41	0.16*/0.48**	0.14/0.45	0.1/0.31	0.16/0.48
WT2	855	13.22	0.19/0.47	0.2/0.24	0.18/0.39	0.2/0.46	0.21/0.48	0.22*/0.47	0.2/0.54**	0.18/0.37	0.22/0.54
WT1	876	11.87	0.16/0.45	0.1/0.37	0.25*/0.36	0.23/0.39	0.24/0.35	0.21/0.46**	0.08/0.48	0.12/0.44	0.25/0.46
WT4	864	11.69	0.14/0.52**	0.11/0.21	0.14/0.26	0.15/0.33	0.13/0.32	0.16*/0.27	0.15/0.42	0.14/0.32	0.16/0.52

Table 5.10: Average of *recall* values for 10 runs.

Recall is the percentage from the full set of anomalies. In the AD domain one could argue that we want our algorithms to detect anomalies when they occur. An AD algorithm that has a high number of TN can lead to severe problems in a long-term basis. The CBLOF got consistently good results, > 0.7, for the datasets with more outliers, and and LOF and CBLOF also achieved some interesting results. In a general way, all algorithms got better results for the test sets.

Accuracy

Data	N	ϕ	LOF	COF	KNN	AvgKNN	MedKNN	CBLOF	HBOS	SOD	Best(↑)
WT8	876	53	0.48*/0.5**	0.45/0.45	0.45/0.45	0.44/0.45	0.44/0.44	0.48*/0.49	0.49/0.47	0.46/0.46	0.48/0.5
WT3	876	42.01	0.51/0.5	0.49/0.51**	0.53/0.51**	0.53/0.46	0.52/0.46	0.52/0.45	0.51/0.45	0.54*/0.48	0.54/0.51
WT7	876	38.01	0.55/0.47	0.51/0.55**	0.55/0.52	0.56*/0.51	0.55/0.51	0.55/0.46	0.55/0.5	0.56*/0.54	0.56/0.55
WT5	871	20.55	0.7*/0.36	0.69/0.69**	0.68/0.5	0.69/0.46	0.68/0.47	0.66/0.48	0.67/0.53	0.68/0.56	0.7/0.69
WT6	876	14.61	0.77*/0.68**	0.75/0.68**	0.76/0.63	0.76/0.6	0.76/0.62	0.75/0.57	0.76/0.6	0.76/0.64	0.77/0.78
WT2	855	13.22	0.79/0.62	0.78/0.78**	0.79/0.66	0.8/0.57	0.8*/0.55	0.79/0.59	0.78/0.52	0.78/0.65	0.8/0.78
WT1	876	11.87	0.8/0.66	0.79/0.81**	0.81/0.71	0.81/0.66	0.82*/0.69	0.8/0.63	0.77/0.52	0.8/0.74	0.82/0.81
WT4	864	11.69	0.81*/0.53	0.79/0.78**	0.8/0.7	0.81*/0.66	0.81*/0.66	0.8/0.69	0.8/0.58	0.8/0.67	0.81/0.78

Table 5.11: Average of *accuracy* values for 10 runs.

It is advisable to use *precision* and *recall* too and not *accuracy* alone, because sometimes the *accuracy* can be very high but the *precision* or *recall* are low. In AD and our problem in particular, where turbines with undetected faults can lead to substantial losses, we want our models to avoid the situation of a turbine having an anomaly but not being classified as one. All algorithms got really high values of accuracy, around 0.8, for the datasets with few outliers where for these same datasets the algorithms got really low values of *precision* and *recall*. *Accuracy* alone is not the solution.

F-measure

Data	N	ϕ	LOF	COF	KNN	AvgKNN	MedKNN	CBLOF	HBOS	SOD	Best(↑)
WT8	876	53	0.49/0.57	0.49/0.5	0.46/0.51	0.46/0.54	0.47/0.53	0.53/0.59**	0.57*/0.59**	0.38/0.52	0.57/0.59
WT3	876	42.01	0.41/0.48	0.39/0.42	0.45/0.49	0.44/0.49	0.42/0.48	0.47*/0.52**	0.44/0.5	0.38/0.48	0.47/0.52
WT7	876	38.01	0.41/0.49	0.37/0.44	0.4/0.47	0.41/0.48	0.4/0.48	0.44*/0.51**	0.43/0.48	0.36/0.49	0.44/0.51
WT5	871	20.55	0.24*/0.32**	0.24*/0.19	0.18/0.28	0.19/0.3	0.19/0.3	0.2/0.29	0.19/0.27	0.19/0.24	0.24/0.32
WT6	876	14.61	0.16*/0.23**	0.15/0.16	0.15/0.21	0.15/0.22	0.15/0.22	0.16*/0.23**	0.15/0.24	0.11/0.21	0.16/0.23
WT2	855	13.22	0.2/0.24**	0.19/0.2	0.19/0.23	0.21/0.2	0.22*/0.2	0.22/0.21	0.2/0.23	0.18/0.21	0.22/0.24
WT1	876	11.87	0.16/0.23	0.1/0.3**	0.24*/0.24	0.23/0.23	0.24*/0.23	0.2/0.24	0.07/0.19	0.12/0.29	0.24/0.3
WT4	864	11.69	0.14/0.19**	0.11/0.19**	0.14/0.15	0.15*/0.16	0.13/0.16	0.15*/0.16	0.15*/0.18	0.14/0.19**	0.15/0.19

Table 5.12: Average of *F1* values for 10 runs.

This metric is really important in our problem and for AD in general for two reasons: (1) is good for imbalanced data; (2) is the trade-off between *precision* and *recall*, two metrics we want our models to have. Instead of balancing *precision* and *recall*, we can aim for a good *F1* score which indicates a good *precision* and *recall*. The algorithms that achieved the best *F1* values are again the LOF, CBLOF and HBOS with values close to 0.6 for the dataset with more outliers.

5.3.2 Unsupervised

Results for the unsupervised experiments are presented in Tables 5.13, 5.14, 5.15, 5.16, 5.17, 5.18, 5.19 and 5.20. When running unsupervised techniques there is no split of the data into train/test sets therefore, the values with "*" simply represent the best value of that metric for that turbine. It is also shown the value of the *standard deviation* in parentheses, "()".

Execution Time

Data	N	ϕ	AA	FCM	LoMST	Best(↓)
WT8	1096	53.38	5.13	0.37*	8.81	0.37
WT3	1096	42.06	3.71	0.35*	9.16	0.35
WT7	1096	38.05	2.97	0.48*	9.79	0.48
WT5	1089	20.57	7.89	0.45*	12.35	0.45
WT6	1096	14.6	4.12	0.45*	21.36	0.45
WT2	1070	13.27	6.43	0.44*	13.76	0.44
WT1	1096	11.95	3.42	0.08*	1.2	0.08
WT4	1081	11.75	3.55	0.44*	11.16	0.44

Table 5.13: Average execution times (seconds) for 10 runs.

When running the experiments with the dataset of the total wind farm, it heavily affected the *efficiency* of the algorithms. It doesn't seem to be a direct correlation between the amount of outliers present and execution time. The most interesting turbine is turbine 1 where all algorithms showed their best *efficiency*. When compared with the AA-MMTS and LoMST, FCM-MMTS is the clear winner, achieving really low execution times in all turbines.

Area Under Curve

Data	N	ϕ	AA	FCM	LoMST	Best(↑)
WT8	1096	53.38	0.46 (0)	0.45 (0.01)	0.52* (0)	0.52
WT3	1096	42.06	0.57* (0)	0.52 (0)	0.53 (0)	0.57
WT7	1096	38.05	0.48 (0)	0.5 (0.01)	0.52* (0)	0.52
WT5	1089	20.57	0.56* (0)	0.51 (0.04)	0.54 (0)	0.56
WT6	1096	14.6	0.51 (0)	0.45 (0)	0.52* (0)	0.52
WT2	1070	13.27	0.57 (0)	0.47 (0)	0.58* (0)	0.58
WT1	1096	11.95	0.54* (0)	0.51 (0.02)	0.53 (0)	0.54
WT4	1081	11.75	0.49 (0)	0.5 (0.01)	0.56* (0)	0.56

Table 5.14: Average of AUC values for 10 runs.

By looking at the table we can conclude that LoMST wins at the probability game of assigning a random normal observation a lower score than a random anomalous observation. The LoMST achieves better results when the neighbourhood becomes more global (higher values for k). The AA-MMTS got interesting results when generating partitions with the highest number of archetypes, 5, and for the turbines with top most outliers.

Average Precision

Data	N	ϕ	AA	FCM	LoMST	Best(↑)
WT8	1096	53.38	0.5	0.49	0.53*	0.53
WT3	1096	42.06	0.47*	0.45	0.45	0.47
WT7	1096	38.05	0.37	0.39	0.4*	0.4
WT5	1089	20.57	0.23*	0.21	0.22	0.23
WT6	1096	14.6	0.16*	0.14	0.16*	0.16
WT2	1070	13.27	0.19	0.15	0.2*	0.2
WT1	1096	11.95	0.14*	0.13	0.14*	0.14
WT4	1081	11.75	0.12	0.12	0.14*	0.14

Table 5.15: Average of APR values for 10 runs.

Like in the semi-supervised scenario, APR seems to benefit from datasets with high percentage of outliers. The three algorithms got similar results and LoMST is the winner and benefits from a really local neighbourhood, k set to 1.

Precision at N

Data	N	ϕ	AA	FCM	LoMST	Best(↑)
WT8	1096	53.38	0.51	0.5	0.54*	0.54
WT3	1096	42.06	0.47*	0.43	0.45	0.47
WT7	1096	38.05	0.38	0.39*	0.39*	0.39
WT5	1089	20.57	0.22	0.2	0.23*	0.23
WT6	1096	14.6	0.16*	0.11	0.15	0.16
WT2	1070	13.27	0.19	0.16	0.21*	0.21
WT1	1096	11.95	0.16*	0.14	0.15	0.16
WT4	1081	11.75	0.12	0.1	0.15*	0.15

Table 5.16: Average of *P@n* values for 10 runs.

The tendency seems to follow, *P@n* got low values for datasets with few outliers. There isn't a clear winner since the three algorithms got similar values.

Precision

Data	N	ϕ	AA	FCM	LoMST	Best(↑)
WT8	1096	53.38	0.52* (0)	0.5 (0.01)	0.49 (0)	0.52
WT3	1096	42.06	0.47 (0)	0.43 (0)	0.6* (0)	0.6
WT7	1096	38.05	0.37 (0)	0.38 (0)	0.63* (0)	0.63
WT5	1089	20.57	0.26 (0)	0.22 (0.03)	0.82* (0)	0.82
WT6	1096	14.6	0.17 (0)	0.13 (0)	0.86* (0)	0.86
WT2	1070	13.27	0.17 (0)	0.12 (0)	0.89* (0)	0.89
WT1	1096	11.95	0.12 (0)	0.12 (0.01)	0.89* (0)	0.89
WT4	1081	11.75	0.11 (0)	0.12 (0)	0.91* (0)	0.91

Table 5.17: Average of *precision* values for 10 runs.

Algorithms in the AD domain with good values for *precision* are desired. For this metric there is no doubt that LoMST is the clear winner achieving really high values, 0.9, for datasets with few outliers present in the data.

Recall

Data	N	ϕ	AA	FCM	LoMST	Best(↑)
WT8	1096	53.38	0.54 (0)	0.49 (0.04)	0.55* (0)	0.55
WT3	1096	42.06	0.58* (0.04)	0.44 (0.02)	0.55 (0)	0.58
WT7	1096	38.05	0.44 (0)	0.48 (0.05)	0.51* (0)	0.51
WT5	1089	20.57	0.54* (0)	0.51 (0.04)	0.52 (0)	0.54
WT6	1096	14.6	0.46 (0)	0.49 (0.05)	0.5* (0)	0.5
WT2	1070	13.27	0.59* (0.01)	0.44 (0.03)	0.52 (0)	0.59
WT1	1096	11.95	0.57* (0)	0.47 (0.01)	0.51 (0)	0.57
WT4	1081	11.75	0.42 (0)	0.47 (0.05)	0.51* (0)	0.51

Table 5.18: Average of *recall* values for 10 runs.

Recall is the other metric we try to balance together with *precision* for AD algorithms. In the semi-supervised scenario *recall* got small values for datasets with few outliers and higher values for the top most outliers datasets, but this doesn't happen here. The AA achieved the best results for turbines 3 and 2 while LoMST was more consistent achieving somewhat the same results for all turbines.

Accuracy

Data	N	ϕ	AA	FCM	LoMST	Best(↑)
WT8	1096	53.38	0.48 (0)	0.47 (0.01)	0.52* (0)	0.52
WT3	1096	42.06	0.54* (0)	0.52 (0.01)	0.53 (0)	0.54
WT7	1096	38.05	0.5 (0)	0.51* (0.02)	0.51* (0)	0.51
WT5	1089	20.57	0.58* (0)	0.52 (0.05)	0.52 (0)	0.58
WT6	1096	14.6	0.59* (0)	0.45 (0.03)	0.51 (0)	0.59
WT2	1070	13.27	0.56* (0.01)	0.5 (0.03)	0.53 (0)	0.56
WT1	1096	11.95	0.46 (0)	0.53* (0.02)	0.51 (0)	0.53
WT4	1081	11.75	0.55* (0)	0.54 (0.02)	0.52 (0)	0.55

Table 5.19: Average of *accuracy* values for 10 runs.

In the unsupervised scenario *accuracy* has a similar behaviour to *recall*. In the turbines where *recall* is higher, *accuracy* is also higher. For unsupervised AD algorithms one can draw interesting conclusions, dependent on the data, but never forgetting that *accuracy* is not recommended to be used alone. For this metric, AA-MMTS is the winner when running with 4 and 5 archetypes.

F-measure

Data	N	ϕ	AA	FCM	LoMST	Best(↑)
WT8	1096	53.38	0.53* (0.01)	0.49 (0.03)	0.52 (0)	0.53
WT3	1096	42.06	0.52 (0.02)	0.44 (0.01)	0.58* (0)	0.58
WT7	1096	38.05	0.4(0)	0.42 (0.02)	0.56* (0)	0.56
WT5	1089	20.57	0.35 (0)	0.3 (0.03)	0.63* (0)	0.63
WT6	1096	14.6	0.24 (0)	0.2 (0.01)	0.63* (0)	0.63
WT2	1070	13.27	0.26 (0)	0.19 (0)	0.65* (0)	0.65
WT1	1096	11.95	0.2 (0)	0.19 (0.01)	0.65* (0)	0.65
WT4	1081	11.75	0.18 (0)	0.19 (0.01)	0.66* (0)	0.66

Table 5.20: Average of *F1* values for 10 runs.

One of the most important performance assessment metric for AD. It is easy to see here that LoMST wins in practically almost every turbine, achieving the best results for the top lowest outliers datasets.

5.3.3 Unsupervised: on Bootstrap Sampling

The bootstrap sampling technique, also called bootstrapping, is a statistical technique used when data is scarse. It differs on k-fold cross-validation in the sense it samples with replacement from the initial sample, hence the generated samples will have the same size of the original population. Some data points may be duplicated, and others data points from the initial sample may be omitted in a bootstrap sample. In its early days this technique was considered really heavy in terms of computation, but as computing power has increased and becomes less expensive, bootstrap techniques have become more widespread.

Since this technique was used in [24] to minimize the cluster instability and then select the number of clusters for k-Means, we thought it would be interesting to run the unsupervised experiments with bootstrapping to see if it improves some of the most relevant metrics. The initial plan was to run for the semi-supervised algorithms as well, but due to lack of time it was not possible.

The turbines selected for these new experiments were the turbines 1 and 4 since they presented not-so-great values for the *precision*, *recall* and *f1* metrics. It was possible to see that AA-MMTS had a nice improvement from 0.42 to 0.56 for *recall* and, a slight improvement from 0.11 to 0.12 and from 0.18 to 0.2 for *precision* and *f1* respectively, for turbine 4. Turbine 1 showed no improvement for these metrics. The FCM got worse results for these metrics on both turbines, and LoMST simply didn't benefit.

Chapter 6

Conclusion and Future Work

In this work, fuzzy clustering approaches and semi-supervised techniques were explored in time series data sets to find anomalies in wind turbines. In such tasks, evaluation measures must be established, and in this work the ROC AUC, the APR, *P@n* and the remainder mentioned in Section 5.3 were used.

With this work we aim to answer the following questions:

- **Does the "one class" methods produce acceptable results?**: The nearest-based algorithms are sensitive to the hyperparameter k and tend to favor datasets with few outlier for the majority of the metrics. The results in general are not that great with the exception of *accuracy*;
- **What is the performance of Archetypal Analysis with the Modified Mahalanobis Taguchi System extension when compared with FCM-MMTS?**: In terms of *efficiency* FCM-MMTS is optimal and doesn't seem to be afflicted by the amount of outliers in the data. It is really hard to choose a winner but given into account the trade-off between *recall*/*F-measure* and time, AA-MMTS is preferable;
- **What is best semi-supervised and unsupervised approach?**: For the semi-supervised scenario we point out the CBLOF as the winner since in terms of *efficiency* is really similar to LOF and HBOS, which got the lowest execution times, and when it comes to most important metrics in AD, CBLOF has the best values. For the unsupervised scenario, LoMST is the clear winner and the preferred choice if time is not an issue.

As future work we propose:

a) Accurate labelling of the anomalies

b) To explore more advanced ML techniques for re-sampling imbalanced data

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. "TensorFlow: A system for large-scale machine learning." In: *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*. 2016. ISBN: 9781931971331. arXiv: 1605.08695.

[2] H. Abdi and L. J. Williams. *Principal component analysis*. 2010. DOI: 10.1002/wics.101.

[3] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. "On the surprising behavior of distance metrics in high dimensional space." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2001. ISBN: 9783540414568. DOI: 10.1007/3-540-44503-x_27.

[4] R. Agrawal and R. Srikant. "Fast Algorithms for Mining Association Rules." In: *Proc. of 20th International Conference on Very Large Data Bases, {VLDB'94}*. 1994. ISBN: 1-55860-153-8.

[5] I. Ahmed, A. Dagnino, and Y. Ding. "Unsupervised Anomaly Detection Based on Minimum Spanning Tree Approximated Distance Measures and its Application to Hydropower Turbines." In: *IEEE Transactions on Automation Science and Engineering* (2019). ISSN: 15455955. DOI: 10.1109/TASE.2018.2848198.

[6] M. E. Aminanto, H. Kim, K. M. Kim, and K. Kim. "Another fuzzy anomaly detection system based on ant clustering algorithm." In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E100A.1 (2017), pp. 176–183. ISSN: 17451337. DOI: 10.1587/transfun.E100.A.176.

[7] B. Auslander, K. M. Gupta, and D. W. Aha. "A comparative evaluation of anomaly detection algorithms for maritime video surveillance." In: *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense X* 8019 (2011), p. 801907. ISSN: 0277786X. DOI: 10.1117/12.883535.

[8] A. S. de Baêna. "Dissertação de mestrado em eng mec, FCT NOVA, Análise de Temperaturas de Componentes para Controlo de Condição de Turbinas Eólicas." In: (2016).

[9] D. Bailey and E. Wright. "Background to SCADA." In: *Practical SCADA for Industry*. 2003. DOI: 10.1016/b978-075065805-8/50001-5.

[10] J. C. Bezdek. "Physical Interpretation of Fuzzy Isodata." In: *IEEE Transactions on Systems, Man and Cybernetics* (1976). ISSN: 00189472. DOI: 10.1109/TSMC.1976.4309506.

[11] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. 1981. DOI: 10.1007/978-1-4757-0450-1.

[12] J. C. Bezdek, R. Ehrlich, and W. Full. "FCM: The fuzzy c-means clustering algorithm." In: *Computers and Geosciences* (1984). ISSN: 00983004. DOI: 10.1016/0098-3004(84)90020-7.

[13] A. P. Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms." In: *Pattern Recognition* (1997). ISSN: 00313203. DOI: 10.1016/S0031-3203(96)00142-2.

[14] M. M. Breuniq, H. P. Kriegel, R. T. Ng, and J. Sander. "LOF: Identifying density-based local outliers." In: *SIGMOD Record (ACM Special Interest Group on Management of Data)* (2000). ISSN: 01635808. DOI: 10.1145/335191.335388.

[15] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle. "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study." In: *Data Mining and Knowledge Discovery* (2016). ISSN: 1573756X. DOI: 10.1007/s10618-015-0444-8.

[16] V. Chandola, A. Banerjee, and V. Kumar. *Anomaly detection: A survey*. 2009. DOI: 10.1145/1541880.1541882.

[17] P. Contreras and F. Murtagh. "Hierarchical clustering." In: *Handbook of Cluster Analysis*. 2015. ISBN: 9781466551893. DOI: 10.1201/b19706.

[18] N. Craswell. "Precision at n." In: *Encyclopedia of Database Systems*. 2009. DOI: 10.1007/978-0-387-39940-9_484.

[19] A. Cutler and L. Breiman. "Archetypal analysis." In: *Technometrics* (1994). ISSN: 15372723. DOI: 10.1080/00401706.1994.10485840.

[20] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui. "A comparative evaluation of outlier detection algorithms: Experiments and analyses." In: *Pattern Recognition* (2018). ISSN: 00313203. DOI: 10.1016/j.patcog.2017.09.037.

[21] M. Du, L. B. Tjernberg, S. Ma, Q. He, L. Cheng, and J. Guo. *A SOM based Anomaly Detection Method for Wind Turbines Health Management through SCADA Data*. Tech. rep. 2016.

[22] M. El-Banna. "Modified Mahalanobis Taguchi System for Imbalance Data Classification." In: *Computational Intelligence and Neuroscience* (2017). ISSN: 16875273. DOI: 10.1155/2017/5874896.

[23] I. Epifanio. "Functional archetype and archetypoid analysis." In: *Computational Statistics and Data Analysis* (2016). ISSN: 01679473. DOI: 10.1016/j.csda.2016.06.007. arXiv: 1601.06911.

[24] Y. Fang and J. Wang. "Selection of the number of clusters via the bootstrap method." In: *Computational Statistics and Data Analysis* (2012). ISSN: 01679473. DOI: 10.1016/j.csda.2011.09.003.

[25] M. B. Ferraro and P. Giordani. "A toolbox for fuzzy clustering using the R programming language." In: *Fuzzy Sets and Systems* (2015). ISSN: 01650114. DOI: 10.1016/j.fss.2015.05.001.

[26] S. R. Gaddam, V. V. Phoha, and K. S. Balagani. "K-Means+ID3: A novel method for supervised anomaly detection by cascading k-Means clustering and ID3 decision tree learning methods." In: *IEEE Transactions on Knowledge and Data Engineering* (2007). ISSN: 10414347. DOI: 10.1109/TKDE.2007.44.

[27] M. Goldstein and A. Dengel. "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm." In: *KI-2012: Poster and Demo Track* (2012).

[28] M. Goldstein and S. Uchida. "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data." In: *PLoS ONE* (2016). ISSN: 19326203. DOI: 10.1371/journal.pone.0152173.

[29] E. Gonzalez, B. Stephen, D. Infield, and J. J. Melero. "Using high-frequency SCADA data for wind turbine performance monitoring: A sensitivity study." In: *Renewable Energy* (2019). ISSN: 18790682. DOI: 10.1016/j.renene.2018.07.068.

[30] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. 2012. ISBN: 9780123814791. DOI: 10.1016/C2009-0-61819-5.

[31] R. Hathaway and J. Bezdek. "Local convergence of the fuzzy c-Means algorithms." In: *Pattern Recognition* 19 (Dec. 1986), pp. 477–480. DOI: 10.1016/0031-3203(86)90047-6.

[32] R. Hathaway and J. Bezdek. "Recent convergence results for the fuzzy C-means clustering algorithms." In: *Journal of Classification* 5 (Feb. 1988), pp. 237–247. DOI: 10.1007/BF01897166.

[33] Z. He, X. Xu, and S. Deng. "Squeezer: An efficient algorithm for clustering categorical data." In: *Journal of Computer Science and Technology* (2002). ISSN: 10009000. DOI: 10.1007/BF02948829.

[34] Z. He, X. Xu, and S. Deng. "Discovering cluster-based local outliers." In: *Pattern Recognition Letters* (2003). ISSN: 01678655. DOI: 10.1016/S0167-8655(03)00003-5.

[35] D. S. Hochbaum and D. B. Shmoys. "BEST POSSIBLE HEURISTIC FOR THE k-CENTER PROBLEM." In: *Mathematics of Operations Research* (1985). ISSN: 0364765X. DOI: 10.1287/moor.10.2.180.

[36] A. K. Jain. "Data clustering: 50 years beyond K-means." In: *Pattern Recognition Letters* (2010). ISSN: 01678655. DOI: 10.1016/j.patrec.2009.09.011.

[37] R. F. Kashef. "Ensemble-Based Anomaly Detection Using Cooperative Learning." In: *Proceedings of Machine Learning Research* 71 (2017), pp. 43–55. URL: http://proceedings.mlr.press/v71/kashef18a/kashef18a.pdf.

[38] D. Kateris, D. Moshou, X. E. Pantazi, I. Gravalos, N. Sawalhi, and S. Loutridis. "A machine learning approach for the condition monitoring of rotating machinery." In: *Journal of Mechanical Science and Technology* (2014). ISSN: 1738494X. DOI: 10.1007/s12206-013-1102-y.

[39] H. P. Kriege, P. Kroger, E. Schubert, and A. Zimek. "Outlier detection in axis-parallel subspaces of high dimensional data." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2009. ISBN: 3642013066. DOI: 10.1007/978-3-642-01307-2_86.

[40] D. Liu, C. H. Lung, I. Lambadaris, and N. Seddigh. "Network traffic anomaly detection using clustering techniques and performance comparison." In: *Canadian Conference on Electrical and Computer Engineering* (2013), pp. 3–6. ISSN: 08407789. DOI: 10.1109/CCECE.2013.6567739.

[41] F. T. Liu, K. M. Ting, and Z. H. Zhou. "Isolation forest." In: *Proceedings - IEEE International Conference on Data Mining, ICDM*. 2008. ISBN: 9780769535029. DOI: 10.1109/ICDM.2008.17.

[42] H. Louie and A. Miguel. "Lossless compression of wind plant data." In: *IEEE Transactions on Sustainable Energy* (2012). ISSN: 19493029. DOI: 10.1109/TSTE.2012.2195039.

[43] P. Marti-Puig, A. M. Blanco, J. J. Cárdenas, J. Cusidó, and J. Solé-Casals. "Feature selection algorithms for wind turbine failure prediction." In: *Energies* (2019). ISSN: 19961073. DOI: 10.3390/en12030453.

[44] M. Mehrotra and N. Joshi. "Anomaly Detection in Temporal data Using Kmeans Clustering with C5 . 0." In: (2017), pp. 77–81. DOI: 10.9790/1813-0605017781.

[45] L. Millán-Roures, I. Epifanio, and V. Martínez. "Detection of anomalies in water networks by functional data analysis." In: *Mathematical Problems in Engineering* 2018 (2018). ISSN: 15635147. DOI: 10.1155/2018/5129735.

[46] T. M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2.

[47] S. Nascimento, B. Mirkin, and F. Moura-Pires. "Modeling proportional membership in fuzzy clustering." In: *IEEE Transactions on Fuzzy Systems* 11.2 (2003), pp. 173–186. ISSN: 1941-0034. DOI: 10.1109/TFUZZ.2003.809889.

[48] S. Nascimento and N. Madaleno. "Unsupervised Initialization of Archetypal Analysis and Proportional Membership Fuzzy Clustering." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2019. ISBN: 9783030336165. DOI: 10.1007/978-3-030-33617-2_2.

[49] N. Noppe, W. Weijtjens, and C. Devriendt. "High frequent SCADA-based thrust load modeling of wind turbines." In: *Wind Energy Science Discussions* (2017). DOI: 10.5194/wes-2017-46.

[50] S. Novakov, C. H. Lung, I. Lambadaris, and N. Seddigh. "Combining statistical and spectral analysis techniques in network traffic anomaly detection." In: *International Conference on Next Generation Networks and Services, NGNS*. 2012. ISBN: 9781479921683. DOI: 10.1109/NGNS.2012.6656106.

[51] R. Pandya and J. Pandya. "C5. 0 Algorithm to Improved Decision Tree with Feature Selection and Reduced Error Pruning." In: *International Journal of Computer Applications* (2015). DOI: 10.5120/20639-3318.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. "Scikit-learn: Machine learning in Python." In: *Journal of Machine Learning Research* (2011). ISSN: 15324435. arXiv: 1201.0490.

[53] S. Ramaswamy, R. Rastogi, and K. Shim. "Efficient algorithms for mining outliers from large data sets." In: *SIGMOD Record (ACM Special Interest Group on Management of Data)* (2000). ISSN: 01635808. DOI: 10.1145/335191.335437.

[54] G. P. Rédei. "Euclidean Distance." In: *Encyclopedia of Genetics, Genomics, Proteomics and Informatics*. 2008. DOI: 10.1007/978-1-4020-6754-9_5603.

[55] E. Roberts, B. A. Bassett, and M. Lochner. "Bayesian Anomaly Detection and Classification." In: (2019). arXiv: 1902.08627. URL: http://arxiv.org/abs/1902.08627.

[56] F. Role, S. Morbieu, and M. Nadif. "CoClust: A Python Package for Co-Clustering." In: *Journal of statistical software* 88 (Mar. 2019), pp. 1–29. DOI: 10.18637/jss.v088.i07.

[57] A. Romero, Y. Lage, S. Soua, B. Wang, and T. H. Gan. "Vestas V90-3MW Wind Turbine Gearbox Health Assessment Using a Vibration-Based Condition Monitoring System." In: *Shock and Vibration* (2016). ISSN: 10709622. DOI: 10.1155/2016/6423587.

[58] K. A. Ross et al. "Cross-Validation." In: *Encyclopedia of Database Systems*. 2009. DOI: 10.1007/978-0-387-39940-9_565.

[59] A. V. Sadr, B. A. Bassett, and M. Kunz. "A Flexible Framework for Anomaly Detection via Dimensionality Reduction." In: (2019). eprint: 1909.04060. URL: http://arxiv.org/abs/1909.04060.

[60] G. Sancho, D. Q. D. Moncada, and S. Mendes. "Mining Extremes through Fuzzy Clustering." In: (2018).

[61] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane, and G. Nenadic. *Machine learning methods for wind turbine condition monitoring: A review*. 2019. DOI: 10.1016/j.renene.2018.10.047.

[62] R Suganya and R Shanthi. "Fuzzy C-Means Algorithm-A Review." In: *International Journal of Scientific and Research Publications* (2012).

[63] J. Tang, Z. Chen, A. W. C. Fu, and D. W. Cheung. "Enhancing effectiveness of Outlier detections for low Density Patterns." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2002. ISBN: 9783540437048. DOI: 10.1007/3-540-47887-6_53.

[64] S. Tripathy and P. L. Sahoo. "A Survey of different methods of clustering for anomaly detection." In: *International Journal of Scientific & Engineering Research* 6.1 (2015), pp. 351–357.

[65] C. H. Tsang and S. Kwong. "Ant colony clustering and feature extraction for anomaly intrusion detection." In: *Studies in Computational Intelligence* (2006). ISSN: 1860949X. DOI: 10.1007/978-3-540-34956-3_5.

[66] A. Turnbull, J. Carroll, A. McDonald, and S. Koukoura. "Prediction of wind turbine generator failure using two-stage cluster-classification methodology." In: *Wind Energy* (2019). ISSN: 10991824. DOI: 10.1002/we.2391.

[67] Vestas Wind Systems. *General Specification V90 - 3.0 MW*. Tech. rep. 2004. DOI: 10.1007/s10661-011-2038-2.

[68] Y. X. Wang and Y. J. Zhang. *Nonnegative matrix factorization: A comprehensive review*. 2013. DOI: 10.1109/TKDE.2012.51.

[69] F. Wanner. "Anomaly Analysis using Host-behavior Clustering." In: (Jan. 2007).

[70] C. H. Yeh, M. H. Lin, C. H. Lin, C. E. Yu, and M. J. Chen. "Machine learning for long cycle maintenance prediction of wind turbine." In: *Sensors (Switzerland)* (2019). ISSN: 14248220. DOI: 10.3390/s19071671.

[71] B. Yuan, C. Wang, C. Luo, F. Jiang, M. Long, P. S. Yu, and Y. Liu. *WaveletAE: A Wavelet-enhanced Autoencoder for Wind Turbine Blade Icing Detection*. 2019.

[72] Y. Zhao, Z. Nasrullah, and Z. Li. "PyOD: A Python Toolbox for Scalable Outlier Detection." In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7. URL: http://jmlr.org/papers/v20/19-011.html.