ISCTE IUL

University Institute of Lisbon

Department of Information Science and Technology

Big Data Analytics Applied to Sensor Data of Engineering Structures: Automatic Detection of Outliers

António Lorvão Ferreira Antunes

A Dissertation presented in partial fulfillment of the Requirements for the Degree of

Master in Computer Science

Supervisor

Elsa Alexandra Cabral da Rocha Cardoso, Assistant Professor, PhD

ISCTE-IUL

Co-Supervisor

José Eduardo de Mendonça Tomás Barateiro, Invited Assistant Professor, PhD

LNEC

September 2017

Resumo

De forma a controlar a segurança de estruturas de Engenharia Civil, estas são monitorizadas através de diversos sensores. Os dados de sensores são utilizados para construir modelos estatísticos e preditivos, porém necessitam de ser previamente tratados. A deteção e tratamento de Outliers (erros nos dados) é um processo lento e difícil que vamos tentar melhorar através da utilização de técnicas de Aprendizagem Automática e de Data Mining. Com o crescimento de Big Data, Outliers vão aparecer com mais frequência, e sem um método automático de deteção podemos não ser capazes de antecipar problemas e agir a tempo.

Nesta dissertação temos como objectivo identificar e tratar Outliers em dados de sensores (utilizando dados reais de uma barragem), comparando e tentando melhorar os métodos actuais. Devido à falta de datasets classificados, vamos utilizar métodos de Clustering (agrupamento de dados) que nos permitem compreender que dados devem ou não ser classificados como outliers. Vamos introduzir um algoritmo que utiliza dados dos Sistemas de Aquisição Manuais e utilizá-lo juntamente com um algoritmo de clustering (DBSCAN) e métodos actuais de maneira a criar um método que é capaz de identificar e remover a maioria dos outliers nos datasets usados para demonstração.

Palavras-chave: Detecção de Outliers, Big Data Analytics, Dados de Sensores, Aprendizagem Automática, Data Mining

Abstract

To be able to control structural safety, engineering structures are monitored by different kinds of sensors. The sensor data collected is used to create statistical and predictive models, but data needs to be processed. Outlier detection and treatment is a costly and slow process that we will try to improve through the use of data mining and machine learning techniques. In a Big Data centered world, outliers will appear more often and without an automated way to detect them, we may not be able to anticipate and act on time.

In this dissertation we will try to identify and treat outliers from sensor data (using real datasets from a dam), comparing and trying to improve the current baseline methods. Since we do not have labeled datasets, we will use clustering methods that allow us to group data and therefore understand which points should be classified as an outlier. We will introduce an algorithm that makes use of Manual Acquisition System measurements and combines it with a clustering algorithm (DBSCAN) and baseline methods to create a method that is able to identify and remove most of the outliers in the datasets used for demonstration.

Keywords: Outlier Detection, Big Data Analytics, Sensor Data, Machine Learning, Data Mining.

Acknowledgements

I would like to express my special thanks to the Professors Elsa Cardoso and José Barateiro for allowing and encouraging me to take this opportunity and the knowledge I took from it. A special thanks to my colleague Filipe, who accompanied me during this process from the beginning.

A very very big Thank You to my friends and my girlfriend for keeping me in track (not!) during this Thesis, and to all of my big Family, in special to my mother, my father, my uncle and aunt Rui and Cati, my grandmother Mami and to my uncle Luis, who allowed me to accomplish this Thesis and, beyond that, develop my whole academic path.

Contents

List of Figures

Acronyms

ADAS	**A**utomatic **D**ata **A**cquisition **S**ystem
ANN	**A**rtificial **N**eural **N**etworks
BDA	**B**ig **D**ata **A**nalytics
BI	**B**usiness **I**ntelligence
BPNN	**B**ack**P**ropagation **N**eural **N**etworks
CM	**C**onfusion **M**atrix
CRISP-DM	**CR**oss **I**ndustry **S**tandard **P**rocess for **D**ata **M**ining
DBSCAN	**D**ensity-**B**ased **S**pacial **C**lustering of **A**pplications with **N**oise
DM	**D**ata **M**ining
DSR	**D**esign **S**cience **R**esearch
DSRM	**D**esign **S**cience **R**esearch **M**ethodology
DSS	**D**ecision **S**upport **S**ystem
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
IoT	**I**nternet of **T**hings
IT	**I**nformation **T**echnology
KDD	**K**nowledge **D**iscovery **P**rocess
KNN	**K**-**N**earest **N**eighbor
LNEC	**L**aboratório **N**acional de **E**ngenharia **C**ivil
LOF	**L**ocal **O**utlier **F**actor
MDAS	**M**anual **D**ata **A**cquisition **S**ystem
ML	**M**achine **L**earning
MLR	**M**ultiple **L**inear **R**egression
MSE	**M**ean **S**quared **E**rror
OP	**O**bservation **P**lan
PCA	**P**rincipal **C**omponent **A**nalysis
RQ	**R**esearch **Q**uestion

SaaS	**S**oftware **as a** **S**ervice
SD	**S**tandard **D**eviation
SPE	**S**quared **P**redition **E**rror
SVM	**S**upport **V**ector **M**achine
TN	**T**rue **N**egative
TP	**T**rue **P**ositive

Chapter 1

Introduction

Structures are design to support heavy loads and resist weather, seismic activity and even time (buildings from previous civilizations can still be seen today). The need for observation of those structures is, now more than ever, a case of study, not only due to construction limits being pushed everyday but also the fact that the technology is rapidly advancing (sensors, video, images, etc.), helping us to better study and understand what is happening to these structures.

Structural Engineers attempt to predict how a certain structure is going to act under a set of circumstances, in order to prevent any accidents that may happen. The way that data is gathered, transformed, visualized and used to gain knowledge about each structure is of utmost importance for structural engineers, and it is how structural safety is controlled.

Safety control in large civil engineering structures, like dams and bridges, impacts not only the entities involved directly with the structure but also the whole society that benefits from it without the risk of accidents. However, the correct interpretation of the state of each structure depends on the quality of the data collected, so it is essential to eliminate any noise and outliers from the process of data acquisition. Failing to identify and remove these outliers can produce a chain reaction, since there are complex processes that are dependent on the quality of collected data, that may lead to bad evaluations about the current safety state of the structure.

1.1 Motivation and Research Context

Nowadays, the increasing level of population demands a good management of water resources, not only for consumption but also for irrigation purposes, among others. A type of structures that is critical for that are dams, which, simply put, are barriers that impound water. Structural engineers have the challenge of managing those structures and the risks that are associated with them, not only during construction but also when in exploration and after abandonment of the structure. The lack of management and safety control may lead to the loss of human lives and heavy economic and environmental damage to the dam's region.

Safety control on large engineering structures, like dams and bridges, is key for controlling risks that may cause environmental, human and economic disasters. To be able to anticipate and act in a timely manner, those structures are monitored by different kinds of sensors that allow experts to check and ensure their structural safety. When talking about safety and monitoring on engineering structures, detecting damage is of the most importance. The identification and discovery of structural damage is based on numeric and statistical models that use data collected in their monitoring systems.

In Portugal, the DSR (Dam Safety Regulation) is followed and, accordingly, every dam project needs to include an OP (Observation Plan). The OP defines a set of inspections, observations, and studies that need to happen during the life cycle of the dam, while taking into account the classification of the dam and the main risks associated with it (Castro & Barateiro, 2015). Physical quantities, such as the water level in the reservoir, water and air temperature, accelerations representing seismic activity, displacement, tensions, deformations, movements, etc., are worth observing to understand the dam behavior and predict future reactions to a certain situation. The observation can be manual or automatic, with different frequencies (some data is worth registering once a day while other may be necessary to register every hour, and the frequency may change throughout the life cycle of the structure).

In Portugal, data from those observations is registered to an information system (currently GESTBARRAGENS is used, developed by LNEC (Laboratório

Nacional de Engenharia Civil [1]). Within this information system, that was developed to aid the safety control in dams, data is processed and can be used to obtain predictive and statistical models that help structural engineers to anticipate and prepare for any problem that may happen.

The emerging ability to acquire, process and manage data about different businesses leads to a new paradigm where we can create knowledge by analyzing the different patterns, correlations, dependencies, etc., on data with different kinds of properties and representation. With sensors, there is a high potential for acquiring data representing actual events (that are to be monitored) and lead to multiple and heterogeneous data sources. These sensors can easily be treated on a big data scenario, depending on the number of sensors, acquisition frequency, variety and volume, the processing of which fits into big data analytics. Some of the sensors are as old as the structures, since they are embedded during the construction, and in case of error they cannot be fixed or swapped, we also have lots of redundant sensors and therefore, data.

To adequately use statistical models on the monitoring data, the data needs to be previously processed. Part of this process is through outlier detection, which is one of the most important steps since the existence of errors might lead to worse statistical models. Outlier detection, when made by humans, it is a time consuming, slow and prone to error process, so it is desirable to find a reliable way to automate this process.

Outliers are not only errors resulting from the acquisition process, they can also represent a problem or a deviation in the response of the structure. To be able to differentiate an isolated outlier, for example noise in the acquisition of data on a specific sensor, the monitoring systems of the structures are redundant. In order to distinguish an isolated outlier from a deviation in the structures' response, we need to analyze the dependencies between different sensors (temperature, water level, etc). The removal of outliers in any dataset is necessary when using it to create prediction models, since we are going to train the future models in an outlier-free dataset, the results should be better than the dataset with outliers, thus improving the ability of the structural engineer to understand and predict results.

[1]National Laboratory for Civil Engineering

The proposed study aims to define a methodology for automatic detection of outliers in data from continuous monitoring of engineering structures, based on data mining and machine learning techniques. In addition, we intend to implement the proposed methodology and apply it to real data from one of the structures supervised by LNEC.

1.2 Research Methodology

To develop this thesis, we will be using the Design Science Research Methodology (DSRM) that focus on the design, development, demonstration and evaluation of artifacts (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007), as displayed in Figure 1.1. Artifacts are tools that, when used, allow us to solve an existing and perceived problem.

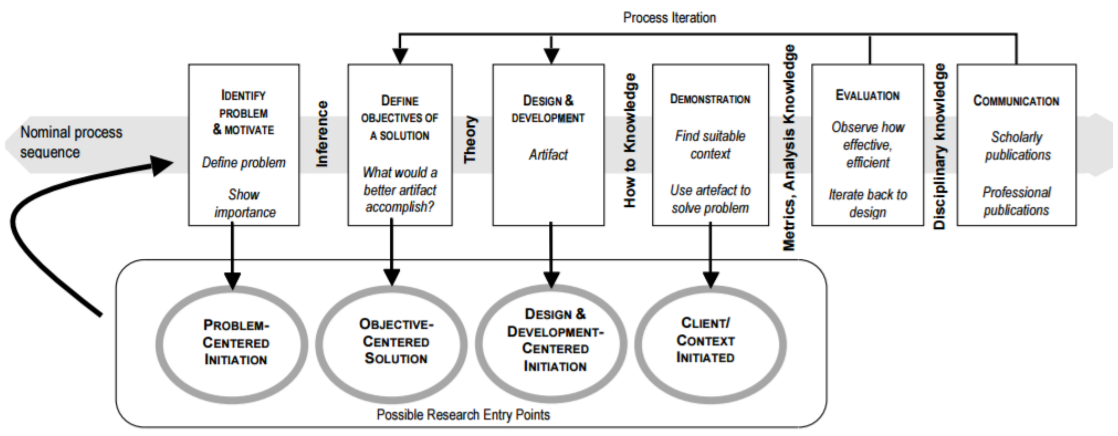
Process Iteration
Nominal process sequence
IDENTIFY PROBLEM & MOTIVATE
Define problem
Show importance
Inference
DEFINE OBJECTIVES OF A SOLUTION
What would a better artifact accomplish?
Theory
DESIGN & DEVELOPMENT
Artifact
How to Knowledge
DEMONSTRATION
Find suitable context
Use artefact to solve problem
Metrics, Analysis Knowledge
EVALUATION
Observe how effective, efficient
Iterate back to design
Disciplinary knowledge
COMMUNICATION
Scholarly publications
Professional publications
PROBLEM-CENTERED INITIATION
OBJECTIVE-CENTERED SOLUTION
DESIGN & DEVELOPMENT-CENTERED INITIATION
CLIENT/CONTEXT INITIATED
Possible Research Entry Points

Figure 1.1: Design Science Research Methodology process model (Retrieved from (Peffers et al., 2007), p. 14)

Currently, outliers are not automatically identified in the datasets. We have a desired state where we can automatically identify those outliers, resulting in a gap between our current state and our desired state that we will address by designing and developing our artifacts. We have an Objective-Centered Solution research entry point, since our research started with goal to be able to classify and remove outliers from sensor data. To achieve this, in this thesis, we will create three artifacts: (1) A baseline of how we can identify and remove outliers with existing techniques, (2) a method for outlier treatment, (3) a comparative model so we can assess which technique achieves better results. After the creation of the artifacts, we need to Demonstrate and Evaluate them, ensuring they fulfill our objectives (see Figure 1.2).

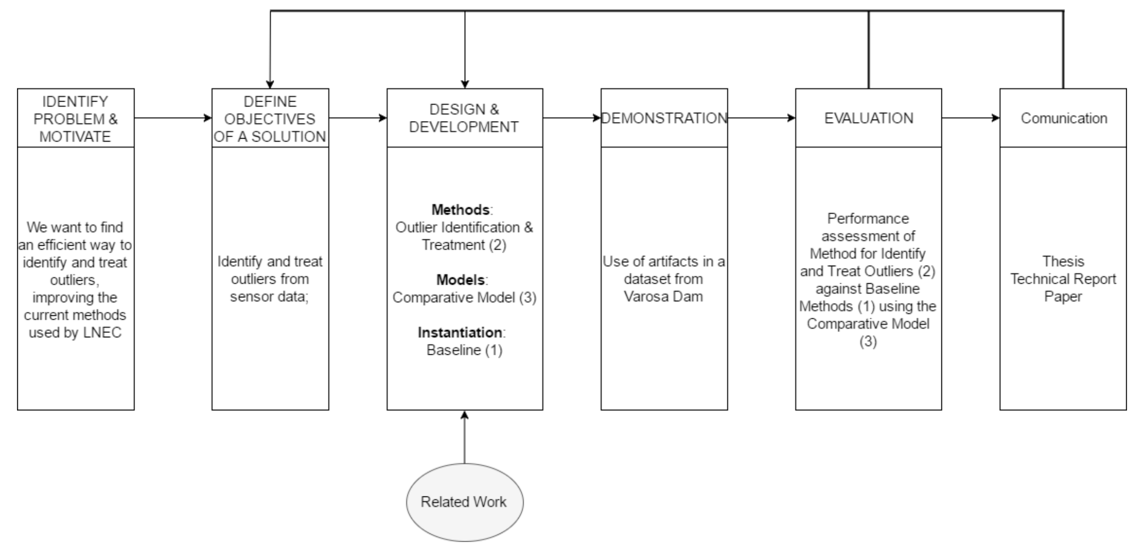
IDENTIFY PROBLEM & MOTIVATE
We want to find an efficient way to identify and treat outliers, improving the current methods used by LNEC
DEFINE OBJECTIVES OF A SOLUTION
Identify and treat outliers from sensor data;
DESIGN & DEVELOPMENT
Methods: Outlier Identification & Treatment (2)
Models: Comparative Model (3)
Instantiation: Baseline (1)
DEMONSTRATION
Use of artifacts in a dataset from Varosa Dam
EVALUATION
Performance assessment of Method for Identify and Treat Outliers (2) against Baseline Methods (1) using the Comparative Model (3)
Comunication
Thesis Technical Report Paper
Related Work

Figure 1.2: Guideline for this research (DSRM) (Adapted from Figure 1.1)

In the following subsections we will detail each step of the DSR Methodology.

1.2.1 Identify problem and motivate

Outlier identification and treatment is a problem present when using almost any dataset. When it comes to sensor data, outliers can be caused by a lot of things, for instance, mechanical problems of the sensor, computer problems when storing or transferring data, manual registration or treatment of data or a structural problem. These outliers need to be identified and treated so we can correctly assess the data and be able to make/base decisions on it. Manually identifying and treating the outliers is not an efficient and fast option, and it can be prone to errors, so we want to find an efficient way to do it.

Currently, in the analyzed case study, outliers are being identified through standard deviation and data limits (which caps the data to avoid absurd values, that is, there is a minimum and a maximum value for the data).

1.2.2 Define objectives for a solution

The objective is to develop a method to identify and treat outliers from sensor data, using Machine Learning algorithms and improving the existing solutions.

During this thesis we will try to address and answer the following research questions:

RQ1. Can we use clustering algorithms to detect outliers in sensor data?

RQ2. Is the information from other sensors useful when detecting outliers?

RQ3. Can we find a method that detects all or most of the outliers in a dataset? Does it perform better than baseline methods?

1.2.3 Design and Development

To achieve our goals, we will create three artifacts. Each artifact addresses a part of our problem:

1. **Baseline (Instantiation):** We need to understand how LNEC is currently addressing the outliers, what methods and techniques exist to identify and remove outliers and how well they perform. This is an instantiation type artifact since we will be using a dataset provided by LNEC on which we will use these methods and techniques.

2. **Outlier Identification and Treatment (Method):** After we have the baseline, we are going to develop a method to identify and remove outliers. This method is comprised of the preprocessing of the data followed by a clustering algorithm and/or any other algorithm that can be found fit for outlier identification and removal. In the end, it is expected that the output will be the outlier-free dataset or the same dataset but with the outliers correctly classified as such.

3. **Comparative Model (Model):**To be able to tell if our new method can lead to better results than the baseline techniques, we are going to construct a comparative model. Keep in mind that not all outliers are useless information, some of them (like the ones caused by structural anomalies) are

important to the creation of knowledge. In this comparative model we have to compare the percentage of outliers detected and other information that we find useful.

1.2.4 Demonstration

To demonstrate our artifacts, we will use them on datasets given to us by LNEC, from CaseStudy[2] dam, in Portugal. We will use three different labeled datasets, each one with different types and percentages of outliers (outliers will be inputed in the data by ourselves). In order to show how both the Baseline and the Outlier Identification and Treatment Method perform in this instance of the problem, we will apply them in the three datasets, showing the results. At the end, we should be able to identify and treat outliers in the datasets.

Since our method can be decomposed into several s teps, we will show the evolution and impact of each step in the datasets. The impact of each step will depend on the chosen parameters. So, beforehand, we will perform preliminary tests to determine one of the combinations of parameters that show the usefulness and better represent the impact that the method can have on this type of problem.

1.2.5 Evaluation

In the evaluation we assess if our artifacts support our proposed solutions to the problem. We will use our Comparative Model to assess if our Method for Outlier Identification and Treatment performs better than the Baseline and respond to our Research Questions. The performance evaluation will focus on the Demonstration results, showing the real impact (through the metrics utilized in the Comparative Model) that both the Baseline and our Method (and each of the steps) had on each of the datasets.

The Comparative Model should provide performance metrics that allow us to answer all of our Research Questions, based not only on the results from this section, but also demonstrations and results obtained during our research.

[2] Due to the confidentiality of the dam's real name, we have replaced it with CaseStudy

1.2.6 Communication

The communication of this research is done with this document, a LNEC report (published as a book) and a paper. The LNEC Report was already submitted and is currently under review. It includes a first approach to our solution, where we try several (baseline and machine learning) methods for outlier detection. The paper will be submitted to an international conference with a scientific peer-review process. It will include a summary of the outcomes archived in this research project.

1.3 Document Structure

This document is structured as follows:

- Chapter 2 presents the Related Work, introducing the concepts and algorithms that we will be useful for our research;
- Chapter 3 introduces our Case Study and shows the current Baseline Methods. Clustering techniques analysis and the Design and Development of our artifacts (Method for Outlier Identification and Treatment and the Comparative model) is done in this Chapter;
- Chapter 4 shows the creation of three labeled datasets with outliers and demonstrates the impact of the Baseline Method and our Method on them;
- Chapter 5 presents the results from the impact of the Baseline Method and our Method on the labeled datasets;
- Chapter 6 summarizes our research, analyzing our research questions and introducing the limitations of our work. This chapter also contains future work analysis.

Chapter 2

Related Work

In this chapter we will learn and study about subjects that we will need for the Design & Development chapter (Chapter 3). The information and knowledge we will obtain in this chapter serves as an input to the next steps of the research.

Business Intelligence (BI), a term introduced in mid-1990's by the Gartner Group, is now a major cornerstone to most companies and it is defined as an umbrella term that includes applications, infrastructure, tools and good practices to enable access and analysis of information in order to improve and optimize decision making and performance (Burton et al., 2006). Derived from Decision Support System (DSS), BI Systems are built to "get the right information to the right people at the right time", to do that BI Systems collect, transform and store data from multiple sources, in order to measure, analyze it and provide, through report tools, the right information, thus providing knowledge to the stakeholder that is making decisions (Kimball & Ross, 2013).

The amount and the quality of data to process is related to the amount of knowledge we can obtain, so to improve knowledge, companies gather bigger amounts and diverse kinds of data. The fact that almost every device nowadays has an internet connection or another kind of interaction allows the creation of a huge network of connected devices that we call The Internet of Things (IoT) (Evans, 2011). With the IoT paradigm, the quantity of data that can be collected by the connected devices is huge. Devices, like sensors, can capture an enormous amount of data, not only in quantity but in quality and types (environmental, logistic, etc.), leading to a Big Data scenario (Chen, Mao, Zhang, & Leung, 2014).

Big Data represents a challenge to companies. Although the objective of companies stays the same (obtain value and knowledge from data), there are three major differences: Volume, Velocity and Variety. We have more data, more kinds of data (structured and unstructured) and need to obtain better and faster results. To be able to analyze these datasets, companies cannot utilize normal algorithms, which are too inefficient, so they need to use faster and more powerful analytical tools (McAfee, Brynjolfsson, et al., 2012). With the evolution of Big Data, other key concepts of this challenge besides the aforementioned three V's were added. More recently we can see big companies like IBM adding Veracity (our data should be as trustworthy as possible) and Value (we should be able to obtain value from our data) to the other V's (Analytics IBM, 2017; Marr, 2015). Other companies, like SAS, refer Complexity as well, stating that in a Big Data scenario data can "quickly spiral out of control" (Insights SAS, 2017)

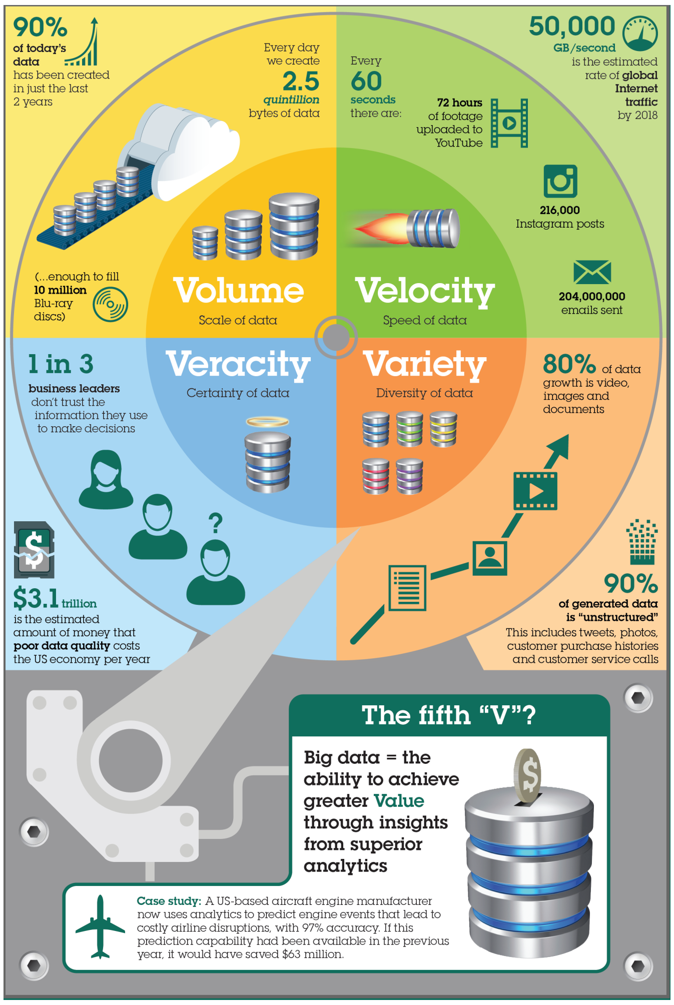
90% of today's data has been created in just the last 2 years
Every day we create 2.5 quintillion bytes of data
(...enough to fill 10 million Blu-ray discs)
Volume
Scale of data
Every 60 seconds there are:
72 hours of footage uploaded to YouTube
216,000 Instagram posts
204,000,000 emails sent
50,000 GB/second is the estimated rate of global Internet traffic by 2018
Velocity
Speed of data
1 in 3 business leaders don't trust the information they use to make decisions
?
$3.1 trillion is the estimated amount of money that poor data quality costs the US economy per year
Veracity
Certainty of data
Variety
Diversity of data
80% of data growth is video, images and documents
90% of generated data is "unstructured"
This includes tweets, photos, customer purchase histories and customer service calls
The fifth "V"?
Big data = the ability to achieve greater Value through insights from superior analytics
$
Case study: A US-based aircraft engine manufacturer now uses analytics to predict engine events that lead to costly airline disruptions, with 97% accuracy. If this prediction capability had been available in the previous year, it would have saved $63 million.

Figure 2.1: The five V's of big data. Retrieved from (IBM, 2015).

Big data analytics is a form of data analytics that focuses on Big Data, in order to discover patterns and obtain knowledge and intelligence. It uses mathematics, statistics, and modeling techniques, utilizing Data Mining and data visualization to uncover knowledge, helping to support decision making (Sun, Zou, & Strang, 2015). It can be seen as the union of several techniques in multiple areas, that combined represent the fundamentals of big data analytics. Big Data Analytics should coexist and complement the BI systems (see Figure 2.2), allowing decision making and analysis on different types of data that BI tools may could not be ready to answer.

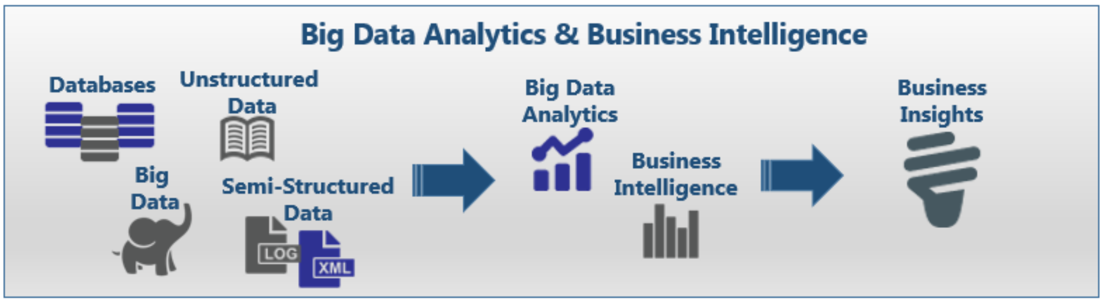

Figure 2.2: Big Data Analytics as a complementary system. Retrieved from (GlobalDataStrategy, 2017).

The remainder of this chapter is organized in the following sections:

- Section 2.1 introduces Data Mining and Data preparation. This includes possible problems that may appear in data (including Outliers);
- Section 2.2 describes some Data Visualization techniques that can be useful for outlier detection;
- Section 2.3 describes the several techniques utilized for outlier detection;
- Section 2.4 presents ways to evaluate the performance from outliers detection methods and ways to treat outliers;
- Section 2.5 introduces some concepts about outlier detection in dams, as well as Data Quality in dams.

2.1 Data Mining and Data Preparation

Data Mining is seen as the extraction of knowledge by finding meaningful patterns and rules in large amounts of data. Through the exploration and analysis of data, utilizing methods of artificial intelligence, machine learning, statistics, etc., Data Mining's goal is to discover models in data (Jain & Srivastava, 2013).

Data Mining, often related to knowledge discovery (Fayyad, Piatetsky-Shapiro, & Smyth, 1996a), can be directed or undirected, depending on the task that we are trying to accomplish. Directed data mining is used to describe or explain how data influences a certain output, while undirected data mining is utilized to find patterns or similarities in data without a specific output. Each type of data mining has techniques and tasks, however the choice of the techniques depends not only on the type of data mining but also the nature of the available data and the situation we are trying to resolve (Berry & Linoff, 1997).

The CRISP-DM (CRoss-Industry Standard Process for Data Mining) reference model is the standard guide/framework that helps organizations and data miners to conduct a data mining project. It is a phase-by-phase methodology, that encourages best practices in order to get better and faster results by breaking the data mining project into six phases and indicating the dependencies between them (Shearer, 2000; Wirth & Hipp, 2000). In this thesis, our focus will be on the Data Preparation stage.

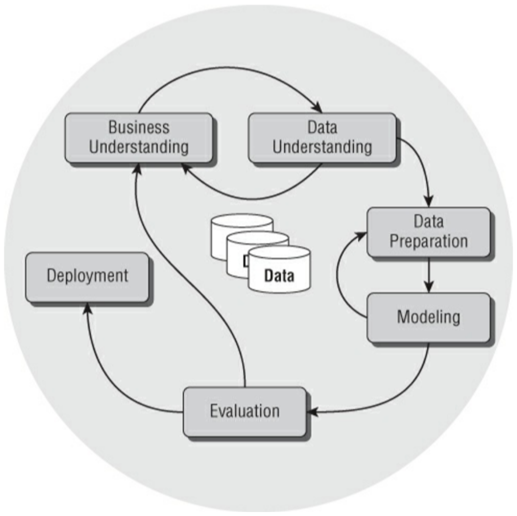

Figure 2.3: The CRISP-DM process model (Retrieved from (Abbott, 2014, chap.2))

Data preparation is a fundamental stage both in the Data Mining cycle and in the Knowledge Discovery from Databases (which sees Data Mining as a step of the process, preceded by the selection, preprocessing and transformation of data) (Fayyad et al., 1996a; Fayyad, Piatetsky-Shapiro, & Smyth, 1996b). Due to the importance of this step, it should usually take 80 percent of the time invested in any data project (Zhang, Zhang, & Yang, 2003). The main objective is to transform raw data into cleaned and quality data to be used as input for the algorithms. This is important because of three things:

1. Data from real-world sources may not be quality data. It can be incomplete (lacking values or attributes), noisy (outliers) or be inconsistent (discrepancies in names for example) and these problems can disguise useful patterns in the data;

2. Improving the efficiency of data mining, by selecting relevant data (removing anomalies, duplicates and selecting the right attributes for the task at hand), therefore reducing the size of the dataset;

3. Quality data leads to quality results. By applying the predictive algorithms in quality data we obtain outputs (patterns) of higher quality.

To achieve this, Data Preparation utilizes several techniques, like Variable Cleaning (in the Data Understanding stage we should look for variables with incorrect values, outliers or problems with Data Formats' consistency) and Feature Creation (creating new variables, in order to add value to the dataset). The choice of the techniques and the treatment we give to the data is heavily related to business knowledge and to our objectives (Abbott, 2014).

2.1.1 Missing Values

One of the most frequent problems in a dataset are Missing Values. These have multiple representations, from null values to an empty cell, but can take special representations depending on the case (for example a missing date can be represented by 11/11/11 or by 99/99/99). These values can result from simple data entry errors, unknown values or lost values from corruption or overrides. But on the other hand a missing value can be important, for instance a question on a survey that was not answered might give us some information about the respondent.

To fix missing data we have lots of options, but our primary goal is to input values into the variables (Abbott, 2014). These are some of the options:

1. **Deletion:** The simplest way to handle a missing value is by deletion. We have two options: list wise or column selection. On the first, we choose to remove any record of the dataset with missing values on the variables that we are using, but it can lead us to a situation with too few lines to make a good predictive model. On the second, we take the option to delete an entire column, removing all the information of a certain variable. This is a good choice on variables with too many missing values, but not useful in some cases (if the variable is very important to the model for example).
2. **Imputation with a Constant:** We can input a value on a variable representing the missing value. For example we can utilize a character or string to symbolize the missing value or an impossible value (-1 for age, for example).
3. **Mean and Median Imputation:** The computation of the mean and median is not expensive, and combined with the low impact to the distribution of the variable, it makes this a common method. However when utilized in a variable with high amount of missing values leads to a considerable impact on the distribution, and therefore should not be used.
4. **Imputing Missing Values from a Model:** Although more expensive and slow, the utilization of a model to input missing values leads, most of the time, to better results. The target of the model is now the variable itself, and utilizes other variables to predict the value. Decision trees and k-nearest neighbors are the most used modeling algorithms in this option.

2.1.2 Outliers

Outlier is a statistical term that represents a data point that is distant from the remaining observations in a sample. It can be seen as an observation that differentiates itself from the others enough to be possible it was not generated the same way (Hawkins, 1980). Statistically speaking an outlier can impact the results of any data analysis or statistical model. For example, in Table 2.1 we have a simple dataset in which we can observe the impact of a single outlier.

Table 2.1: Single Outlier (300) Impact on a Dataset. Adapted from (Ray, 2016, chap.3).

	Without Outlier	**With Outlier**
Values	4,4,5,5,5,5,6,6,6,7,7	4,4,5,5,5,5,6,6,6,7,7,**300**
Mean	5,45	30
Median	5	5,5
Mode	5	5
Standard Deviation	1,04	85,03

Outliers, or anomalies, can be originated by the unusual behavior of the system that is generating the data for a dataset. They can represent not only an informatics system error but also the abnormal activity or characteristics of the system and the entities involved with it. With that in mind, Outlier Analysis is important not only to remove error from data (that can influence the predictive models) but also to give some insights about our system that can be useful to detect some problem. Credit Card Fraud, Intrusion Detection Systems and Medical Diagnosis are some examples (Aggarwal, 2013). These problems are recognized as deviations of data (may not be a single outlier) from the expected pattern that data from that system should follow. In Data Mining this is called anomaly detection.

An Outlier can be caused by various factors. Mechanical and instrumental errors (like sensor failure), IT errors (overrides in databases or data corruption), human errors (manual entries) or even deviations in the system behavior (if the Outlier is caused by this, the information gathered by detection is very important to the creation of knowledge) (Hodge & Austin, 2004).

We can classify an Outlier into three categories, depending on its type: Point Outlier, Contextual Outlier and Collective Outlier (Singh & Upadhyaya, 2012). When we observe a single record on our database that can be seen as anomalous when compared to the rest of the dataset we call it a Point outlier. For example, when comparing a transaction of a very large amount if it is completely off the feature distribution, that entry on the dataset will be a Point Outlier.

The existence of a collection of points that can be seen as anomalous when observing the entire dataset is called a Collective Outlier. Each individual instance of the collection may not be an outlier but when they occur successively, the collection itself can be an outlier. For example a 0 on a dataset of a temperature sensor dataset might not be an error, but a successive occurrence of the same value during some days may indicate a problem with the sensor, and it is called a Collective Outlier.

Sometimes we cannot assume if the record is an outlier without some context. For example, on a temperature sensor, without a certain time or date we cannot assess if it is an outlier. To find Contextual Outliers we rely on two types of attributes: Contextual attributes (like time or location) and Behavioral (like temperature or amount of rainfall). One value can be an outlier given a context but in another context can be a perfectly normal value. Also it is important to note that a Point or Collective Outlier can, sometimes, also be treated as Contextual Outliers, if they have context information related to the problem.

Note that some outliers are identified by observing a single variable, while others are multi-variable outliers, and can only be detected when observing two or more variables (Abbott, 2014). After detecting an outlier it is important to interpret it, trying to understand the type and the cause, in order to gather information and therefore be able to create knowledge about the system and any relevant entities.

2.2 Data Visualization

Data Visualization is useful in Big Data Analytics (BDA) for two reasons: Data Analysis and Communication.

When used for Data Analysis, Data Visualization aims to help understanding and retrieving information from data. Combined with today's computing power, the human knowledge and creativity is used to get insight and conclusions from the data through the direct contact and interaction that visual data allows. Normally, the process of Data Understanding is helped and complemented by visual data (Keim, 2002).

Visual display of data is also important in BDA to improve communication. All the information of our system is useless unless we are able to communicate it in the right way to the stakeholders. The use of dashboards (Few, 2006), graphs, and other visualization techniques, when done right, can improve business decisions and data or information comprehension (Chen et al., 2014). Data Visualization makes use of several visual attributes, like color, form, position, etc., so that we can perceive information in a fast and comprehensive way.

According to (Kimball & Ross, 2013, p.537): “One very current big data theme is visualization of data sets. “Flying around” a petabyte of data requires spectacular performance! Visualization of big data is an exciting new area of development that enables both analysis and discovery of unexpected features and data profiling”. There are several Data Visualization techniques, but in this chapter we will focus on Scatter Plots, Histograms, Box and Whiskers Plots. These techniques can be used for outlier detection as we will see.

2.2.1 Scatter Plots

Scatter Plots are diagrams that allow two-dimensional display of points, along two axes (x, y), as seen in Figure 2.4. They are good tools to observe the relation between two variables (Friendly & Denis, 2005). Utilizing Scatter Plots allows a first look of a dataset and a quick glance of points that probably are outliers.

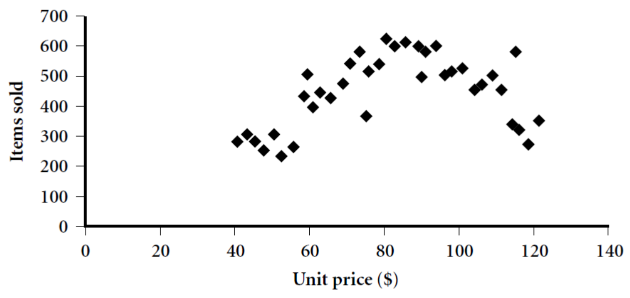

Figure 2.4: Scatter Plot example (Retrieved from (Han et al., 2011, p.59))

2.2.2 Histograms

Histograms are diagrams that allow the visualization of a single variable distribution. This is archieved by dividing the data into groups (for example [0, 10[, [10, 20[, etc.) and then count the occurrence of records for each group. Then we obtain a bar graph where the groups are displayed in the horizontal axis while the count or percentage is presented in the y-axis, as seen in Figure 2.5 (Abbott, 2014; Montgomery & Runger, 2010).

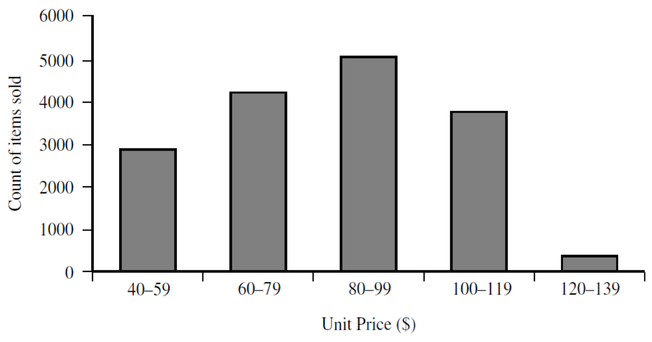

Figure 2.5: Histograms example (Retrieved from (Han et al., 2011, p.57))

2.2.3 Box And Whiskers Plots

Box and Whiskers Plots are a graphic representation, used in Statistics, which gives lots of insight on a single variable distribution. It allows to represent the median, the distribution of the data, maximum and minimum values and even some outliers. This is done by dividing the data into quartiles (each quartile represents a fourth of the data, so the first quartile represents 25% of the data, the second 50% and so on) and then obtaining a box that represents the IQR (Interquartile Range), where 50% of the data is contained, as seen in Figure 2.6. Then horizontal lines are added to represent the minimum, maximum and the median of the data (Abbott, 2014). This technique is great because it allows to identify outliers (every point that is outside a certain value, normally +/- 1.5 * IQR) and visualize them (Montgomery & Runger, 2010).

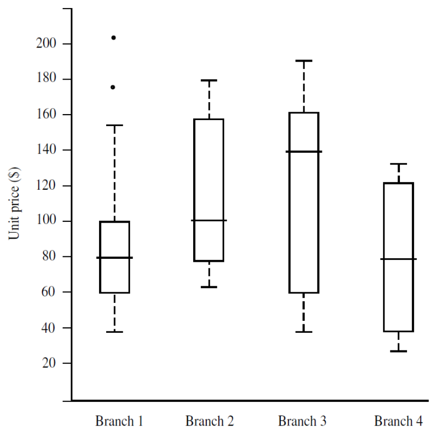

Figure 2.6: Box And Whiskers Plots example (Retrieved from (Han et al., 2011, p.55))

2.3 Outlier Detection

In order to provide cleaned data for processing and also guaranteeing database consistency and integrity, Outlier Detection (sometimes called Anomaly Detection (Chandola, Banerjee, & Kumar, 2009; Yao, Sharma, Golubchik, & Govindan, 2010)) is used to detect anomalous observations in data sets. It utilizes a set of techniques to detect system failures and behavior deviations to prevent further consequences (a small error may escalate into an enormous problem in the future) (Hodge & Austin, 2004).

A lot of applications utilize Outlier Detection, in several different fields, such as: (a) Fraud detection (e.g., detecting unusual activity in credit card or mobile phone); (b) Sensor monitoring (e.g., detecting problems or events of interest in structures, networks and several instruments); (c) Medicine (e.g., unusual patterns in heart-rate monitors may reveal a real-time urgent problem); (d) Environmental Impact (e.g., anomalies in temperature and other environmental features can reveal hidden trends) (Aggarwal, 2013; Hodge & Austin, 2004).

Several techniques have been used to detect outliers. Depending on the knowledge we have of the data we can use supervised, semi-supervised or unsupervised techniques (Chandola et al., 2009). If we have a response associated with the predictor in each observation we have a supervised problem, while in an unsupervised problem we do not have any response, so techniques that fit the model to a certain target variable will not work. In a semi-supervised problem, just a percentage of observations has a response associated, not all of them, so a different treatment is required (James, Witten, Hastie, & Tibshirani, 2013).

Techniques have two kinds of outputs: Scores (a value or percentage depending on how certain the algorithm is on the fact that each point is an outlier) or Labels (a simple binary value depending on if each point is or is not an outlier) (Hodge & Austin, 2004). The first enables a better interpretation, since different deviations may provide different interests. For example, it allows to differentiate noise (low deviation) from an actual anomaly (significant deviation) as we can see in Figure 2.7 (Aggarwal, 2013).

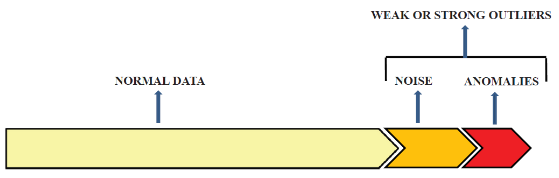

Figure 2.7: Spectrum from normal data to outlier (Retrieved from (Aggarwal, 2013, p.4))

2.3.1 Extreme Value Analysis

Mostly focused on a single variable (although it can be generalized to multi-variable data), Extreme Value Analysis is the easiest and simplest form of Outlier Detection. The basic concept is that a value that is too large or too small in that dataset is considered an outlier. Normally we find through statistical models what is the normal data distribution and therefore we are able interpret the outlier score of the points (Aggarwal, 2013).

Extreme Value Analysis proves to play an important role in Outlier Detection, since most algorithms output is a numerical value related to the deviation of the normal pattern, and extreme value analysis acts on the original value, complementing the detection algorithm (Aggarwal, 2013). We can observe this extreme values through a set of Data Visualization techniques, such as Scatter Plots, Histograms and Box and Whiskers Plots, or through statistical models and techniques (definition of a maximum and minimum value or utilizing a Gaussian distribution).

The simplest technique is to remove all values that are absurd. This is done by defining a minimum and maximum threshold (normally manually), but requires some business knowledge. For example, when observing an airplane altitude if we find a value bigger than the earth atmosphere we can infer with certainty that it is an outlier, or if a sensor observing the distance of two objects records a negative value we also can label it as an outlier (since the distance between two objects can not be inferior to zero). These limits can also be discovered using standard deviation(Montgomery & Runger, 2010).

2.3.2 Supervised Learning

Supervised Learning requires the dataset to have pre-labeled observations in order to be able to build a predictive model. In our case we will need the training set to have each observation labeled as part of one of two classes (normal data or anomaly, this is called a classification problem) or with the corresponded score of "outlierness" (from 0 to 1 for example, this is called a regression problem, since we are trying to obtain a quantitative value) (Chandola et al., 2009; Abbott, 2014).

The main objective of the Supervised Learning, also called predictive modeling, is to build a model that will be able to relate the values of known and observable variables (called predictors) to a target variable (that can be quantitative or qualitative), in a way that is useful to gain information and therefore knowledge (Abbott, 2014).

We will look into some supervised learning algorithms that we will use, like Linear Regression, Support Vector Machine and Neural Networks, however there are more examples (like Decisions Trees algorithm or The Naive Bayes Classifier).

2.3.2.1 Linear Regression

Linear Regression is one of the simplest methods used to predict a quantitative output (target). It assumes and identifies linear relationships between the output and a single input (predictor) (Simple Linear Regression) or more than one input (Multiple Linear Regression) (James et al., 2013).

The algorithm tries to find the influence of the input on the output, and represents it, normally obtaining a regression line on top of a scatter plot (see Figure 2.8), allowing to visualize the relationship between the variables and detecting "out of the normal" points (anomalies) (Abbott, 2014).

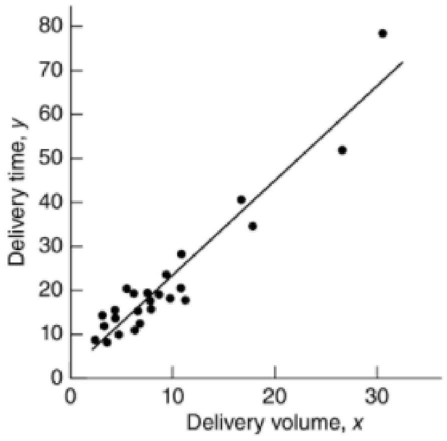

Figure 2.8: Simple Linear Regression (Retrieved from (Montgomery et al., 2012, p.2))

Linear Regression may be useful to us for two reasons:

1. We can seek predictors for the outlier score (our target) of each observation. Linear Regression can be used for Classification as well;
2. We can apply a Linear Regression to the data and identify existing values that are too distant of the expected value (predicted value). Normally, we use the standard deviation times x to set a threshold. It can be an iterative regression, since the outliers are utilized by the algorithm to build the model, so once removed, we will obtain a different (probably more realistic) model.

2.3.2.2 Logistic Regression

Logistic Regression is a linear classification technique that, instead of modeling a response, models its probability based on one or more predictors (Multiple Logistic Regression). It is mostly used for binary classification, creating a linear decision boundary that separates one category from the other in the data (Abbott, 2014; James et al., 2013).

2.3.2.3 Support Vector Machines

Support Vector Machines (SVM) is a classification algorithm, created for binary classification which allows the creation of both linear and nonlinear boundaries (James et al., 2013). The decision boundary is represented by a hyperplane that is found by maximizing the distance between the hyperplane and the points of the dataset, while separating most of the observations (it allows some misclassified points, so it is robust compared to other algorithms to individual points). The further away the point is from the hyperplane, the more certain we are that it belongs to that category (Leskovec, Rajaraman, & Ullman, 2014).

2.3.2.4 Neural Networks

Neural Networks are known as one of the most flexible algorithms, they allow the creation of a model on non linearly separable data. Based on biological neurons, this algorithm creates an artificial network of connected nodes, each one having a transfer function, that weights each one of the inputs of the node and obtains an output (normally between 0 and 1) (Abbott, 2014).

The algorithm receives n predictors and searches for values for m target variables. The artificial network contains a set of layers, each one with the respective nodes. The outputs of the layers, excluding the input and output layer, are hidden to the user. Neural Networks modeling is uses iterative learning, adapting the transfer function of the nodes (they update the weights of each input according to the error they got in the previous cycle) (Abbott, 2014).

2.3.2.5 K-Nearest Neighbor

As a classification and regression algorithm, K-Nearest Neighbor (K-NN) is known to be a model with little training cost since it uses the dataset as the model itself. It uses the k-nearest points (calculated by a distance measure) and the labels of these points to obtain an output for new observations, as seen in Figure 2.9 (Leskovec et al., 2014).

In a classification problem the new label assigned will be the most frequent class in the k nearest points (so in the case where k=1 we will assign the same label of the closest point), while in a regression we can use the average value for example.

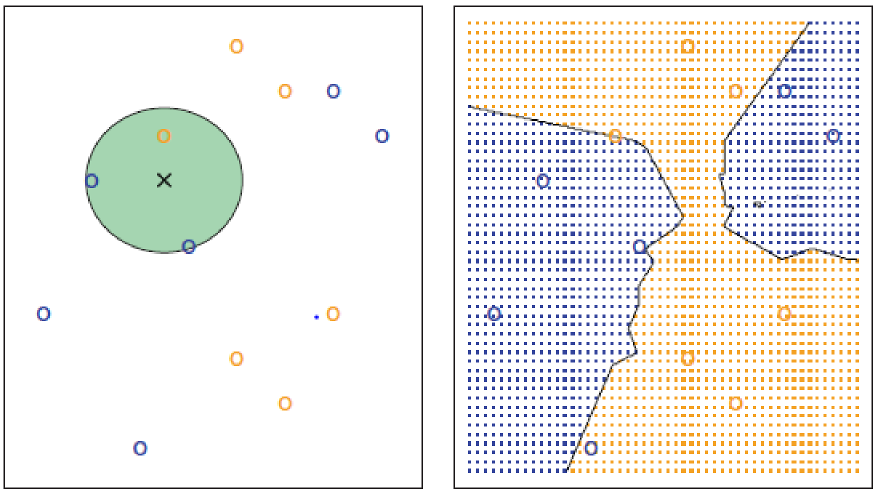

Figure 2.9: K-NN example with k=3. In this case point x will be classified as blue. On the right we can see the boundaries created by the algorithm (Retrieved from (James et al., 2013, p.40))

We can also give weights to the neighbors, so each one will have a different impact when calculating the output (for example if we have k=2, the weight of each one can be divided ($\frac{1}{2}$) or we can give a weight based on their distance, so that closer points will have more impact) (Leskovec et al., 2014).

2.3.3 Unsupervised Learning

Unsupervised Learning is used when prediction is not our primary goal, since we do not have a labeled dataset, we do not have a response variable for each observation. Unlike Supervised Learning, we have a set of features (inputs) but we are trying to discover the relationships and interesting information between them rather than their impact on other variable (James et al., 2013). Also called Descriptive Modeling, Unsupervised Learning is used on problems where the target value can not be quantified or when we are trying to find relationships between features (Abbott, 2014).

We will be focusing on Principal Component Analysis (PCA) and Clustering algorithms (like K-Means, DBSCAN and Local Outlier Factor).

2.3.3.1 Principal Component Analysis

One of the oldest multi-variable analytic method, Principal Component Analysis (PCA) is used to reduce the complexity of a dataset. This is archived by reducing the number of variables, combining them into a smaller number of variables. Although it is mainly used as a Dimensional Reduction algorithm, PCA has a lot of uses, for example, it can be used to identify correlations, infer and visualize characteristics of data (Abbott, 2014).

The main goals of PCA are data reduction, simplification, variable selection (get only the most important information) and finding relationships between objects (Wold, Esbensen, & Geladi, 1987). To do this, PCA finds new variables, called principal components, where each one is computed based on existing features variations and the previous component combination of features (Abdi & Williams, 2010).

2.3.3.2 Clustering and distance measures

The creation and visualization of groups in a dataset can be used to obtain information about the data and features involved. A group of observations, called a cluster, is obtained by finding similar observations in the dataset. The grouping of similar objects is called Clustering (Hartigan & Hartigan, 1975). In Clustering the definition of similar is domain-specific and the similarity is obtained through a distance algorithm (points closer to each other belong to the same cluster and therefore are considered similar while a point in a different cluster is different) (Leskovec et al., 2014).

Similarity between two points (observations of a dataset) is not something that can be simply calculated and the same distance algorithm may not have the same application in all problems. In any kind of space, the distance between two points cannot be negative, needs to be symmetric and follow the triangle of inequality axiom. Following these rules, some of the more known distance measures are: Euclidean Distance, Manhattan Distance, Jaccard Distance, Cosine Distance, Hamming Distance (Leskovec et al., 2014).

2.3.3.3 K-Means

K-means is a point-assignment algorithm used for classification and extraction of information about our features. Giving a number k, representing how many clusters the algorithm will try to find, K-means iteratively assigns each point to one of the k clusters (Leskovec et al., 2014). In the end each observation needs to have a cluster assigned, but can only belong to one cluster at a time, as seen in Figure 2.10 (James et al., 2013).

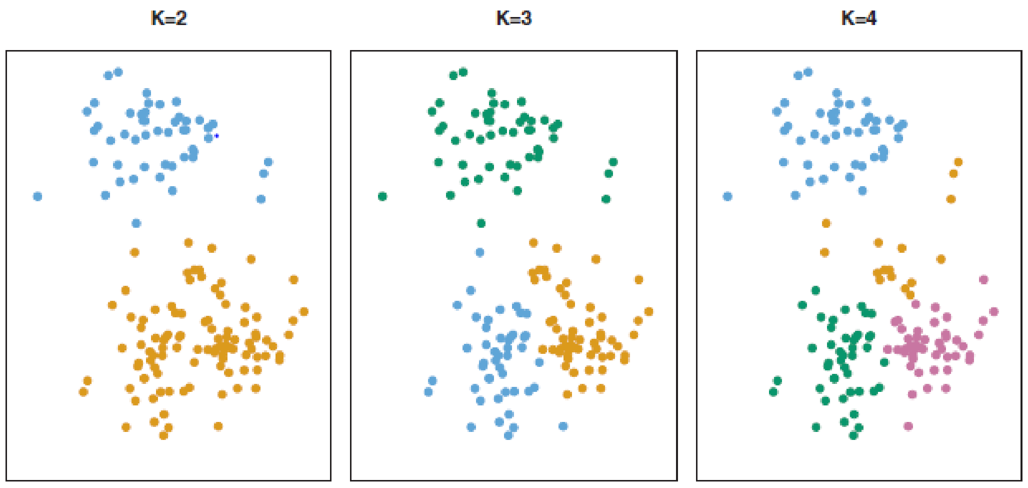

Figure 2.10: K-Means algorithm with k=2, 3 and 4. The clusters are identified by color (Retrieved from (James et al., 2013, p384))

While there are different ways to implement the algorithm on a Euclidean space, the main idea is to assign each point to the nearest cluster (by calculating the distance between the point and the center of each cluster (centroid)). In each iteration, the centroid is recalculated, and the algorithm ends when none of the points gets assigned to a different cluster in that iteration. In the end we can classify each observation based on the cluster they are assigned to.

2.3.3.4 DBSCAN

DBSCAN (Density-based spatial clustering of applications with noise) is a clustering algorithm proposed by (Ester, Kriegel, Sander, Xu, et al., 1996) It searches for "core objects", points that contain a minimum of observations (MinPoints) within its neighborhood (defined by an epsilon radius). If a point is found outside of any of the core objects neighborhood, it is considered to be noise (Han et al., 2011). DBSCAN is known to be able to discover clusters with other than linear shape, and be robust enough to handle outliers and noise.

The clusters are created as follows: after uncovering a core object (i.e., a point with high density of neighbors according to the parameters), DBSCAN starts a cluster with it and all of its neighbors. All points within the epsilon distance of the neighbors are then all added to the cluster, and this process continues until there are no more points within distance. All the points that do not belong to a neighborhood are considered noise or outliers.

2.3.3.5 Local Outlier Factor

Local Outlier Factor (LOF) is an algorithm created for outlier detection that allows a gradient classification, a degree of "outlierness". This is a different concept from the normal binary classification from other algorithms. The degree of "outlierness" is essentially related to the isolation of a certain point (Breunig, Kriegel, Ng, & Sander, 2000). This algorithm implements the similar concepts of density and neighborhood from DBSCAN. The main idea is comparing the density of a point to the density of its neighbors.

2.4 Model Evaluation and Outlier Treatment

As discussed, there are a lot of possible algorithms to detect and identify outliers, however which algorithm performs better (i.e., achieves better results)? In order to answer this question, we need to evaluate the model created by the algorithms for our dataset. Evaluation is useful not only for the choice of the algorithm but also for the choice of parameters, for example, choosing the best k (number of neighbors) in K-NN algorithm. Keep in mind that the perfect algorithm does not exist, and each specific dataset has an algorithm that will perform better when dealing with a certain task (James et al., 2013).

2.4.1 Error Measures

In a predictive setting, we can evaluate the algorithm performance by comparing the predicted response to the real one for each observation. One of the most used measures is the Mean Square Error (MSE) that is calculated using the following equation:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 \tag{2.1}$$

Where x_i is the real response for observation i, n is the number of observations and f(x_i) is the predicted response. As we can see, the larger the value obtained, the further away our predictions are from the real values (James et al., 2013).

We can calculate MSE in both training and test data, however, to find the best performing algorithm, we want to find the one with the lowest test MSE (MSE value in the test dataset). Since we are predicting, we are looking for a model that can obtain best predictions on unseen (test) observations, and not the ones we already known (and if the model was trained with the training dataset, it will probably have a small training MSE value) (James et al., 2013).

With this in mind, we can use MSE to adapt the model to the training data (by changing some parameters of the algorithm) and by doing this we can have a model that fits really well the training data (reducing the MSE value). However, in most cases a small training MSE does not correlate to a good predictive model, since test MSE may not be as good (this is called over-fitting). We often search for a small MSE in both test and training datasets, giving more importance to the test MSE (James et al., 2013).

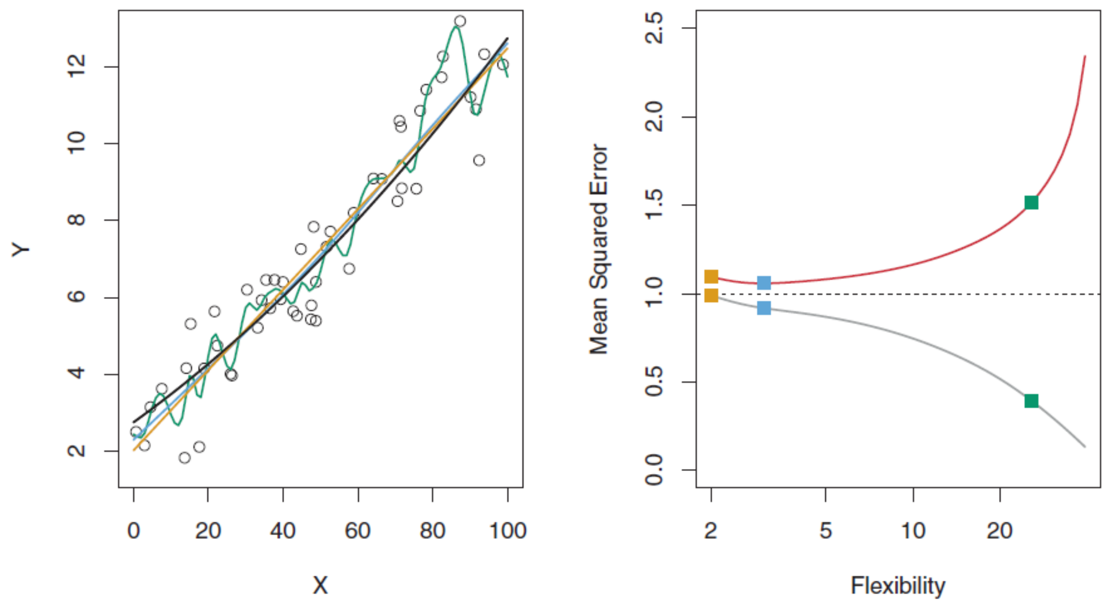

Figure 2.11: An example of Over-fitting. The model represented by green has a lower MSE in the training set (represented on the right by the gray curve – training MSE), however is the model with the highest MSE in the test set (represented by the red curve – test MSE) (Retrieved from (James et al., 2013, p.31))

In a classification setting, where we do not have a quantitative response, MSE can not be applied. We use instead an error rate given by the following equation:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i) \tag{2.2}$$

That translates in the number of misclassified predictions divided by the number of observations. If the observed response is different from the predicted one, the function I is equal to 1, otherwise is 0. Like in the regression setting, we can calculate the error both in the training (training error) and the test (test error) dataset, and the best algorithm is the one with smallest test error.

2.4.2 Confusion Matrix

Binary classification is used for problems where we intend to assign each observation to one of two classes (e.g., normal or anomaly, healthy or disease, relevant or spam, normal translation or fraudulent, etc.). Normally the model has two types of results positive (1) or negative (0). When we compare the predicted classes against the real ones, we can have four types of results for each observation (Abbott, 2014):

1. **True positives (TP):** Both the real and predicted class have a positive result;
2. **False positive (FP):** The predictive value is positive, but the observation is actually classified as negative (also called a false alarm);
3. **True negatives (TN):** Both the real and predicted class have a negative result;
4. **False negatives (FN):** The real classification for the observation is positive, but the predicted classification is negative (also called false dismissal).

Table 2.2: Confusion Matrix. Adapted from (Abbott, 2014, chap.9).

		Predicted Class	
Actual Class		0	1
	0	**TN**	**FP**
	1	**FN**	**TP**

The use of a Confusion Matrix (that lays out the count of each outcome, as seen in Table 2.2) can be of some use when evaluating a model. Depending on each problem, a false alarm error may be better than a false dismissal, for example, in cancer detection is better to have a false alarm that can be found with posterior examinations than a false dismissal. Additionally we can obtain the precision of the model (rate of correctly assigned predicted positive observations) and the recall (rate of correctly actual positives classified), as seen in Equation 2.3 and 2.4 (Abbott, 2014).

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

2.4.3 Resampling Methods

Resampling methods allow us to obtain more information about each model. Instead of fitting a model to all observations in the training set, we use only a partial amount of observations each time, allowing to obtain different fits each time and therefore more information about the model. We can use Resampling Methods to evaluate the model performance (model assessment) or to choose the right parameters for the algorithm, thus increasing or decreasing model flexibility (model selection). The most used resampling methods are Hold-Out, Cross-Validation and Bootstrap (James et al., 2013).

The most common way to split the data to evaluate a model is called Hold-Out. Normally the dataset is divided into two sets of observations, called Training and Test set, where two thirds (depending on the dataset and problem, the percentage can change) of the data are used to train and fit the model, while the remaining third is left out and used in the end to test the model (Kohavi et al., 1995).

Cross Validation is a method that splits a dataset into k subsets, and rotates the training of the model by using all but one subset, and then proceeds by testing the model in the one subset left out. Cross Validation repeats this process k-times, by leaving each one of the subsets for testing once. Cross Validation can also be used by leaving only one observation for testing, and repeating the process for every observation (this is called Leave-One-Out) (Kohavi et al., 1995).

Bootstrap is used to quantify the uncertainty of a estimator or statistical learning method. It is used, for example, to calculate the regression coefficients of linear models.

2.4.4 Outlier Treatment

After detecting outliers in the dataset, we must choose how to treat them. Remember that we have several types of outliers, and the way we treat it should depend on the data itself, the problem and the business knowledge (Abbott, 2014). We have four alternatives when treating outliers:

1. **Remove outliers:** When we find at an outlier in an observation, and we are using algorithms that may create a distorted model due to using the

anomalous observation to train, we can choose to delete the observation. This is the simplest alternative, however it is not perfect. Since the model will not be trained with outliers, when dealing with new data inputs (never seen observations), the output can be unreliable. The model will not be prepared to deal with new outliers. Another problem comes from models that should be used to find these outliers (fraud detection, anomaly detection, etc.), so they need to be trained using the existing ones.

2. **Split and model outliers from the remaining data:** Complementing the first approach, we can use this to create a new dataset with the deleted outliers and obtain a new model for the data. Doing this allows the creation of a simpler model fitted to the normal data and a second model prepared to deal with outliers. When dealing with new inputs and outliers, the model will be prepared to deal with outliers, treating them differently from the remaining data.

3. **Transforming outliers:** Here we have two options. We chose to leave the outliers with the data, but first we transform them, so their impact on the model is minimized. The first approach is to smooth extreme values, to a state where they are no longer considered outliers, while the second approach is to transform the outlier into a categorical value, or dummy variables (by doing this we allow the analytic to treat the outlier at his will).

4. **Leave the outlier:** We may chose to leave the outlier in the dataset. Once identified, the presence of outliers can lead to more robust models (built with algorithms unaffected by outliers or flexible enough), capable of dealing with outliers. This is a good choice if we aim to have a model capable of dealing with preprocessed data (or at least with noise and/or outliers).

2.5 Outlier Detection in Structural Engineering

In (Portela, Pina dos Santos, Silva, Galhardas, & Barateiro, 2005), GESTBARRAGENS system is presented as a modular system used for safety control in dams. It includes several report tools that allow the user to obtain information and analyze structures in any computer with Internet access. (Castro & Barateiro, 2015; Portela et al., 2005) both highlight the importance of safety control on dams and both agree that anomaly detection on these structures is key to ensure a good evaluation and response to problems that may lead to dangerous situations.

With the advances on technology, automated systems are utilized to monitor dams, providing an increase of information that beforehand had to be collected manually and with less frequency. However, with this increased amount of information, the potential for measurement errors also increases. Automated data acquisition system (ADAS) in Portugal are being continuously installed in large dams. The detection of measurement errors is essential to ensure safety control on these dams, and in (Mata & Tavares de Castro, 2015) a methodology where automated measurements are compared with the manual ones (manual measurements) are utilized as reference elements) is presented in order to assess the quality of stored data. A more detailed comparison between the manual data acquisition system (MDAS) and the ADAS can be seen in Table 2.3, and in Figure 2.12 we can see how the comparison between the two is useful to observe outliers (measurement errors).

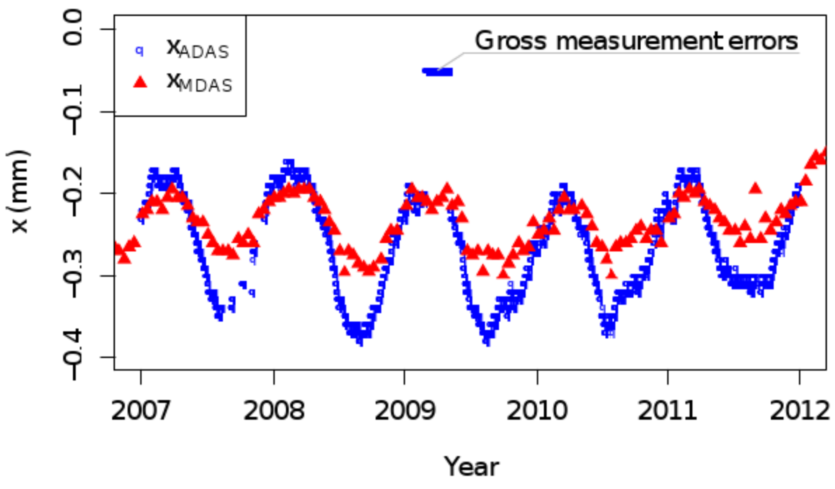

Figure 2.12: Manual vs Automatic measurements (Retrieved from (Mata & Tavares de Castro, 2015)

Table 2.3: ADAS vs MDAS Analysis

	ADAS	MDAS
Acquisition	Automated - Sensors collect measurements to data-loggers and the information is transmitted to managements systems.	Manual - Operators read measurements from display elements and record the information.
Frequency	High Frequency - Depending on the sensor, measures can be made every 15 minutes to some hours.	Low Frequency - Since an operator is obligated to be present, this readings are done daily to a monthly basis, depending on the sensor and the dam lifecycle current phase.
Errors	Errors can emerge from system or instrument malfunction.	Errors can emerge from human interpretation and decisions, but can also be affected by system or instrument malfunction.
Age	ADAS measurements started being collected around the year 2000	MDAS measurements are collected after the dam has been built, some of them dating back to 1940.
Quality	The quantity of data increases the quantity of possible errors.	Since the instruments are relatively simple and there was lots of experience gathered throughout the years, MDAS measurements are considered of good quality.

As stated in (Mata & Tavares de Castro, 2015),a measurement error is obtained when the stored value is different from the true value of the quantity measured. It consists of three elements: Methodological, Instrumental and Human, but also can be seen as a composition of gross, systematic and random errors, being that the last one is impossible to be estimated and therefore difficult to correct.

The creation of statistical and predictive models for dam behavior and safety analysis is a current subject of study. In (Salazar, Morán, Toledo, & Oñate, 2017) a survey is presented with almost 60 study cases and methods used for assessing the condition of the structure, however there is little focus on Data Preparation. Although anomaly detection can be a part of abnormal behavior analysis, in most cases does not play a role on the Data Preparation phase.

Another solution for anomaly classification in dams is presented in(Cheng & Zheng, 2013), where data is separated into the environmental variables and noise, allowing the creation of two models, in order to identify relationships with those variables and the dam response. It looks for thresholds in data and uses SPE (Squared Prediction Error) to classify anomalies into several qualitative labels (extreme environmental value, global damage, malfunction or local damage).

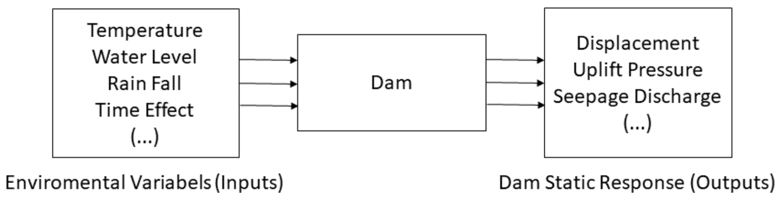

Figure 2.13: Dam System (Adapted from (Cheng & Zheng, 2013, p.49))

In (Tayfur, Swiatek, Wita, & Singh, 2005) compares Artificial Neural Networks (ANN) and Finite Element Method Models (FEM) and concludes that ANN is capable of detecting anomalous seepage on dams. (Yu, Wu, Bao, & Zhang, 2010) uses PCA on monitoring data as a method to ensure dam safety, and identifies false alarms, data reduction and noise elimination as the three main problems encountered.

BackPropagation Neural Networks (BPNN) is used to simulate environmental effects and find relationships between them, in order to find anomalous data that may need further analysis (create a system warning). It identifies as problems the limitations of the training samples and over-fitting, as well as algorithm parameters (Li, Li, Shi, Yan, & Ren, 2010).

As seen in (Mata, 2011), the Multiple Linear Regression model, when used for dam behavior prediction uses as predictors the water level (represented as H), the temperature (represented as θ) and time (represented as a date). Sometimes time can also be represented as T (number of days since the beginning of the exploration phase). The model can be trained to predict several effects (*Abertura, Deslizamento, Deslocamento, Deslocamento radial & Deslocamento tangencial*[1]).

[1]Names in Portuguese (Opening, Slippage, Displacement, Radial Displacement and Tangential Displacement)

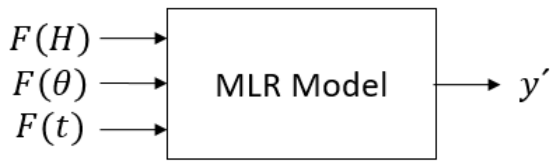

Figure 2.14: Multiple Linear Regression Model used in Dam Behavior Prediction

Most of the times, predictors are obtained through a function. (Mata, 2011) presented below, where each predictor is obtained differently: Water Level effect is represented through a polynomial function while Temperature is obtained through a linear combination of sinusoidal functions that depend on the day of the year (although it can be obtained by other sensors).

$$y'(H, \theta, t) = F(H) + F(\theta) + F(t) \tag{2.5}$$

In (Ljunggren, Logan, & Campbell, 2013) it is discussed the importance and role of data quality on dam safety assurance and assessment. Most of the data is collected by instruments, and due to new and more reliable sensors and increasing monitoring and reporting obligations, the quantity of dam safety data saw a significant growth over the last decade.

Dam safety data is considered to be of quality ("good") if we have enough information about the sensors, data acquisition and data processing procedures, as well enough data accuracy and frequency to represent the true measured values and understand dam behavior, along with the linked causes. In the end we try to obtain data "free of errors", so that the value extracted from it can be seen as true and reliable.

In Figure 2.15 we have the data quality cycle presented in (Ljunggren et al., 2013) that includes several tasks that can impact quality on dam data. Each task requires standards and guidelines, combined with careful planning and engineering expertise, to ensure quality objectives. Examples and explanations about Data Management tasks (Raw Data Collection, Processing & Data Storage, and Data Analysis) are given in the aforementionedned paper, along with ways to minimize risks and possible errors.

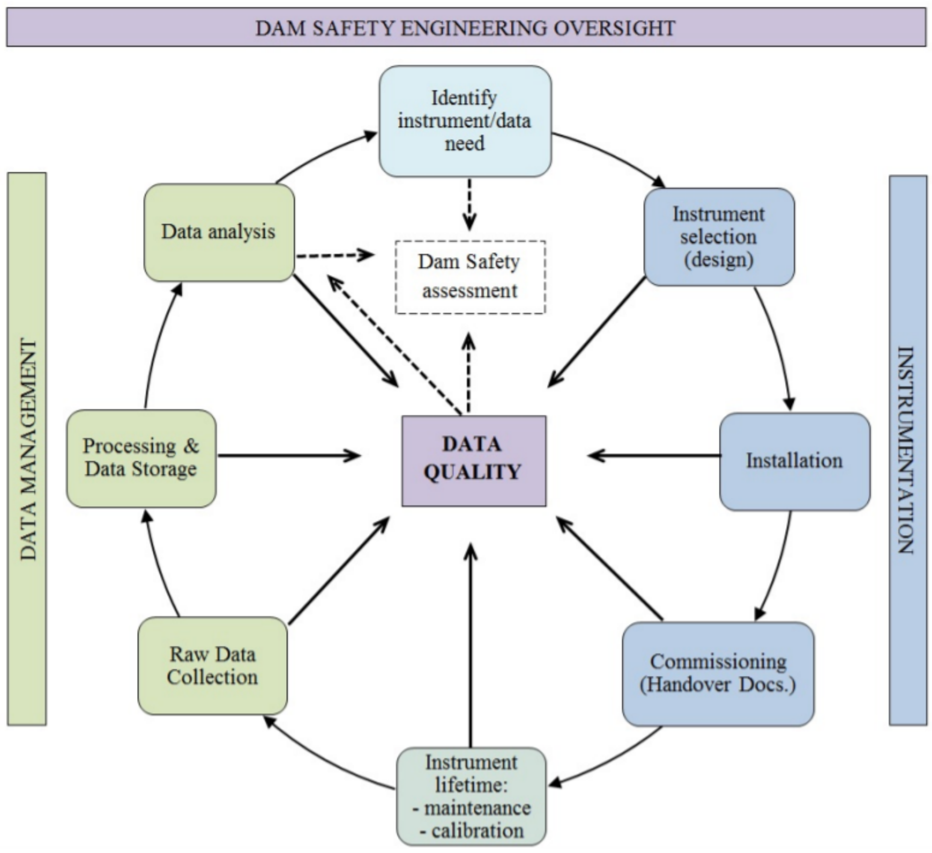
DAM SAFETY ENGINEERING OVERSIGHT
DATA MANAGEMENT
INSTRUMENTATION
Identify instrument/data need
Instrument selection (design)
Installation
Commissioning (Handover Docs.)
Instrument lifetime: - maintenance - calibration
Raw Data Collection
Processing & Data Storage
Data analysis
Dam Safety assessment
DATA QUALITY

Figure 2.15: Data Quality Cycle in Dams' Sensor Data (Retrieve from (Ljunggren et al., 2013))

Chapter 3

Design and Development

In this chapter we will describe the design and development of the artifacts. To do so, we will need to focus in a case study (CaseStudy Dam), so that we have an environment where we can develop and test the artifacts. This chapter is organized in the following sections:

- Section 3.1 describes the data from GESTBARRAGENS provided to us by LNEC and introduces our case study: CaseStudy Dam;
- Section 3.2 presents the selection of the development language for this research;
- Section 3.3 introduces the baseline methods currently used for outlier detection;
- Section 3.4 demonstrates the use of clustering algorithms in outlier detection;
- Section 3.5 introduces a new algorithm for outlier detection using MDAS measurements;
- Section 3.6 introduces our method for outlier identification and treatment;
- Section 3.7 shows how we can compare and infer about the performance of the methods and techniques presented in the previous sections.

3.1 Case Study

LNEC, by utilizing GESTBARRAGENS system, provided us with sensor data and information about the sensors monitoring dams in Portugal. In table 3.1 we can see metadata about the information. Note that ADAS sensors are very recent compared to MDAS and are fewer in different types of sensor (62 vs 20), however we can observe that are growing at a faster rate, due to their frequency (almost 34 millions of records in 74 years (0.46 millions per year) compared to 8 million in 16 years (0.5 millions per year)). So each ADAS sensor has a mean of 25 thousand records against a mean of 7.4 from the MDAS sensors. In figure 3.1 we can see the growth of data in the system.

Table 3.1: GESTBARRAGENS system meta data information

	Manual / **Automatic**	Total (Manual + Automatic)
Types of Sensors	62	82
	20	
Amount of Sensors	38651	39597
	946	
Number of Records	33719325	41720466
	8001141	
Records Date Distribution	1943 - 2017	1943 - 2017
	2001 - 2017	

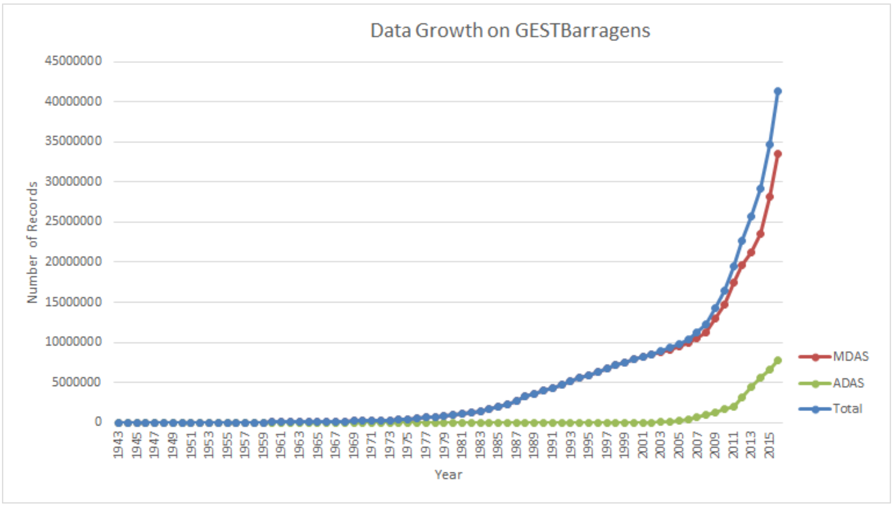

Figure 3.1: GESTBARRAGENS Data Growth from 1942 to 2016

3.1.1 CaseStudy Dam

In this thesis we use a dataset from the CaseStudy Dam to develop our artifacts. CaseStudy dam is established in Portugal. Beginning its exploration phase in 1976, CaseStudy dam is mostly used for energetic purposes. It has a maximum height of 76 meters, a crest length of 213 meters and it is divided into 13 concrete blocks. According to the OP, CaseStudy dam has a monitoring system that allows the observation of structural behavior, environmental actions and material properties, guaranteeing the safety of the dam and the 13 x 10^6 m^3 water reservoir.

Besides the MDAS (Manual data acquisition system), CaseStudy dam has an ADAS (Automatic Data Acquisition System) that collects data from several variables, such as water level, air temperature, etc., and displacements and other structure behavior variables. In table 3.2 we can see information about the monitoring system and data present in GESTBARRAGENS from CaseStudy Dam.

Table 3.2: GESTBARRAGENS system metadata information about CaseStudy Dam

	Manual / **Automatic**	Total (Manual + Automatic)
Types of Sensors	12	21
	9	
Amount of Sensors	475	509
	34	
Number of Records	452824	1310647
	857823	
Records Date Distribution	1975 - 2017	1975 - 2017
	2006 - 2017	

LNEC provided us with two distinct types of datasets from CaseStudy Dam, one directed for prediction analysis and the other for ADAS/MDAS comparison, both related to three different types of measures.

When comparing ADAS against MDAS measurements, we obtain a report where the comparison between the two kinds of measurements are presented through several plots. We can also observe a mapping table (that relates ADAS sensor to the corresponding MDAS sensor), the number of records, mean values for each sensor and residuals (difference between the measurements). We can also see the variance and correlation between the samples.

In the prediction setting, the main objective is to analyze structural behavior response and relate it to several other variables. Multiple Linear Regression is used to obtain models and the result can be visualized in pdf format (where is possible to visualize the model, allowing us to see if the model correctly fits the data, as seen in Figure 3.2). We can also obtain information about the model such as regression coefficients, adjustments (R2), ANOVA table and residuals.

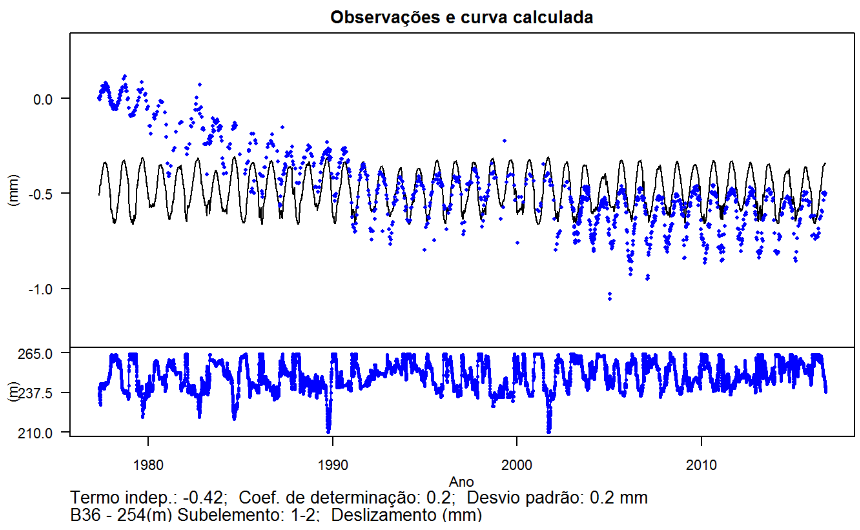

Figure 3.2: Example of a Multiple Linear Regression in the predictive setting (in Portuguese). Water Level used in the algorithm is represented below.

We will be focusing on the ADAS since the amount of data, automation and the fact that ADAS is not supervised by humans will lead to more outliers. Although the MDAS measurements are not outlier free, they are considered to be of quality (as seen in 2.3). They are manually validated before being made available on the platform (GESTBARRAGENS). Below (Table 3.3) we have information about the variables and number of sensors available in the CaseStudy Dam that we will use throughout this chapter.

Table 3.3: ADAS vs MDAS setting Dataset Information

Type of Sensor	Variable Measured	Number of Sensors
Elongometer's base	*Opening (mm)*	12 (6 ADAS + 6 MDAS)
	Horizontal Slippage (mm)	None
	Vertical Slippage(mm)	12 (6 ADAS + 6 MDAS)
Foundation gauge	*Displacement (mm)*	6 (3 ADAS + 3 MDAS)
Plumb line	*Radial Displacement (mm)*	12 (3 ADAS + 9 MDAS)
	Tangential Displacement (mm)	12 (3 ADAS + 9 MDAS)

3.2 Development Language

To build the prototype software we need to choose a programming language. The only restriction imposed by the case study is that we cannot use a SaaS, instead we should use a free platform and language. When talking about Big Data Analytics, the two most common languages are R and Python. Both have a large and active community and both are known as a common tool used by Data Scientists. In terms of popularity Python is the most used, however R is primarily used by academics and research and it is expanding rapidly. Both are ready to deal with Big Data, although both need packages to handle it well. The biggest advantage of R is the visualization of data. It allows an efficient and effective data visualization, and as it is the programming language currently used by LNEC it is the one we will choose.

3.3 Baseline

To identify outliers in this section we will utilize three simple approaches: Whiskers Boxplot, Scatter plots and Standard Deviation and Minimum/Maximum values. These are not only the simplest techniques to identify outliers but also the ones currently used in the case study.

3.3.1 Whiskers Boxplot

We will start by looking at the variables and their distribution. A good way to do so is by utilizing Whiskers Boxplot, a plot that allows us to extract relevant information with just one look. We will be using the standard boxplot from R, where we can find information about the median value, 1st and 3rd quartile, minimum and maximum values and outliers.

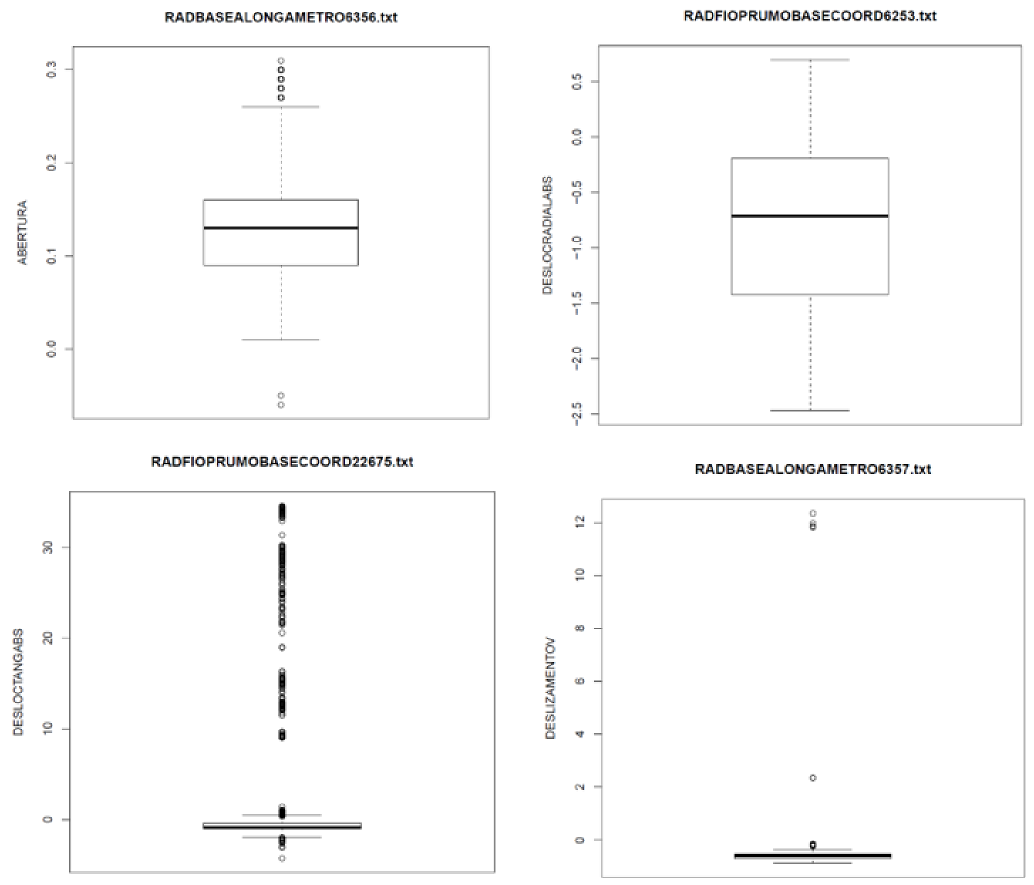

Figure 3.3: Whiskers Boxplot examples

As we can see in the Figure 3.3, we can have several scenarios with different amounts of outliers and some more extreme than others. Despite whiskers boxplot giving us some information, we do not have a full comprehension of what is going on with the variable response, and as we will see, some outliers are not getting caught by this method.

3.3.2 Scatter plots

To have a better comprehension about the variables and their behavior we plot the values according to their timestamp. We obtain a 2D plot where the y-axis represents the variable and the x-axis the date, and each point a pair (value, timestamp). We obtain the following plots, again using the standard plot tool from R.

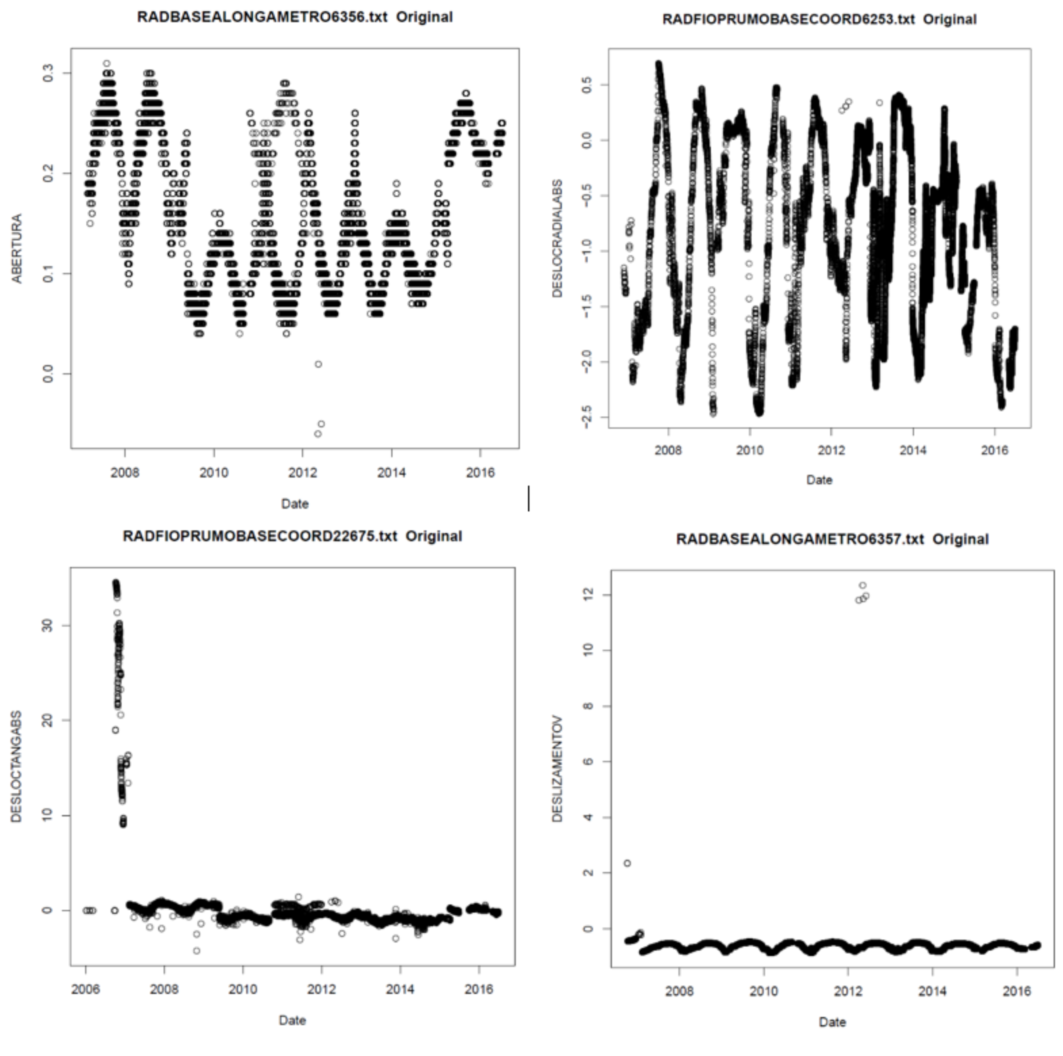

Figure 3.4: Scatter Plots examples

Looking at the plots from Figure 3.4, and without the help of other variables, we can see some points that we may consider outliers but were not caught by the whiskers boxplots. For example in the upper right plot, whiskers boxplots did not show any outliers, however we can see some points, near the year 2012 that appear to be outliers, since they are “out of the expected behavior”.

3.3.3 Standard Deviation and Min/Max approach

To identify outliers, Dam Experts use two separate techniques. Due to the business knowledge possessed for each variable and their predictors, they are able to input minimum and maximum values depending on the month/season of the year, considering any other value as an extreme outlier. Since they expect annual variation of values for the predictors, they can obtain an approximate value for the behavior variable. Another technique consists of using the prediction model created by the Multiple Linear Regression (using Water Level, Temperature and Time as predictors) and accepting any value between the predicted value and a certain threshold (normally +/− 2 or 3 times the standard deviation).

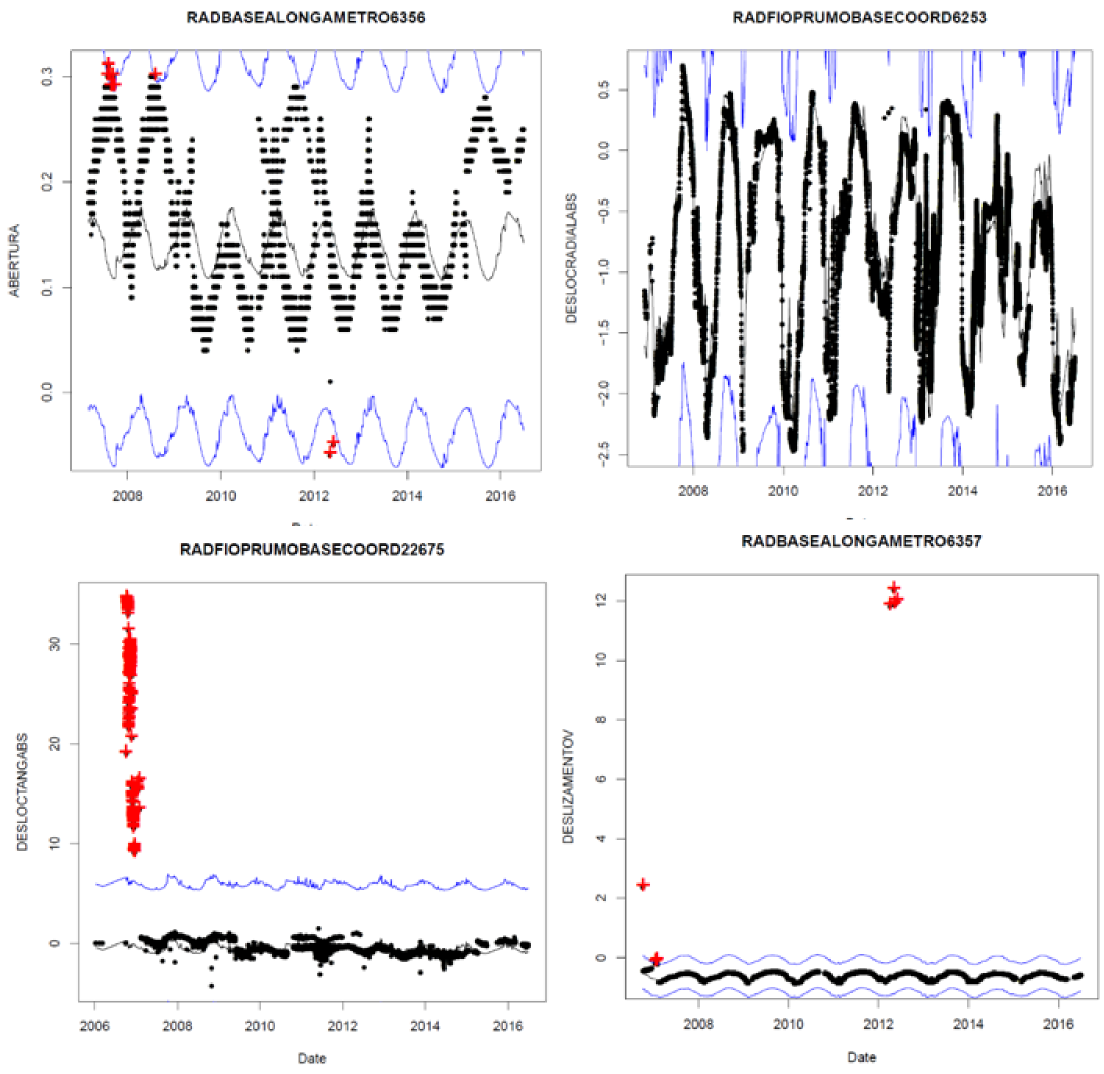

Figure 3.5: Outlier Detection using Standard Deviation to define a threshold.

In Figure 3.5 we can see the threshold (in this case $+/-$ three times the standard deviation) in blue and the predicted model as a black line. We can observe that this approach has good results when used to detect extreme outliers, however we can see that in the upper right plot no outliers were detected, while in the upper left plot only two out of the three identifiable outliers were caught (outliers caught are identified with a red cross). The problem with this technique comes from the predictive model is that: (a) is obtained using a dataset with outliers and; (b) the algorithm or parameters utilized may not be ideal for the variable in question (the model in the upper left plot for example).

3.4 Clustering Methods

In order to identify outliers in our dataset we will use three different clustering algorithms. Since we do not have a labeled dataset, the first approach that we will be using is unsupervised methods that allow us to group the data and therefore understand which points should be considered outliers or not. We will be using K-Means, DBSCAN and Local Outlier Factor, and setting a seed in order to reproduce the same results every time.

Since Dam Experts consider temperature, water level and time (representing the age of the structure) as predictors to dam's behavior due to the influence they have on the response variable, we will use them as variables for clustering. Since our dataset does not contain any information besides the timestamp and the response variable, we needed to obtain the information about the predictors to complete our dataset. We merged values for Water Level (H), Temperature (estimated as a trigonometric function of $\sin(d) + \cos(d)$, based on the day of the year) and Time (T, age of the dam in days). However these values have a daily acquisition frequency, and since the target measurements are obtained several times each day, the same predictor values are added for several measurements obtained in a given day. These values can be calculated using information from the timestamp (predictor for Temperature and Time) or collected by other sensors (predictor for Water Level). We saw in Equation 2.5 that dam experts use several functions to obtain the predictor values. In table 3.4 we show how the values can be obtained.

Table 3.4: Predictors Functions

Name	**Type**	**Function**
$F(H)$	Polynomial Function	$F(H) = H + H^2 + H^3 + H^4$
$F(\theta)$	Linear Combinations	$F(\theta) = \sin(\alpha) + \cos(\alpha)$
$F(t)$	Linear	$F(t) = Dam's\ Age(Days)$

When treating missing values in the rows we caught two different situations: one where a single entry was missing and other where an entire timeframe was missing. In both cases we decided to remove the rows. In the first case we are concerned with the continuous behavior of the variable, and since the sensor collects every set of hours the variance between values in the same day for example is not significant to the analysis. In the second case, we could input the empty values by using a model, as explained in Section 2.1.1.

3.4.1 K-Means

To use the K-Means algorithm, like we saw in Section 2.3.3.3, we need to give the following arguments: a matrix with values (each row is an observation and each column is a variable) and a value for k (number of clusters). We used a script to obtain results for all sensors (in batch) and response variables, varying the variables utilized for clustering and the value of k. We analyze 15 sensors, 27 different datasets (pair sensor/response variable) in 1701 different analysis with a total of 762826 records, taking about 7 to 8 minutes to run.

Each time we build a view, we create and scale (using the *scale*() function from R) a smaller dataset that always contains the response variable and where we add information about the predictors. The value of k is varied between 2 and a parameter value (we used the value 10). Besides utilizing a set containing only the variable, we tested the predictor sets presented in Table 3.5.

Table 3.5: Predictors Used

Predictor Set	**Functions Used**
H	$F(H)$
$H+T$	$F(H)+F(t)$
$H+COSD+SEND$	$F(H)+F(\theta)$
$H^4+COSD+SEND$	$F(H)+F(\theta)$
$H+H^4+COSD+SEND$	$F(H)+F(\theta)$
$H+H^4+COSD+SEND+T$	$F(H)+F(\theta)+F(t)$

After the model creation (we used the standard K-Means algorithm from R), we look for clusters that are composed by a small number of observations. Since the first k points are assigned randomly, one outlier can be tagged as a centroid of a cluster that probably will end up with a small set of observations. With this in mind we look for any cluster with size smaller than a certain limit (since in a space where points are equally distributed we will end up with a mean size of cluster equal to the number of observations divided by the number of clusters, we kept the same idea but always tripling the number of clusters, so that we catch only those with an extreme small size).

The distance between each observation and the centroid (of the cluster they were assigned) is calculated. The distance metric should be the same in the algorithm and the distance calculation, and in this case we used the Euclidean Distance. Note that we do not normalize any value, so Euclidean distance will give more importance to differences between variables with bigger variance (as is the case of any response variable and the water level).

Then we ordered the distances in a decreasing manner, keeping only the index of the distances above a threshold. We tried two different thresholds: (a) we used the same idea from the boxplot, where we classify as outlier any distances above the 3rd quartile + $1.5 * IQR$, since bellow the 1st quartile - $1.5 * IQR$ are points near the cluster centroid; (b) by utilizing the mean plus 3 times the standard deviation. The first option often classified too many points as outliers, so we decided to use the second. We then joined the index's to those with the ones from the clusters of outliers (small clusters), identifying all of them as outliers.

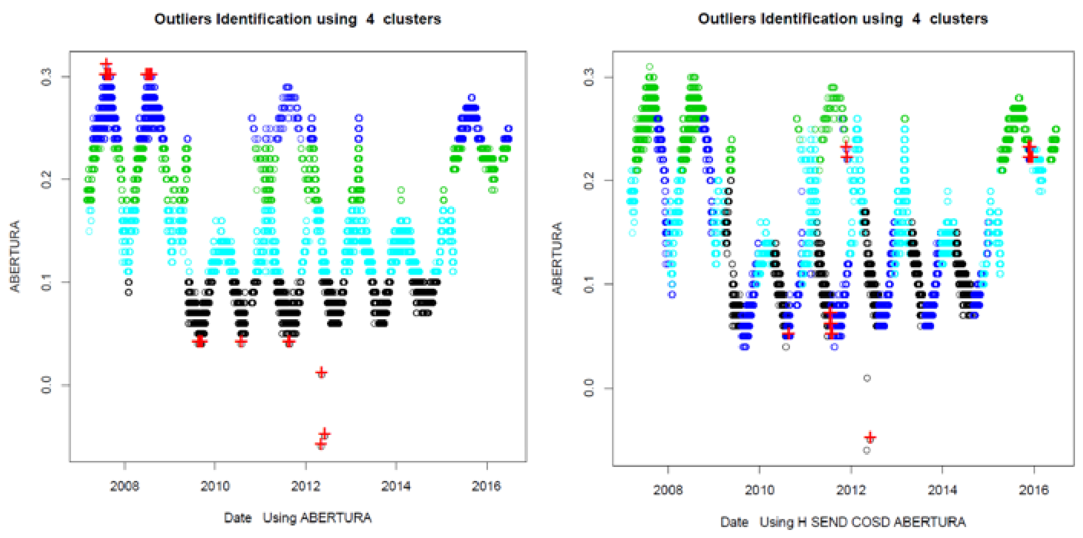

Figure 3.6: K-Means Algorithm Example using 4 clusters

The left plot we can observe the results by clustering using only the response variable and 4 clusters (in different colors). As we can observe in Figure 3.6, it identified correctly the outliers (red crosses), however also incorrectly tagged some extremes values as well. In the right we used the water level (H) and temperature ($SEND\&COSD$) together with the response variable, however it did not obtained better results, as we can see the extreme outliers were not all caught.

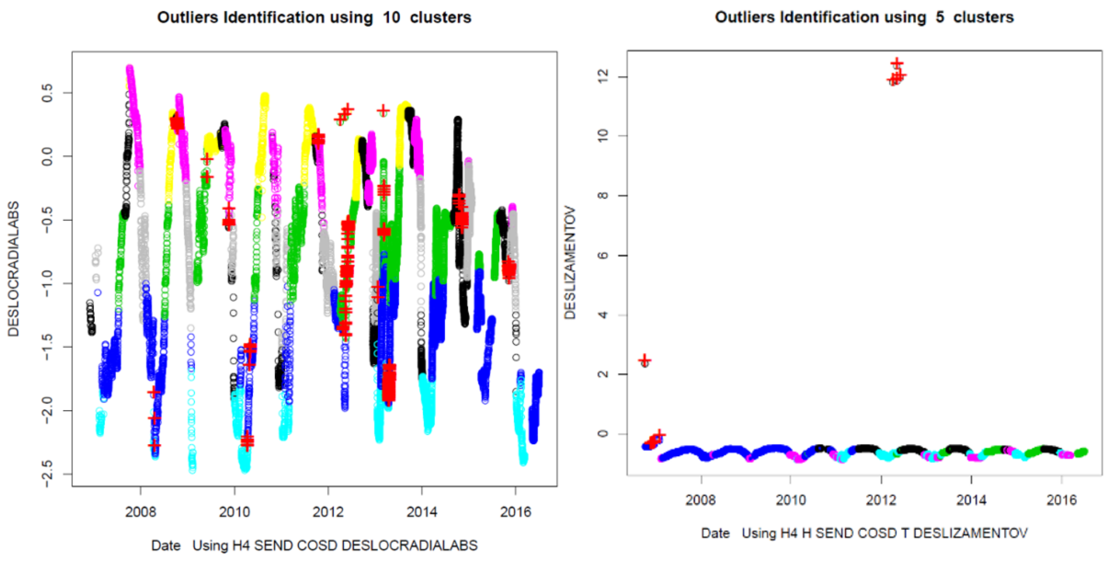

Figure 3.7: K-Means Algorithm Example with 5 and 10 clusters.

Observing Figure 3.7, starting by the left dataset, we can see that, although it caught some of the visible outliers, it labeled too many observations as outliers. When looking at the 5 cluster example (Right), the algorithm performed better, identifying correctly the extreme outliers, however the first points were not all caught.

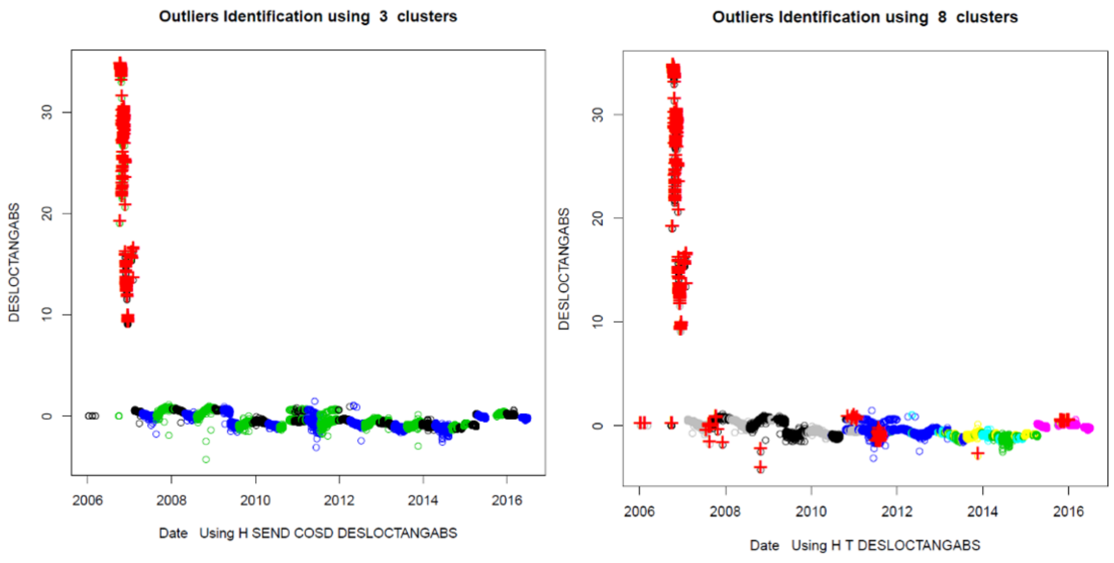

Figure 3.8: K-Means Algorithm Example with 3 and 8 clusters.

In Figure 3.8 we can observe that a lower number of clusters worked well when identifying the extreme outliers, independent of the predictive set. However only in a few test we were able to detect outliers near the main distribution of the variable. As we can see in Figure 3.9, when removing the outliers we still cannot clearly identify which points are outliers and which are not.

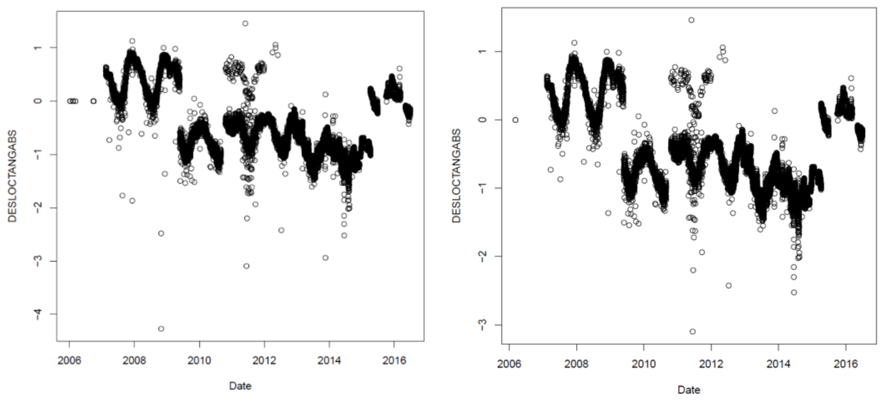

Figure 3.9: Outlier Removal after results from 3.8

3.4.2 DBSCAN

Using DBSCAN differs from the k-means algorithm in two characteristics: instead of receiving as arguments the matrix of observations and a value for k clusters, DBSCAN receives the matrix and a value for epsilon (as seen in Section 2.3.3.4); we also vary the value for Minimum Points.

Initially we used the default value (5), since studying the algorithm and its variations is not our objective, however due to the fact that we have a lot of dimensions we decided to vary it between 5 to 8 and analyze those results. The output already contains a set of outliers or noise identified by the algorithm, so we do not need to do any further calculation.

We used the algorithm (from the dbscan R library) in the same datasets, doing 2268 different analysis, in a total of 58 minutes (most of the time is consumed by the algorithm itself). Using the same predictive sets we used in K-means algorithm, we build and scale the view. Then we vary the value of epsilon (between 0.1, 0.3 and 0.5), run the algorithm, collect the outliers (present in the algorithm output) and plot them.

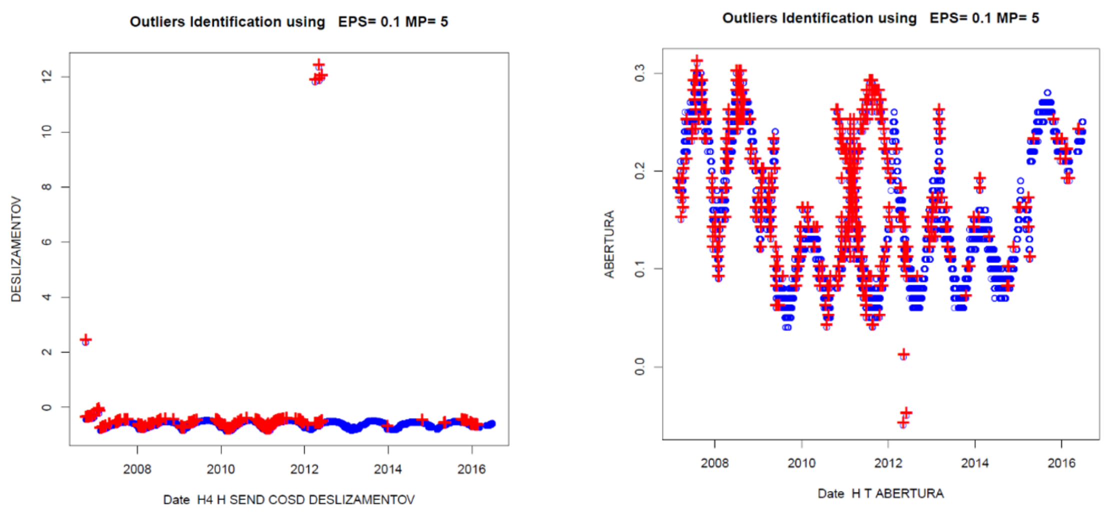

Figure 3.10: DBSCAN Example with Epsilon = 0.1

Most of the time, when using a very low value for epsilon (0.1 being the smallest we used, as seen in Figure 3.10), the algorithm identified incorrectly a great number of observations as outliers. In the other end, a big value for epsilon may not identify as outliers any observation (we tried using 1 as the epsilon value in preliminary tests and it failed to identify any observation as outlier in some cases).

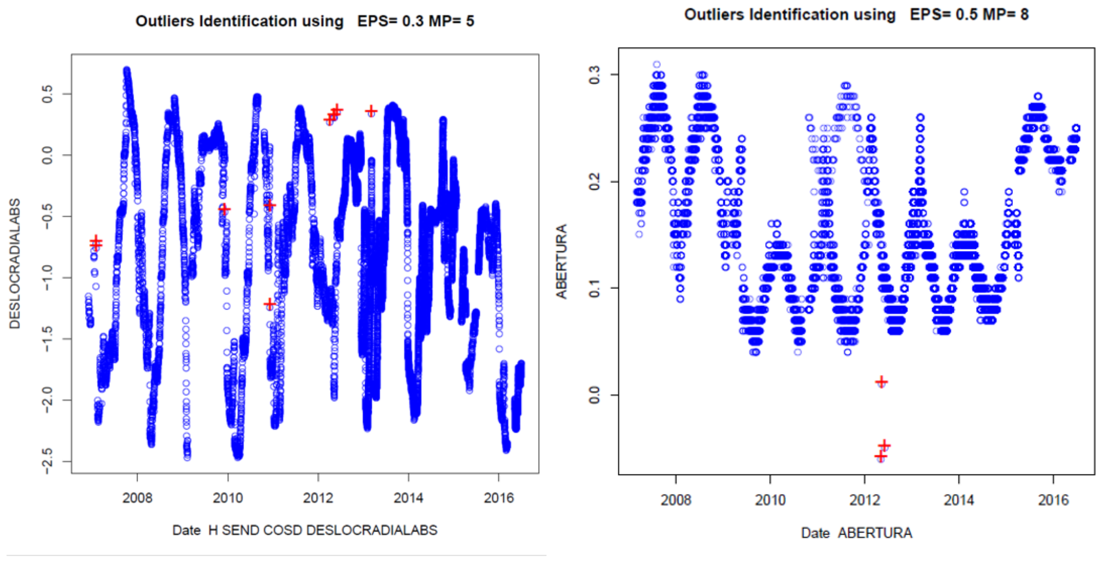

Figure 3.11: Outlier Detection using DBSCAN

We obtained good results in the different type of sensors (Figure 3.11), but when comparing its results with K-Means, DBSCAN was great when dealing with datasets that the previous algorithm struggled to classify.

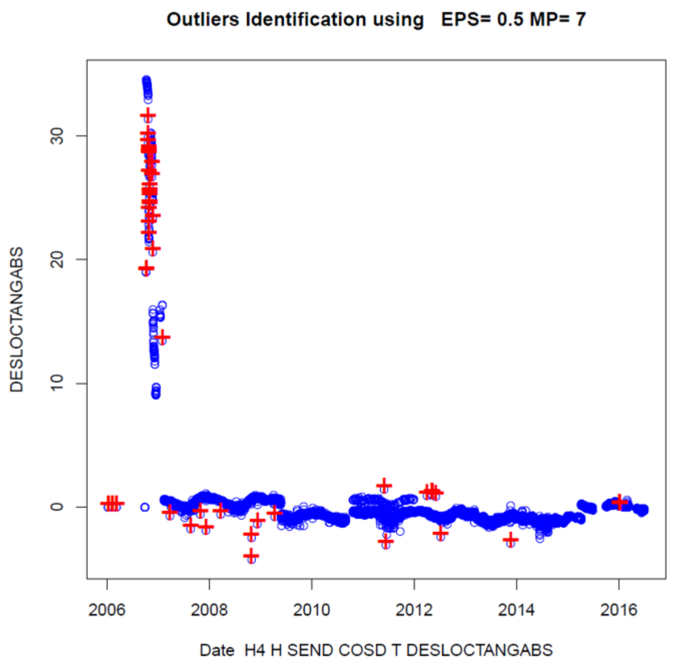

Figure 3.12: DBSCAN Example in DESLOCTANGABS dataset

Note that in the Figure 3.12, DBSCAN failed to identify the extreme outliers in the beginning of the dataset, but was able to identify correctly some outliers (not all, as seen bellow, in Figure 3.4.2) that were not identified previously by other algorithms.

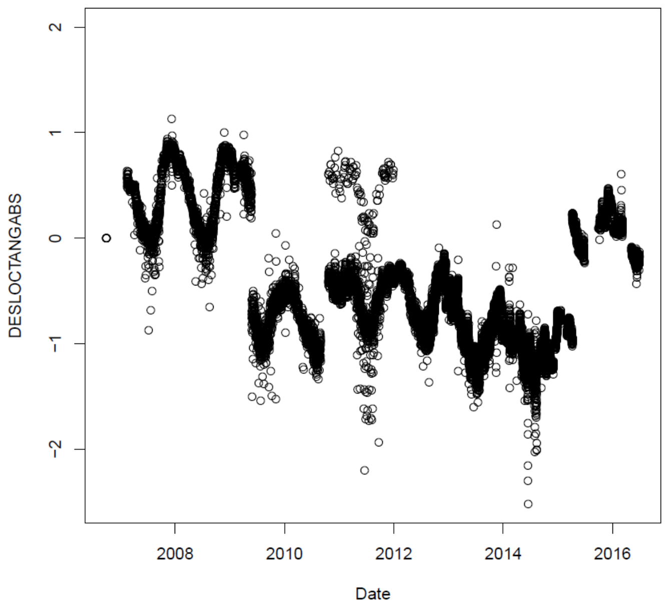

Figure 3.13: Outlier Removal after results from Figure 3.12 in DESLOCTANGABS dataset

3.4.3 Local Outlier Factor

As we saw in Section 2.3.3.5 Local Outlier Factor (LOF) is used to calculate the score of "outlierness" of each observation. To do so, it receives a matrix of values (similar to K-means and DBSCAN) and the value of k (the neighborhood of each observation). To test this algorithm we followed the same approach as before, analyzing 15 sensors, varying the same predictor sets (to build the matrix), in a total of 1323 different analysis in 52 minutes.

The LOF algorithm used was imported from the dbscan package. Since we do not have the threshold between the non-outlier and outlier values, we used the same approach as the K-means distance to calculate which points are to be considered outliers. In the future this threshold should be calculated depending on the response variable by an expert. The k was varied between 6 and 12.

In preliminary tests this set of values obtained the best results, and due to the expensive time it took to run the algorithm in batch we decided to reduce the possibilities of values for k.

During testing we had problems with the algorithm scoring. Due to the presence of several duplicated values in the matrix, the algorithm would return the value of "outlierness" as infinite, creating problems when calculating the mean and standard deviation. To address this problem we added to the matrix (in every analysis) a normalized (0 to 1) value representing the timestamp of each observation. The value is normalized to avoid a big impact, and since every timestamp is different we do not have duplicates, and as such we obtained better results.

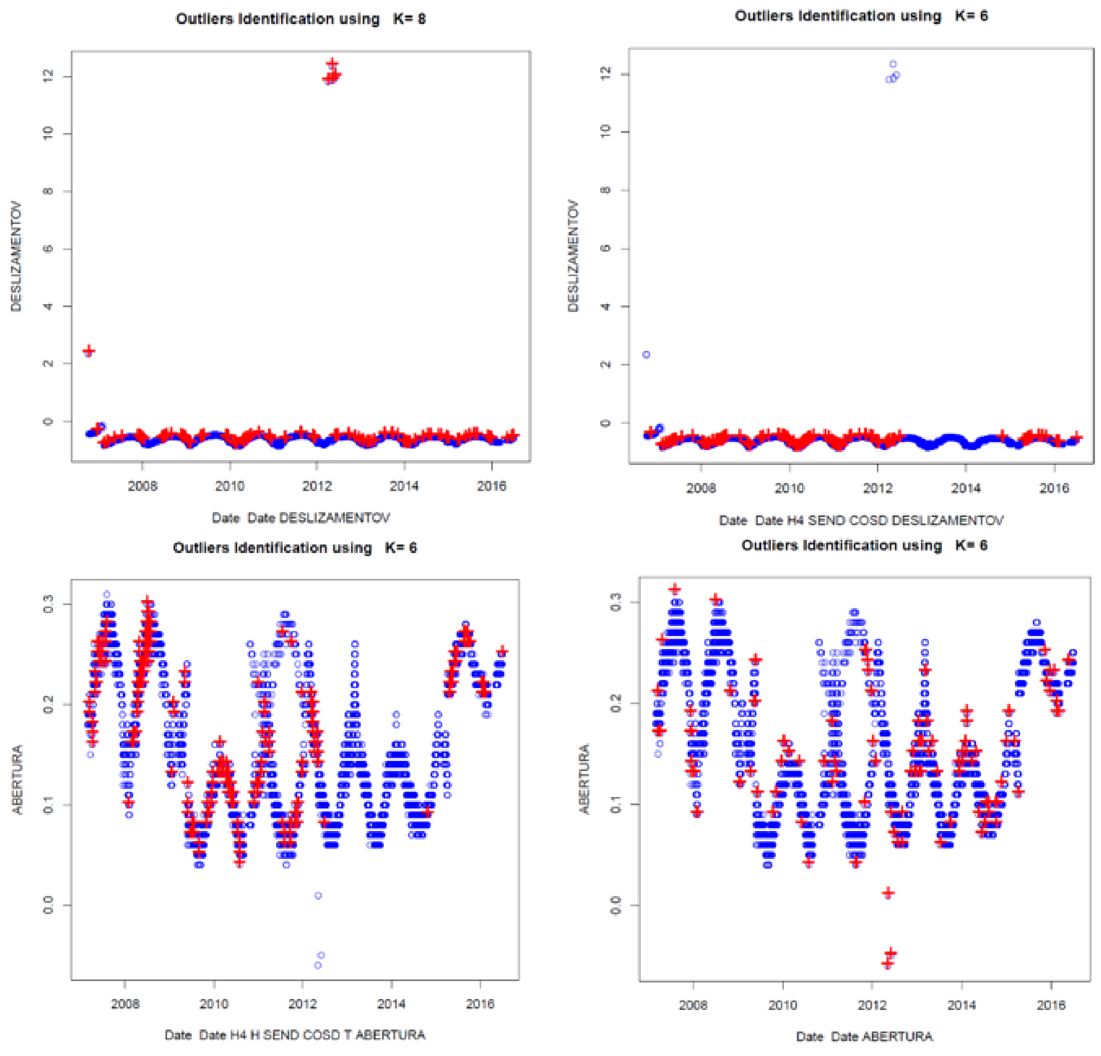

Figure 3.14: Outlier Detection using LOF. Notice that too many points are marked as outliers

As seen in Figure 3.14, LOF marked too many points as outliers, although it was able to identify some of the outliers correctly. This can be corrected by increasing the threshold, however even with the current threshold some extreme outliers are not detected. In this processes, extreme outliers were caught using the variable itself alone, while the addiction of predictors did not improve the results.

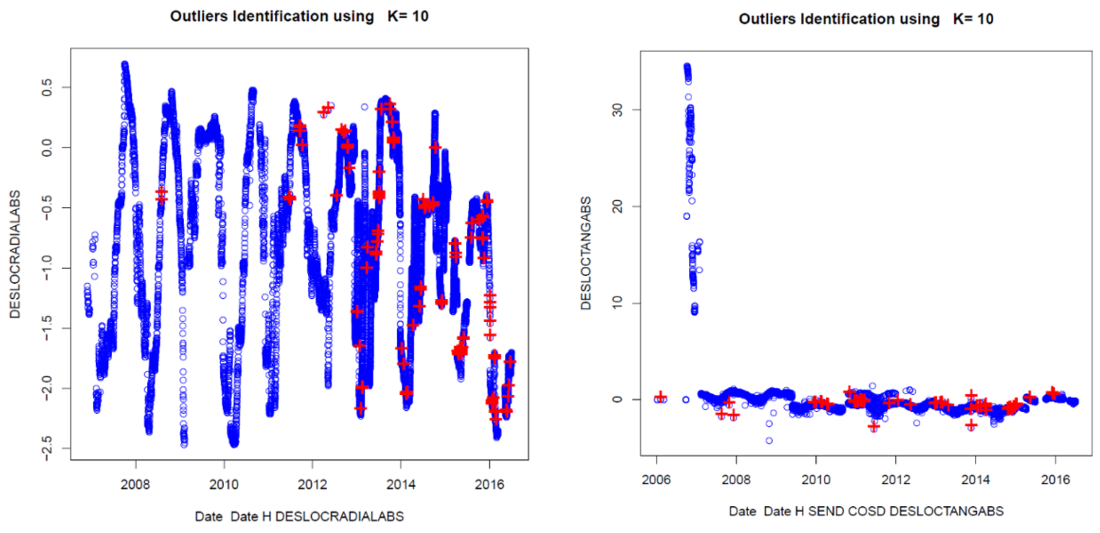

Figure 3.15: Outlier Detection using LOF: Best Results

In general, LOF was not able to identify extreme outliers, misclassifying too many points and failed to catch most of the outliers. For example, for the datasets in Figure 3.15, none of the experiments classified correctly the 4 outliers near 2012 in the left plot and none were able to identify the extreme outliers in the right plot.

3.4.4 Clustering Analysis Summary

We tested three different algorithm (K-Means, DBSCAN and LOF), showing that we are able to identify outliers that baseline methods could not. These algorithms behave differently depending on the dataset, the variable and the predictor variables used. Below we have a small comparison between the three, based on the results obtained in the four showed datasets (1/4 means it was able to catch outliers in one of the four datasets).

Table 3.6: Clustering Analysis Summary

	K-Means	**DBSCAN**	**LOF**
Outlier Classification	Manual	Automatic	Manual
Extreme Outliers Caught	4/4	4/4	3/4
Other Outliers Caught	3/4	4/4	1/4
Time	Very Fast	Slow	Slow

Analysing the various datasets and results we found several observations: (1) most datasets have a group of outliers in the first/second year of analysis, which is probably due to the implementing, testing and calibration phase of each sensor; (2) most of the extreme outliers can be found utilizing Baseline Techniques; (3) some analysis classified too many outliers (low precision); (4) after removing outliers from the datasets and zooming, we can still observe some outliers (low recall); (5) the predictive variables we used sometimes improved the algorithms, while other times did not. This can be due to several reasons: the variable is not a good predictor or the scale and deviation of the variable affects the algorithm.

To address the problems identified above, the following techniques/solutions are applicable:

- Before each analysis remove any observation that occurred in the first/second year of analysis. Since we have a large dataset (most of the analysis began in 2006, so removing up to 2 years will not matter, since we have at least 10 years' worth of data);
- Before running the machine learning algorithms, we can take a first look at the data, removing any extreme outliers found by the baseline techniques (such as Whiskers Plot box or Standard Deviation) or other algorithms;
- Although DBSCAN classification is done by the algorithm itself, K-Means and LOF are depending on a threshold. We found a suitable one, but we can improve by utilizing an expert defined threshold. Another problem may come from the parameters itself (like the number of neighbors in LOF), that should be adapted to each dataset;
- After removing the outliers with the first algorithm, we can utilize another algorithm or the same with different parameters to try and catch outliers

that were not caught and identified before. This can be useful in cases with lots of outliers, like Figure 3.12;

- For each variable and algorithm we should make a deeper analysis of how each predictor variable impacts the results. One option is to scale or normalize each predictor, giving or removing their weight on the results. Another solution is using PCA, allowing several predictors to be collected into a small number, but with bigger impact variables, however, PCA is normally used on scenarios with a large number of variables, but in our case we should not have more than five variables.

3.5 Algorithm for Outlier Detection (Using MDAS)

In this project, we are presenting a new Outlier Detection algorithm that makes use of the Manual Data Acquisition System (MDAS) measurements. The main concept of the algorithm is to validate the Automatic Data Acquisition System (ADAS) measurements by comparing them with the closest MDAS measurements. This is possible due to the following:

1. MDAS measurements are considered to be of quality (the acquisition process includes a manual validation of the value before being available on GEST-BARRAGENS);
2. Despite having less acquisition frequency, the MDAS has enough data to represent the response variable behavior. ADAS measurements should follow the same behavior as the corresponding MDAS sensor;
3. ADASs are in place to compliment the MDAS, not to replace them. MDAS should always exist so that we are able to validate the ADAS measurements and dam behavior responses (See Figure 3.16).

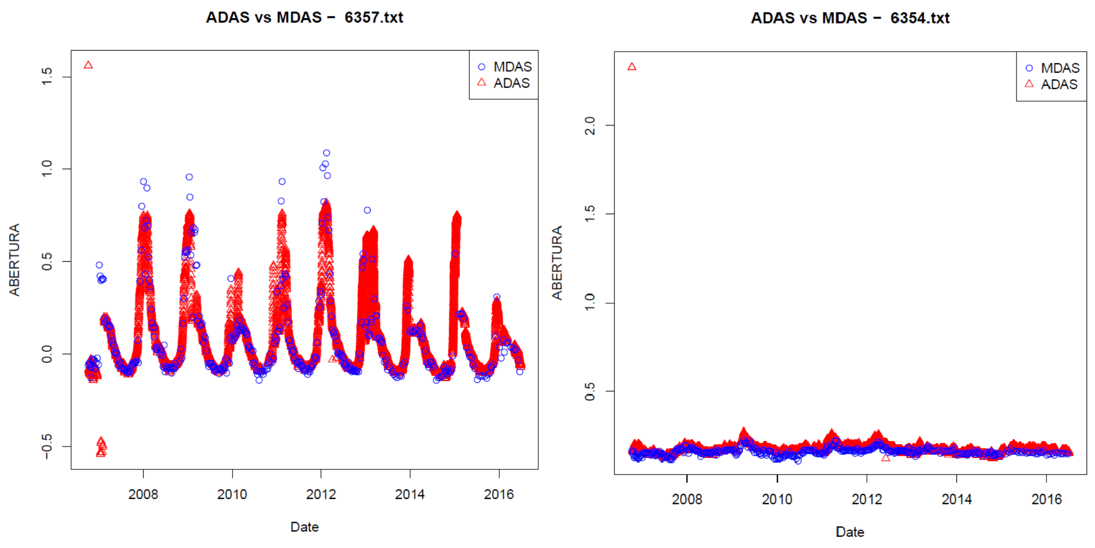

Figure 3.16: ADAS vs MDAS Example

The proposed algorithm comprises the following steps:

1. Given an ADAS dataset, obtain the corresponding MDAS dataset. The information about this relation is already present in the ADAS sensors information.
2. Identify, for each measurement, the closest MDAS measurement by using the timestamp from the records. In case of finding more than one, either one can be chosen, but in this case we chose the earliest record.
3. Identify a gap where the relation of the ADAS measurements and MDAS measurements are representative of the data. Should be a full period and as outliers-free as we can find. We have two ways to define the limits (index or time) and two ways to find them (manually or automatically (e.g. by looking for intervals with less distance between ADAS and MDAS measurements in the data)).
4. Compute the distance between the values from the ADAS and the closest MDAS measurements in the chosen gap.
5. Obtain the mean distance between ADAS and MDAS measurements in the chosen gap.

6. Define a threshold (we can have more than one threshold if we want to differentiate extreme outliers from others). This threshold should be a measure of standard deviation from MDAS values or the distances obtained in d).

7. Classify as outliers any ADAS measurements values that are outside the defined limits (see Equation 3.1). Another option is to see how far the ADAS values are from the expected value, and we can look at this as an outlier score (see Equation 3.2).

$$ADAS_{Values} - mean(Distances) \notin [MDAS_{Values} \pm Threshold] \tag{3.1}$$

$$ADAS_{Values} - mean(Distances) - MDAS_{Values} = Outlierness \tag{3.2}$$

To better explain the algorithm concept we can look at the example in Figure 3.17. Although it does not occur in our dataset, sometimes ADAS sensors have an offset from the corresponding MDAS sensor. In blue we have the ADAS values while MDAS values are represented as orange points. Green points are ADAS values used for measure distance against the MDAS values in the same gap (limited by the red lines).

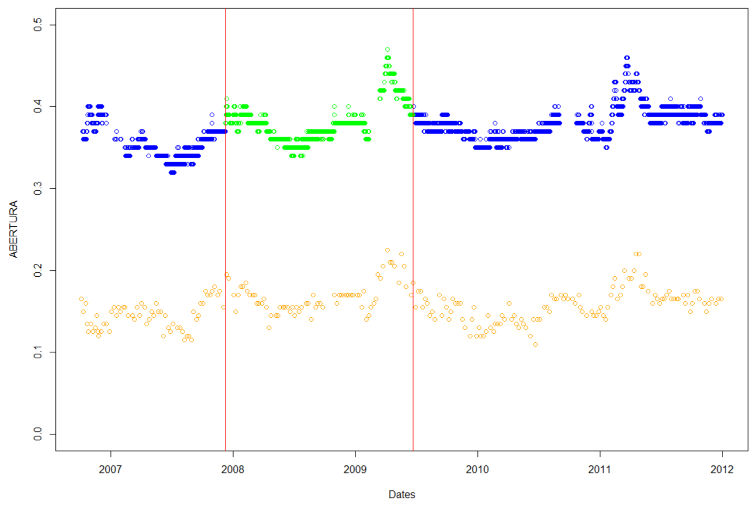

Figure 3.17: Choosing a Gap for the algorithm

To test this algorithm we have chosen a dataset from CaseStudy Dam with some visible outliers and varied the threshold. As stated in step 6, we have used the standard deviation from the distances and the MDAS values, and multiplied this value from 5 to 2 (K).

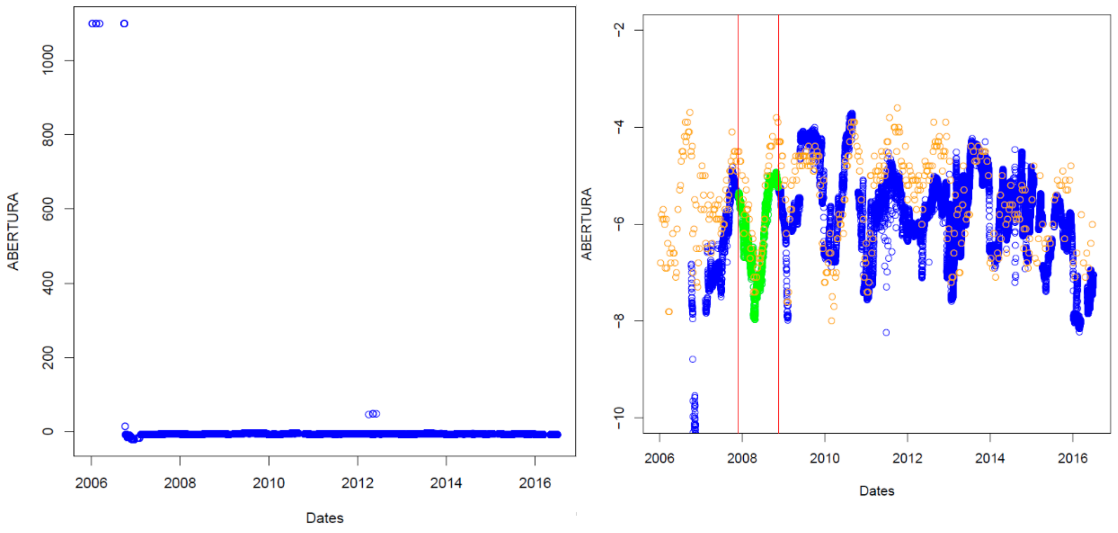

Figure 3.18: For demonstration purpose, we have chosen the dataset in the left, and the gap represented in green in the right plot

As displayed in Figure 3.18, we select a gap (in green) where we calculated the mean distance between ADAS and MDAS measurements. When using the standard deviation from the distance measures, even when multiplied by 4 or 5 (K value), we classified a big number of records as outliers. Since the first option had a bad performance, we tested the Standard Deviation from manual measurements, which delivered better results, as seen in Figure 3.19.

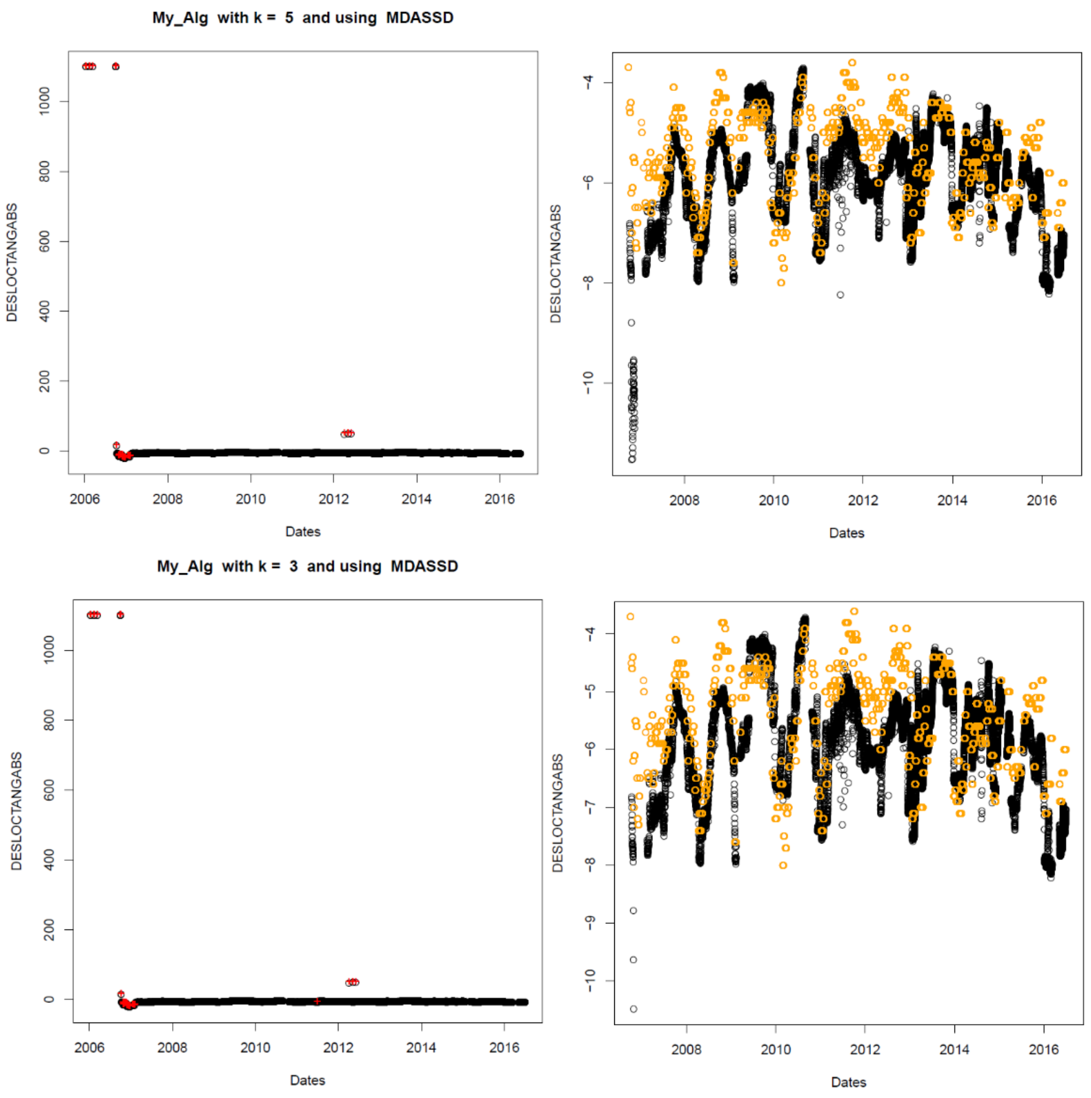

Figure 3.19: Results and Outlier Removal (right) from the algorithm with the Standard Deviation from the Manual Values and K= 5 and 3

When using the standard deviation from the MDAS values we obtained some good results, but from K=5 to K=3, the algorithm could not detect all visible outliers as we can see by the right images (after outlier removal). With K = 2 we were able to detect all outliers, however we detected some points that may not be outliers, as seen in Figure 3.20.

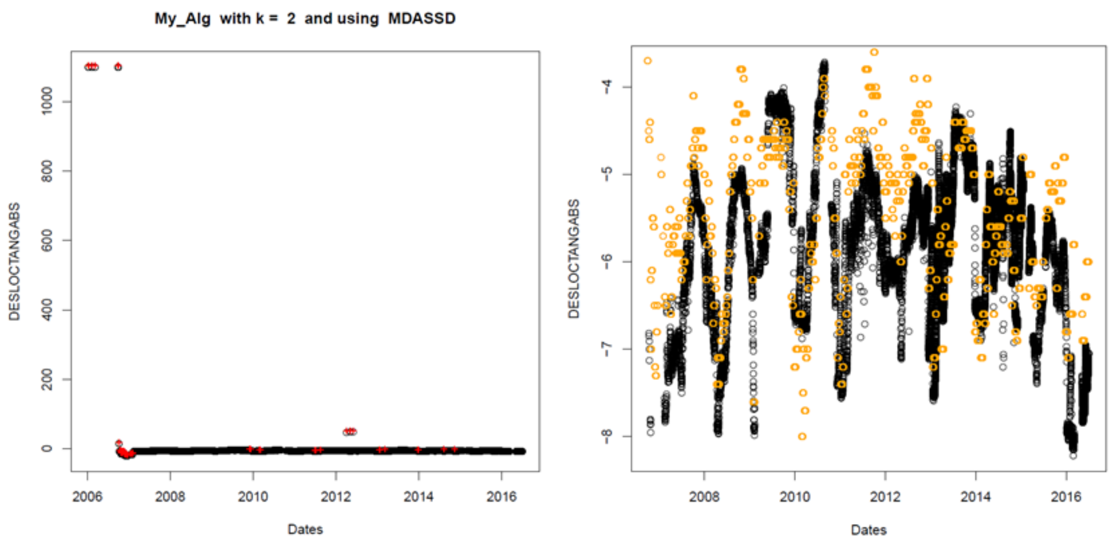

Figure 3.20: Results and Outlier Removal (right) from the algorithm with K = 2

We presented an algorithm that uses Manual Data Acquisition System measurements to validate the ADAS values. This proved useful in one dataset and especially to remove extreme outliers. Since it does not use any other information besides the variable itself, it can be used to remove with a certain confidence the extreme outliers, leaving the other outliers to be caught by other algorithms.

3.6 Method for Outlier Identification and Treatment

As we saw until now, none of the algorithms or techniques are perfect for outlier removal. Some of them are great when dealing with Extreme Outliers (as for example Whiskers Plots and Standard Deviation), while others are able to detect outliers based on previous experience from the predictors (like DBSCAN and other clustering algorithms).

Since DBSCAN is the algorithm that best performed (based on the experiments in the clustering section, see the clustering summary in Section 3.4.4), and the outlier identification is done by the algorithm itself, we will use it to catch the smaller outliers. But as we concluded in the solution, before we use the clustering algorithm we should use an algorithm or technique to remove the Extreme Outliers first. We tested two options: the standard deviation and the algorithm we presented in the solution (Algorithm for Outlier Detection (Using MDAS)). While both are useful to identify extreme outliers and the results were similar, we opted to use our Algorithm first, then DBSCAN and in the end use the Standard Deviation with a predictive model to catch any of left out outliers. Since the MDAS algorithm is based on a dataset with validated (manual) data, we will use it to make the first approach, allowing us to observe problems that Standard Deviation alone would not catch due to being single variable.

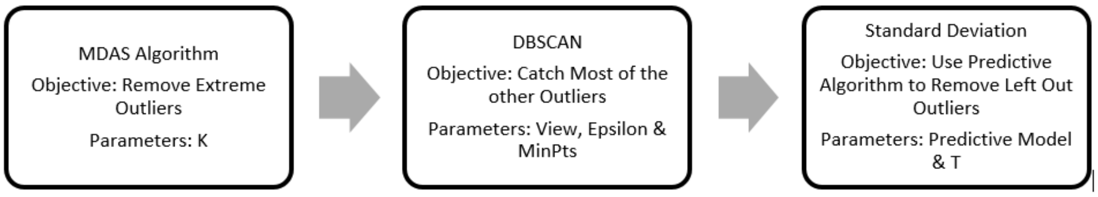
MDAS Algorithm
Objective: Remove Extreme Outliers
Parameters: K
DBSCAN
Objective: Catch Most of the other Outliers
Parameters: View, Epsilon & MinPts
Standard Deviation
Objective: Use Predictive Algorithm to Remove Left Out Outliers
Parameters: Predictive Model & T

Figure 3.21: Method for Outlier Identification and Treatment

Table 3.7: Method Parameters

Parameter	**Algorithm/Technique**	**Observation**
K	MDAS Algorithm	Used for obtaining the Threshold
View	DBSCAN	Matrix with the information about the variable plus the predictors
Epsilon	DBSCAN	-
MinPts	DBSCAN	-
Predictive Model	Standard Deviation	We will use Multiple Linear Regression with SEND, COSD & H4, however we can use other predictive algorithm and predictors
T	Standard Deviation	Used for obtaining the Threshold

We will use this method in the next chapter. With it, we expect to remove most or all of the outliers in a dataset.

3.7 Comparative Model

When comparing classification algorithms to infer which performed better we must use the metrics available in the Confusion Matrix, as well as Error Rate, Accuracy and other measures that allow us to retrieve information about the algorithm behavior. In this sense we will use the following metrics, adapted to our case (keep in mind that we obtain the Confusion Matrix (CM) by comparing the results with the true labels in the dataset):

Table 3.8: Classification Metrics

Metric	Type	Description	Obtained by	Observation
True Positive (TP)	Numeric	Outliers correctly identified	Directly retrieved from the CM	-
True Negative (TN)	Numeric	Non-Outliers correctly identified	Directly retrieved from the CM	-
False Positive (FP)	Numeric	Non-Outliers misclassified	From the CM	Correct information that may be deleted
False Negative (FN)	Numeric	Outliers misclassified	From the CM	Outliers that were not caught by the algorithm
Precision (P)	Percentage	Positive results that are actual Outliers	Equation (2.3)	Low Precision -> Algorithm identified too many points as Outliers
Recall (R)	Percentage	Outliers correctly identified	Equation (2.4)	Low Recall -> Algorithm did not caught enough Outliers

None of the metrics presented in Table 3.8 by itself is a good identifier of the algorithm behavior. For example, when looking at a 100% Recall one may conclude that the algorithm is behaving well, however if the same algorithm has a low Precision we can conclude that is classifying too many points as Outliers.

The reverse can also be observed. A 100% Precision may lead us to think that the algorithm did well, however we can obtain that value if the algorithm only classified (correctly) one point as an Outlier. When looking at the Recall we would see a very small value, giving us a better view of the algorithm performance. With this in mind, we can combine these metrics into the metrics presented in Table 3.9

Table 3.9: Classification Metrics

Metric	**Type**	**Description**	**Obtained by**
Accuracy	Percentage	Correctness of the Algorithm	Equation (3.3)
F1-Measure (F1)	Numeric	Measure that combines Precision and Recall	Equation (3.4)
F2-Measure	Numeric	Measure that combines Precision and Recall, giving more weight to Recall	Equation (3.5)

$$Accuracy = \frac{TP + TN}{TotalPoints} \tag{3.3}$$

$$F1_Measure = 2 * \frac{P * R}{P + R} \tag{3.4}$$

$$F2_Measure = 3 * \frac{P * R}{2 * P + R} \tag{3.5}$$

The metrics presented in Table 3.9 allow us to have a good insight about the algorithm behavior, by combining several of the previous metrics from Table 3.8. However we have to be careful about the accuracy, since we can expect high values of accuracy in our algorithms. This is due to the small percentage of outliers in the very large datasets. Even if we have 10% of Outliers, if the algorithm fails to identify any as Outliers, we still have a 90% of accuracy which can be seen as a misleading value. F1-Measure is the one o f the most important metrics since it correlates the Precision and the Recall, giving us a broader evaluation of the algorithm performance.

Since our objective is to identify correctly most of the outliers, we can use the F2-Measure (derived from F1-Measure). Since F-Measure is a harmonic mean, we can remove weight from the Precision metric and, therefore, give more weight to the Recall. Although we do not have to remove correct information from the data, since we have a huge number of observation we can remove some observations without losing information about the variable behavior.

Chapter 4

Demonstration

This Chapter shows the impact of both the Baseline and the Method for Identification and Treatment of Outliers presented in Section3.6 on a labeled dataset. Now that we have a good insight about the algorithms and techniques that can be used for Outlier Detection and Treatment, we will demonstrate their use on one of the datasets. This Chapter is structured as follows:

- Section 4.1 shows the creation and labeling of our dataset. We will use a dataset from our case study and manually introduce the outliers, with the respective label;
- Section 4.2 demonstrates the impact of the baseline method in our dataset;
- Section 4.3 demonstrates the impact of the the method introduced in 3.6 in our dataset.

4.1 Dataset creation and labeling

Since none of the available datasets are labeled, we do not have any information about outliers present in them. To be able to compare and infer about the performance of any of the methods, we will need labeled datasets. Due to the fact that manual labeling is slow and prone to error, and to the difficulty of identifying outliers in multi-dimensional data, we opted for introducing the outliers in the datasets ourselves. This process was supervised by a dam safety expert from

LNEC, to ensure that we were obtaining the most realistic labeled dataset possible. After a meeting with the expert, we opted to use the dataset from a specific plumb line sensor, identified by number 6251 with the *DESLOCRADIALABS*[1] variable. This dataset was chosen due the lack of visible outliers. To obtain the new datasets we made the following changes to the original dataset:

1. Cut every observation before 2007 due to the existence of some outliers (that were originated from sensor installation and calibration phase);
2. Introduced information about the predictors (Water Level, Temperature and Time). Besides the information used in the solution, we also scaled and normalized some of the values, so that we have more predictive set available;
3. Inserted Outliers and Labels (More information below).

After these changes we obtained the dataset presented in Figure 4.1.

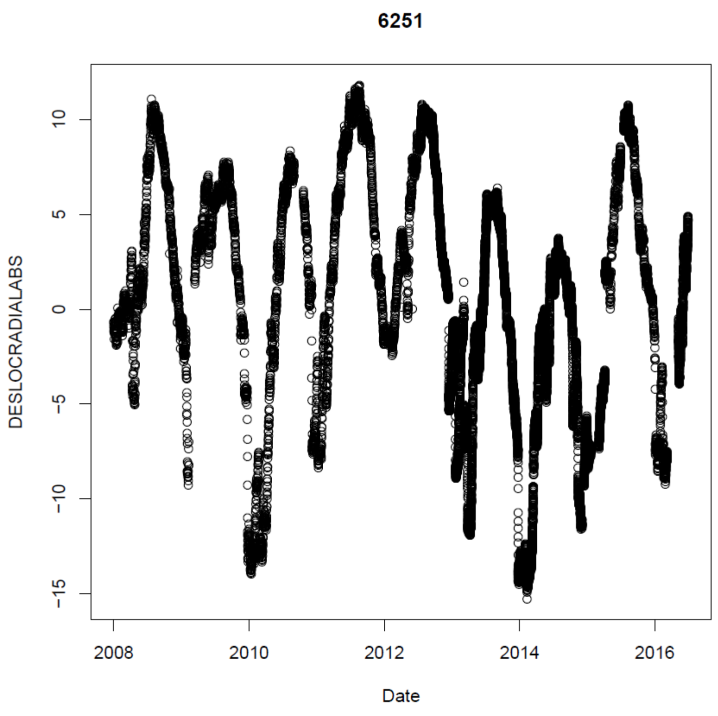

Figure 4.1: Dataset chosen for Demonstration)

The outliers were created by inputting an offset (positive or negative at random) in the response variable values. Since we have several types of outliers, we

[1]Radial Displacement

chose to create outliers with different offsets, so that each of the labels correspond to real life situations. For example, while types 4 and 5 are considered extreme outliers, type 1 offset is so little that the values are still inside the sensor margin of error, so it should be very difficult to be detected. We have the following types of labels (see Table 4.1):

Table 4.1: Outlier Types Created

Label	**Outlier**	**Offset (% of Peak-to-Peak Amplitude)**
0	No	None
1	Yes	5
2	Yes	10
3	Yes	25
4	Yes	50
5	Yes	100

We then created three distinct datasets, each one with a different percentage of outliers. With the help of the expert, we decided to create datasets with 1% (Figure 4.2), 5% and 10% of outliers. After we decided the percentage of outliers, that percentage is divided in the five types of outliers and the dataset is created as seen in Table 4.2.

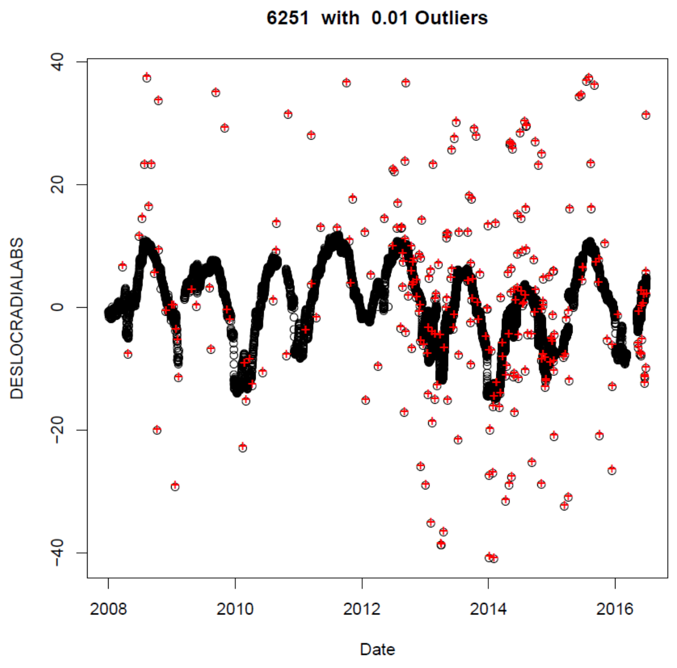

Figure 4.2: Dataset with 1% of Outliers

Table 4.2: Dataset with 1% of Outliers Label Information

Label	Number of Observation (#)	Percentage of Observation (%)
0	27094	99%
1	54	0.2
2	55	0.2
3	55	0.2
4	55	0.2
5	54	0.2
Total	27367	100

4.2 Baseline Method

As seen in Section 3.3, Dam experts use standard deviation with the help of a MLR model to define thresholds in order to identify outliers in the datasets. We created a MLR model for each dataset (1%, 5% and 10% of outliers) using Temperature(COSD and SEND) and Water Level(H4) as predictors. In tests, we used the standard deviation times 1 to 5, and observed the results. The best F2-Measurements in each of the three datasets was obtained with the standard deviation times 1 (T=1), which we show below for each one of the datasets Figures 4.3, 4.4 and 4.5. Note that even with T=1, some of the outliers still went uncaught. As expected, in the tests with higher threshold values this situation is even more visible (the larger the threshold, higher the number of outlier uncaught).

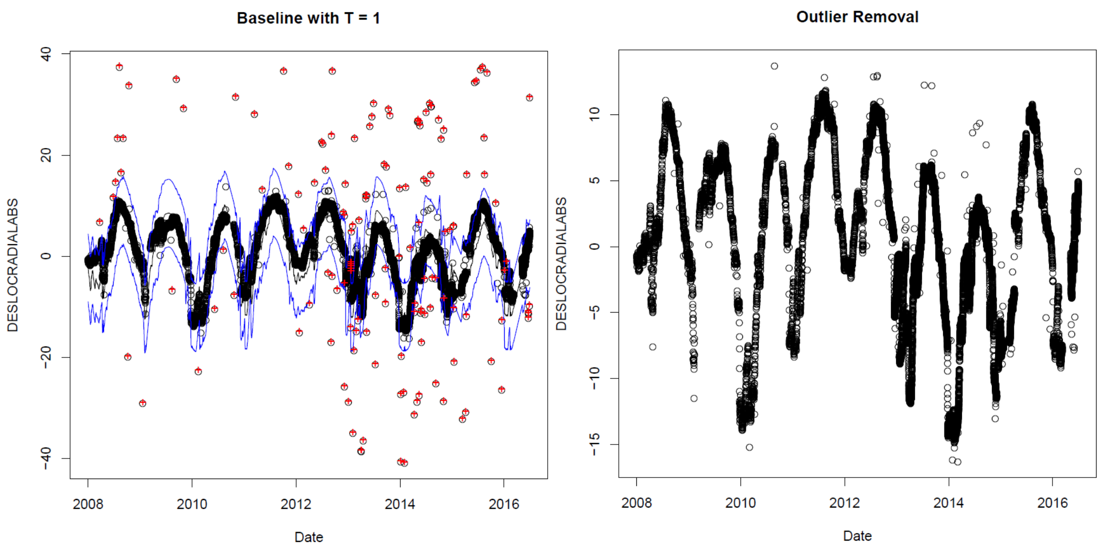

Figure 4.3: Baseline Method on Dataset with 1% of Outliers. Left - Outlier Identification with T=1; Right - Outlier Removal

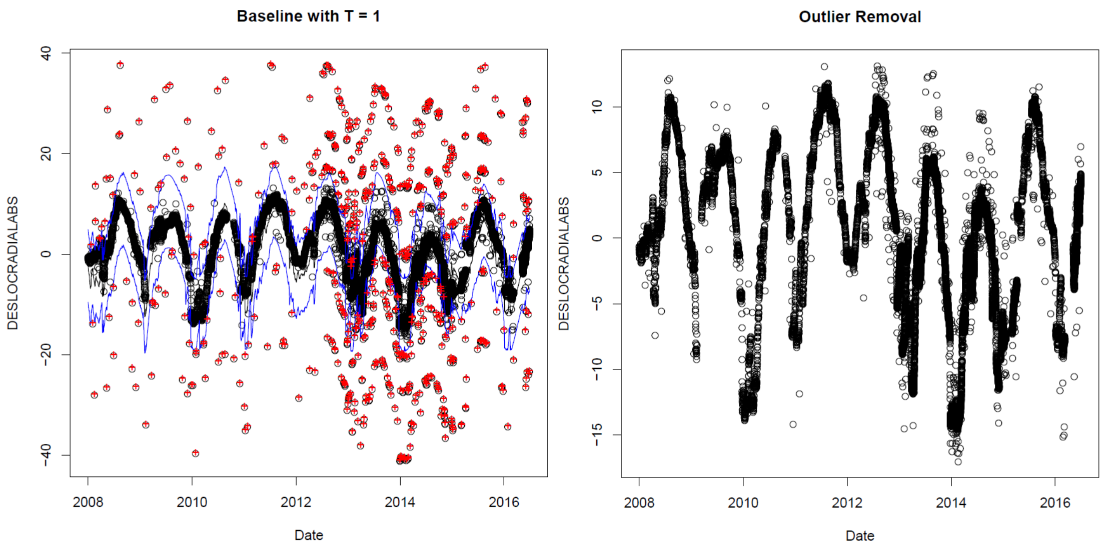

Figure 4.4: Baseline Method on Dataset with 5% of Outliers. Left - Outlier Identification with T=1; Right - Outlier Removal

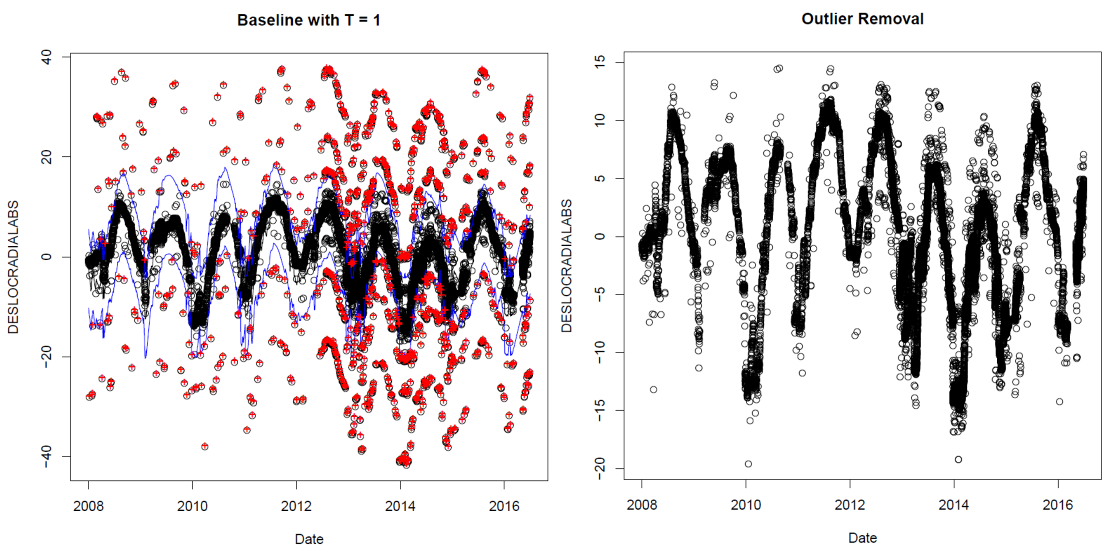

Figure 4.5: Baseline Method on Dataset with 10% of Outliers. Left - Outlier Identification with T=1; Right - Outlier Removal

We can see that higher percentage of outliers leads to more outliers uncaught, as expected (we can see more uncaught outliers in Figure 4.5 than in Figure 4.3). With more outliers, the statistical information about the data (including the mean and the standard deviation) is corrupted. Therefore, and since this method depends not only on the predictive model (which could be improved) but also on statistic values that are "incorrect", the performance decreases.

4.3 Method for Outlier Identification and Treatment

In this section we demonstrate how the method presented in Section 3.6 behaves when applied in our datasets. The method can be decomposed in three different algorithms, so we will analyze the individual impact of each algorithm in the datasets. This Section is organized as follows:

- Section 4.3.1 presents an analysis of preliminary tests we did to select the parameters, before using the method in the datasets;
- Section 4.3.2 shows the impact of the algorithm presented in 3.5 (Outlier Detection using MDAS measurements) in each of the three datasets;

- Section 4.3.3 shows the impact of DBSCAN algorithm in the datasets;
- Section 4.3.4 shows the impact of Standard Deviation algorithm in the datasets.

4.3.1 Preliminarily Tests

Before we analyze the impact of the presented method, we tested in batch mode the variation of parameters (see Table 4.3) and their impact on the datasets. Besides the following variations, we tested scaling the matrix and normalizing the response variable for DBSCAN. In total we made more than 50000 different analysis.

Table 4.3: Testing Parameter Variation

Parameter	Algorithm/Technique	Variation
K	MDAS Algorithm	1:5
View	DBSCAN	12 different combinations of predictors
Epsilon	DBSCAN	0.1, 0.3 & 0.5
MinPts	DBSCAN	2, 4, 6, 8, 10
Predictive Model	Standard Deviation	MLR with SEND, COSD & H4
T	Standard Deviation	1:5

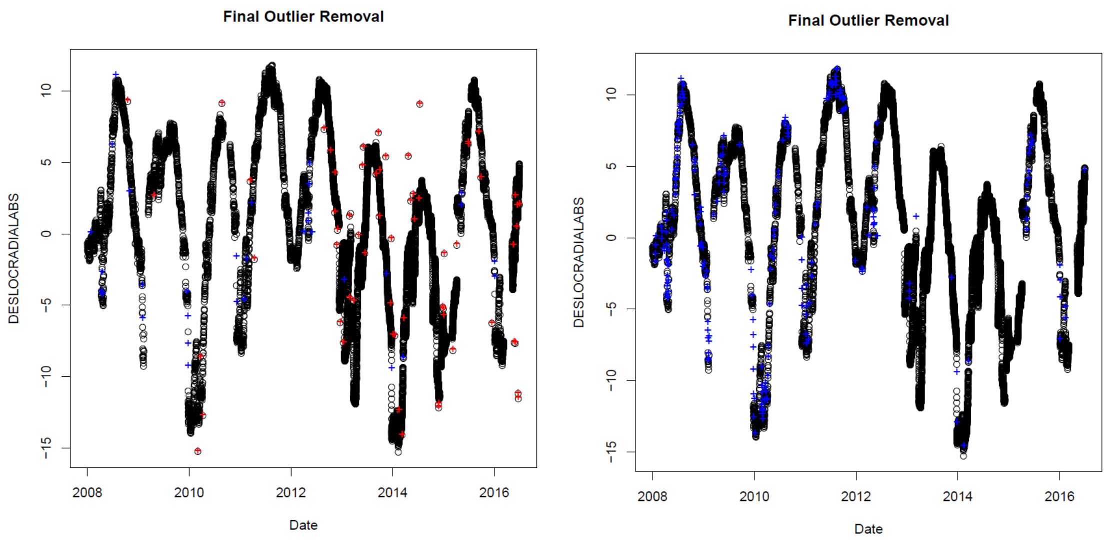

Figure 4.6: Best $F1$-Measure vs Best Recall Results

To observe which combination of parameters performs best in each dataset we will use the $F2$-measure. Above (Figure 4.6) we have a comparison between one result with the best $F1$-measure and Accuracy and the one with the best Recall (R=1, so all outlier were caught) from the dataset with 1% of outliers. As we can see, while the first delivered good results we still have some outliers that we may wish to treat, however the second one removes a lot of points (misclassified removed points are labeled as blue crosses). To calibrate more the search towards our objective, we use the $F2$-Measure that gives more weight to Recall without disregarding the Precision.

In most cases (except the dataset with 1% of outliers) neither the scale nor the normalization of the response variable in the matrix utilized in the DBSCAN algorithm produced better results. In the case of the MinPts, we expected to obtain better results with higher values, due to the multi variable setting, however we obtained better results when MinPts where equal to 2. The best epsilon was also the lowest one tested (0.1). From the tested predictive sets (see Table 4.4) we can observe that scaling the temperature did not produce better results, however normalizing the days (T) was a good change. One thing that may seem strange is that using no predictors produced the best mean accuracy, however the worst F-Measure (again, not identifying any outliers leads to high Accuracy, e.g., in the 10% outlier dataset if we fail to detect any outlier, we still have a 90% accuracy).

Table 4.4: Results per PredSet (Mean Values from all different analysis)

PredSet	F2-Measure	Accuracy	Predictor
4	0.5221522	83.31290	COSD SEND H
9	0.5153264	82.14180	SCALEDCOSD SCALEDSEND NORMH
6	0.5011212	86.11785	NORMT NORMH
12	0.4974072	78.68506	SCALEDCOSD SCALEDSEND H NORMT
10	0.4958177	78.26938	SCALEDCOSD SCALEDSEND NORMH T
11	0.4958177	78.26938	SCALEDCOSD SCALEDSEND H T
5	0.4913435	84.43801	NORMT H
8	0.4689577	78.64226	SCALEDCOSD SCALEDSEND H
3	0.4424414	78.13292	T H
7	0.4424414	78.13292	T NORMH
2	0.4423763	84.86358	H
1	0.4061059	87.82597	None

4.3.2 Outlier Detection using MDAS measurements

The first step of our method consists in removing extreme outliers from the dataset. To do so we used the presented algorithm in our solution (see 3.5). To train the algorithm we used 1000 observations (between index 1000 and 2000), and used as threshold the standard deviation from all MDAS measurements times K (K was varied between 1 and 5). In this demonstration, we will use K = 2 for datasets with 1 and 10% (Figures 4.7 and 4.9) and K = 3 for dataset with 5% of outliers (see Figure 4.8).

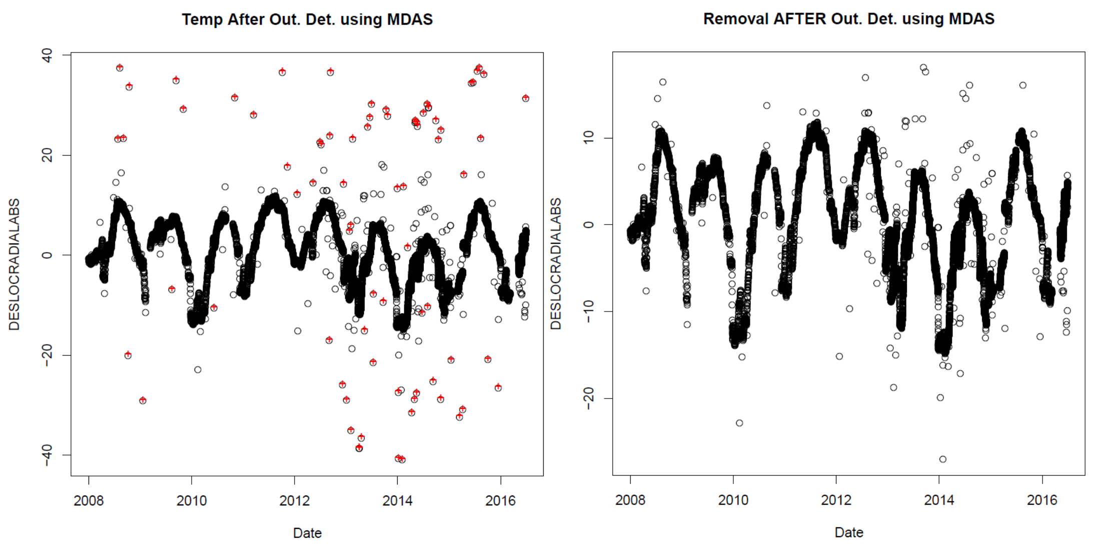

Figure 4.7: Outlier Detection using MDAS impact on Dataset with 1% of Outliers. Left - Outlier Detection with K=2; Right - Outlier Removal

Remember that this method is utilized as a step to remove extreme outliers from the datasets, and therefore it is not our objective to remove all or most of the outliers here. We could even use a lower K to tighten the threshold, however we may mislabel some of the points, and since the next step (DBSCAN) uses more information (Water Level, Temperature, etc.), we will leave most of the outliers for it to detect.

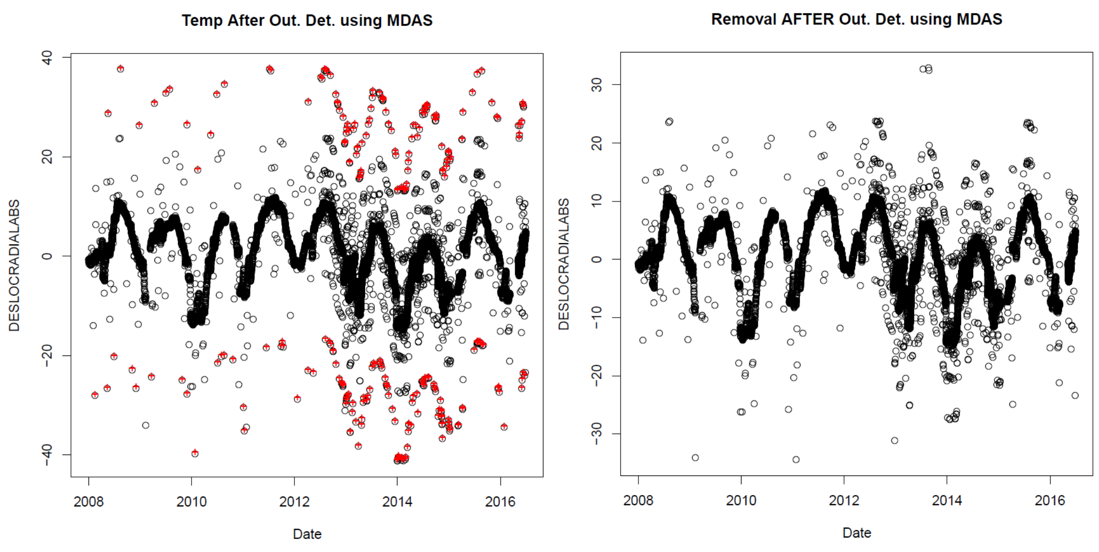

Figure 4.8: Outlier Detection using MDAS impact on Dataset with 5% of Outliers. Left - Outlier Detection with K=3; Right - Outlier Removal

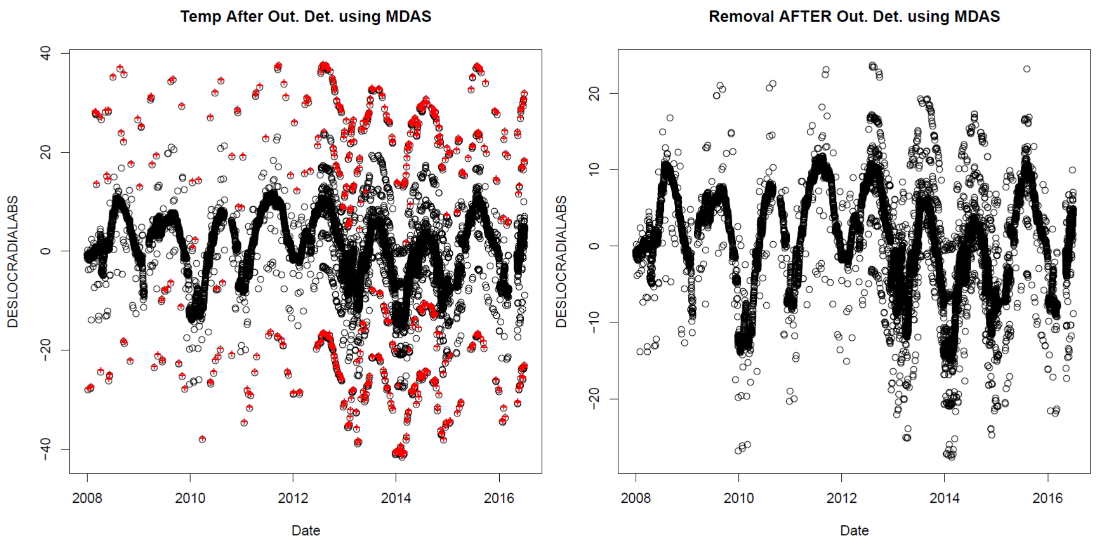

Figure 4.9: Outlier Detection using MDAS impact on Dataset with 10% of Outliers. Left - Outlier Detection with K=2; Right - Outlier Removal

4.3.3 DBSCAN

After removing the outliers identified by the previous algorithm, we used DBSCAN to identify the remaining outliers. To do so, we deliver to DBSCAN not only information about the response variable but also the predictors. This is the main step of our method, and it is here that we try and catch most of the outliers. We will use the following parameter values in this demonstration (see Table 4.5) and show the impact of the algorithm in the datasets.

Table 4.5: DBSCAN Parameters

Dataset	PredSet	EPS	MinPts	Norm	Scale
1%	12	0.1	2	FALSE	TRUE
5%	3	0.5	2	FALSE	FALSE
10%	3	0.5	2	FALSE	FALSE

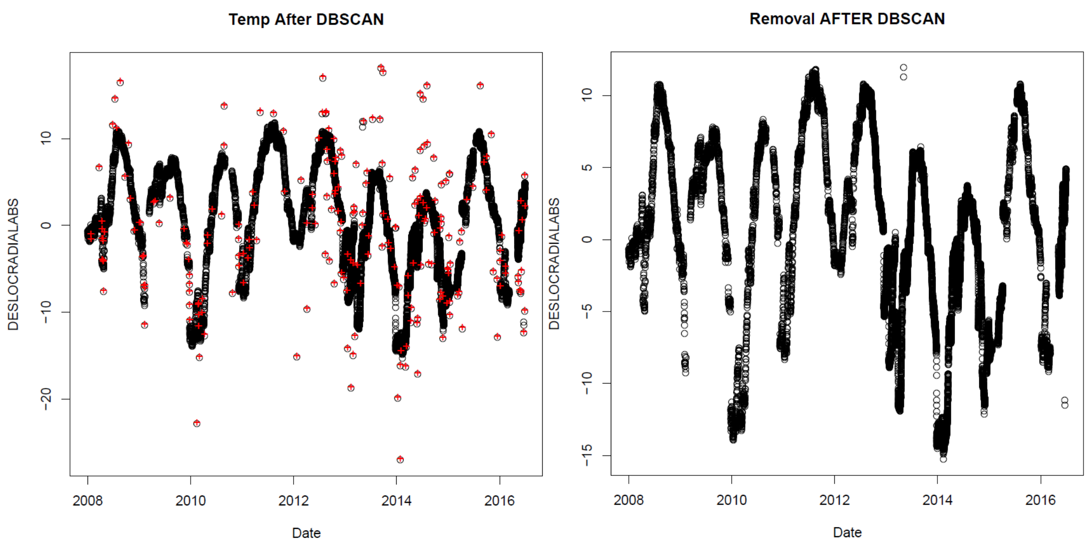

Figure 4.10: DBSCAN impact on Dataset with 1% of Outliers. Left - Outlier Detection with DBSCAN using the parameters showed in table 4.5; Right - Outlier Removal after DBSCAN algorithm

The algorithm was able to identify most of the outliers in the dataset with 1% of outliers (Figure 4.10). Remaining visible outliers are paired, so they probably formed a cluster (since MinPts parameter is set to 2). Since the cluster are multi-variable, and not done in a 2D, we can see isolated outliers in the dataset below.

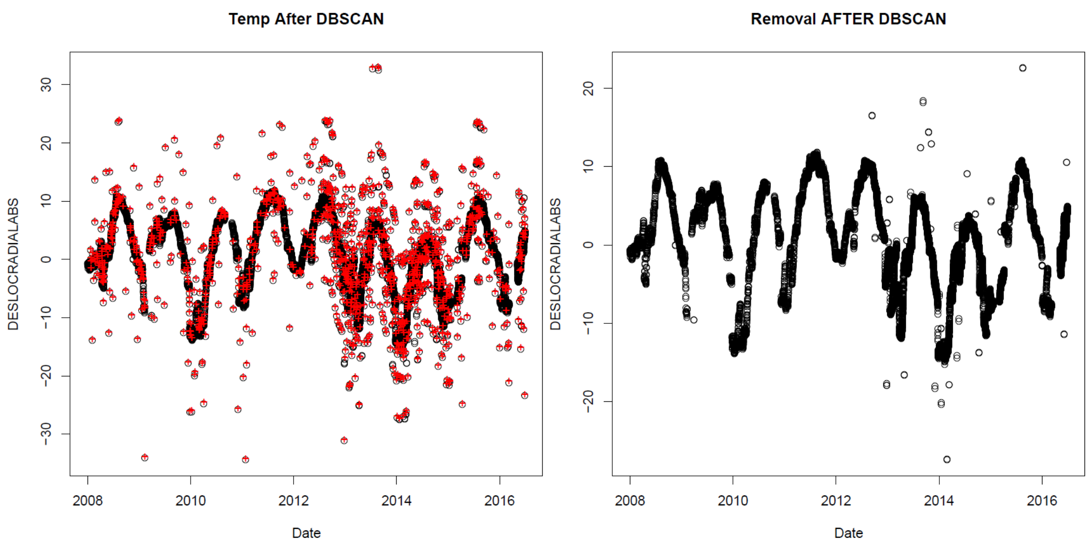

Figure 4.11: DBSCAN impact on Dataset with 5% of Outliers. Left - Outlier Detection with DBSCAN using the parameters showed in table 4.5; Right - Outlier Removal after DBSCAN algorithm

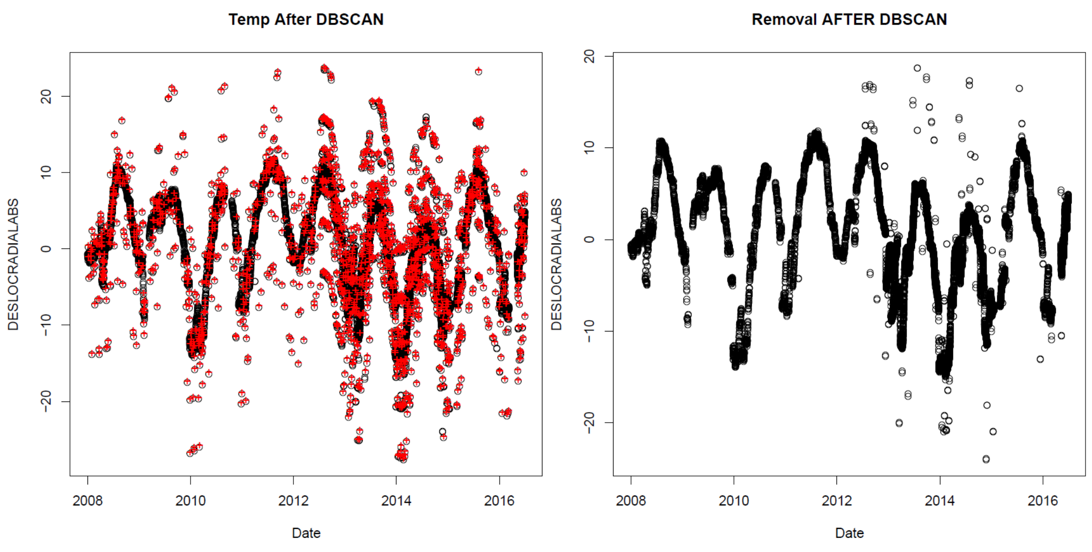

Figure 4.12: DBSCAN impact on Dataset with 10% of Outliers. Left - Outlier Detection with DBSCAN using the parameters showed in table 4.5; Right - Outlier Removal after DBSCAN algorithm

We can observe that with a higher number of outliers the algorithm performance is reduced (we can see a big difference in the number of uncaught outliers between Figures 4.10 and 4.12). Another observation that we can make is that the algorithm labeled correctly most or all of the outliers before 2012. Since not all outliers were caught, we use the next step of the method to refine our final result.

4.3.4 Standard Deviation

The final step of our method consists of utilizing a Predictive Model and observe if the points are within the expected results (inside the threshold, similar to the baseline technique). We utilized a Multiple Linear Regression with temperature (COSD and SEND) and water lever (H4) as predictors. The threshold was obtained with the standard deviation of the values times T. In testing we varied T between 1 and 5.

In the dataset with 1% of outliers this step did not improve the results, and the only impact we obtained was with T=1, where this technique caught the remaining visible outlier in Figure 4.10 however mislabeled a lot of points, decreasing performance (increasing the number of misclassified non-outlier observations leads to worst precision values). In the remaining datasets we used T=1 and were able to improve the results.

We can observe the impact of the Standard Deviation with T = 1 in the 5% dataset in Figure 4.13 and in the 10% dataset in Figure 4.14.

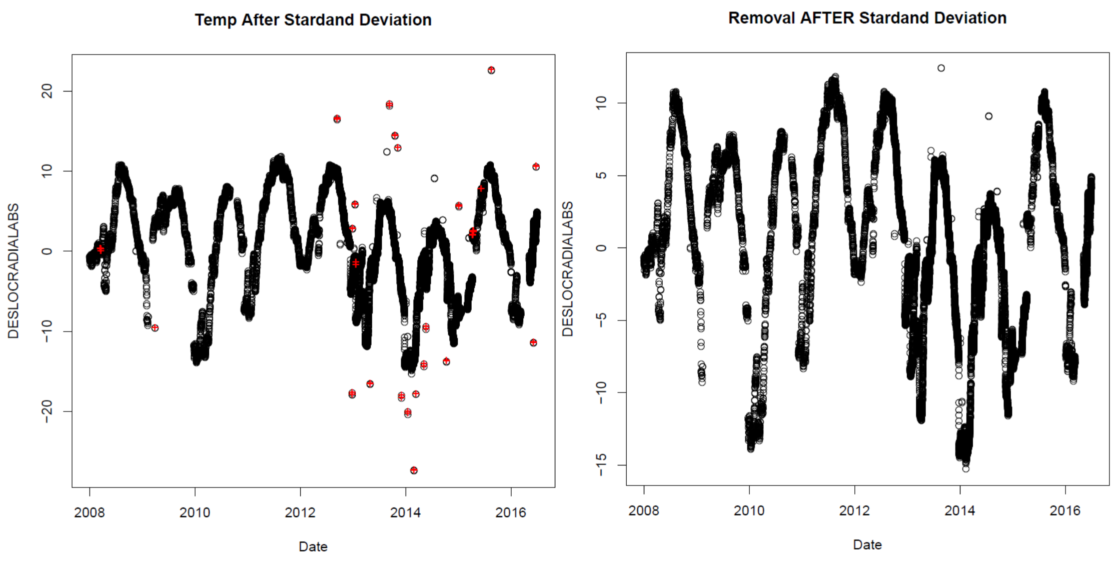

Figure 4.13: Standard Deviation impact on Dataset with 5% of Outliers. Left - Outlier Detection with Standard Deviation technique with T=1. Right - Outlier Removal

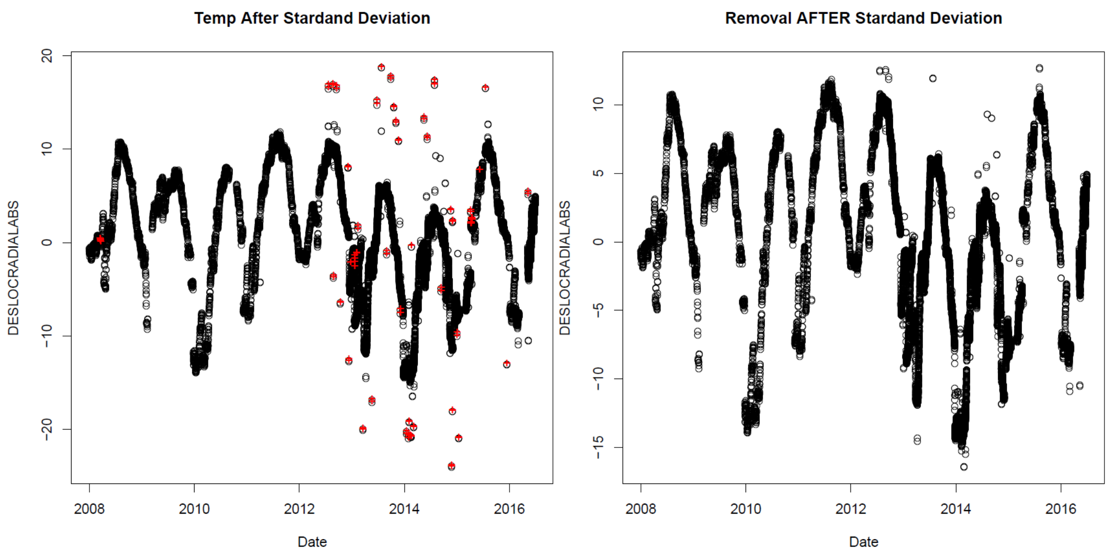

Figure 4.14: Standard Deviation impact on Dataset with 10% of Outliers. Left - Outlier Detection with Standard Deviation technique with T=1. Right - Outlier Removal

Chapter 5

Evaluation

This chapter shows and explains in detail the impact of the Baseline Method and the Method for Identification and Treatment of Outliers presented in Section 3.6 and already demonstrated in the previous chapter. We will use the metrics presented in Section 3.7, so we will analyze the confusion matrix for each of the methods and methods' steps, identified outliers (True Positives), leftover outliers (False Negatives) and misclassified non-outliers that were removed (False Positives). This Chapter is organized in the following Sections:

- Section 5.1 analyzes the performance of the baseline method in the datasets;
- Section 5.2 analyzes the performance of the Method for Identification and Treatment of Outliers presented in 3.6 in the datasets;
- Section 5.3 compares the performance between both methods.

5.1 Baseline

In the Baseline technique demonstrated in Section 4.2, we saw that this method was helpful when treating extreme outliers, however it misbehaved when dealing with other kinds of outliers. As we will see below, when analyzing the confusion matrix for each dataset this can be proven. The confusion matrix(CM) presented in Table 5.1 shows the labels (0=Non Outlier, 1 to 5=Outlier) as rows and the predicted results (FALSE=Non Outlier, TRUE=Outlier) as columns.

Table 5.1: Confusion Matrix for Baseline Results

	Dataset 1 %		Dataset 5 %		Dataset 10 %	
TYPE	**FALSE**	**TRUE**	**FALSE**	**TRUE**	**FALSE**	**TRUE**
0	27088	6	25996	3	24629	2
1	54	0	272	1	547	0
2	54	1	264	10	544	4
3	29	26	130	144	303	244
4	0	55	0	274	7	540
5	0	54	0	273	0	547

Table 5.2: Baseline Results Metrics

	Percentage of Outlier Caught					**Classification Metrics**		
Dataset	Type 1	Type 2	Type 3	Type 4	Type 5	Recall (%)	Accuracy (%)	F2
0,01	0	1.818	47.273	100	100	49,817	97.482	59,302
0,05	0.366	3.650	52.555	100	100	51,316	87.558	61,203
0,1	0	0.730	44.607	98.721	100	48,794	74.882	58,819

In Table 5.2, we can see that all type 4 and 5 outliers (extreme outliers) were caught by the algorithm (except 7 out of the 547 type 4 outliers in the 10% outliers dataset). Only about 50% of the type 3 outliers were labeled correctly, while this method, as expected, is not able to detect type 1 and type 2 outliers. In general, in the three different datasets, it was able to get 50% of all outliers (Recall), and mislabeled as outlier very few outliers (so we have a high Precision metric), which leads to a small increase (compared to the Recall) of the F2 metric to around 60% in every dataset.

5.2 Method for Outlier Identification and Treatment

In this section we will analyze the impact the method showed in Section 4.3 for each of the three datasets. We want to understand how each step of the method affects the data, collecting the metrics from the confusion metric, as well as other classification metrics as F2-Measure and Recall (percentage of caught outliers). We will analyze each of the datasets individually and then compare them together, to obtain a broader look of the method.

5.2.1 Dataset with 1% of Outliers

As we saw in the Demonstration Chapter (Section 4.3), we obtained good results when using our Method for Identifying and Treating Outliers in this dataset. As we can see below, in Figure 5.1, our method was able to identify most of the outliers, leaving only a couple of extreme outliers. Due to the fact that the four remaining extreme (type 4 or above) outliers are paired (when looked at the 2D scatter plot), we can find two reasons why they were not identified: the MDAS algorithm and the Standard Deviation fitting was not good enough (chosen parameters, predictive model, etc) or the DBSCAN combination of parameters led to this situation. Since the DBSCAN's MinPts parameter was set to two, it means that if two points are close enough (the definition of close enough is set by the Epsilon parameter, in the multi-variable setting), they form a cluster, and hence are not considered as an outlier.

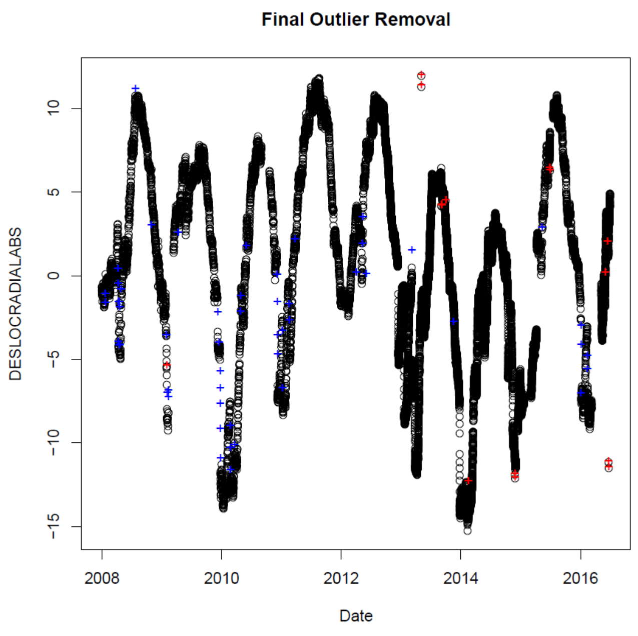

Figure 5.1: Final result in the dataset with 1% of outliers. The mislabeled outliers are pointed out with a red cross (False Negatives), while the mislabeled correct points are presented as a blues crosses (False Positives).

Table 5.3: Confusion Matrix Evolution

	MDAS Algorithm		DBSCAN		Standard Deviation	
TYPE	**FALSE**	**TRUE**	**FALSE**	**TRUE**	**FALSE**	**TRUE**
0	27094	0	27041	53	27041	53
1	54	0	11	43	11	43
2	55	0	0	55	0	55
3	55	0	0	55	0	55
4	32	23	4	51	4	51
5	0	54	0	54	0	54

Observing the Confusion Matrix presented in Table 5.3 we can see that, in this dataset, our method was unable to identify 4 of the 55 type 4 outliers, which should have been easy since they are extreme outliers. We can also see that DBSCAN was the step responsible for identifying most of the outliers, and it did extremely well, getting 55 out of 55 type 2 outliers (100%) and 43 out of 54 type 1 outliers (93%).

Table 5.4: Results Metrics Step by Step (%)

	Percentage of Outlier Caught					**Classification Metrics**		
Step	Type 1	Type 2	Type 3	Type 4	Type 5	Recall (%)	Accuracy (%)	F2
MDAS	0	0	0	41.818	100	28,205	97.289	37,079
DBSCAN	79.623	100	100	92.727	100	94,505	97.756	90,315
SD	79	100	100	92.727	100	94,505	97.756	90,315
Final	79.623	100	100	92.727	100	94,505	97.756	90,315

As we can see in Table 5.4, our method was able to correctly label 94.5% of all outliers, with a F2-Measure standing in the 90 value. Since we have a low number of outliers, the 53 False Positives (see Table 5.3) in a total of 311 identified outliers (TP+FP) lead to a rather low 83% of Precision, decreasing the F2 value.

5.2.2 Dataset with 5% of Outliers

In this dataset, unlike the previous one, the remaining outliers are not paired (at least in the 2D scatter-plot, in the multi-variable they should be). While we can observe that before 2012 all (with the exception of one) outliers were removed, at the cost off some False Positives (seen in blue in Figure 5.2), after 2012 we can see more outliers (False Negatives) and less False Positives.

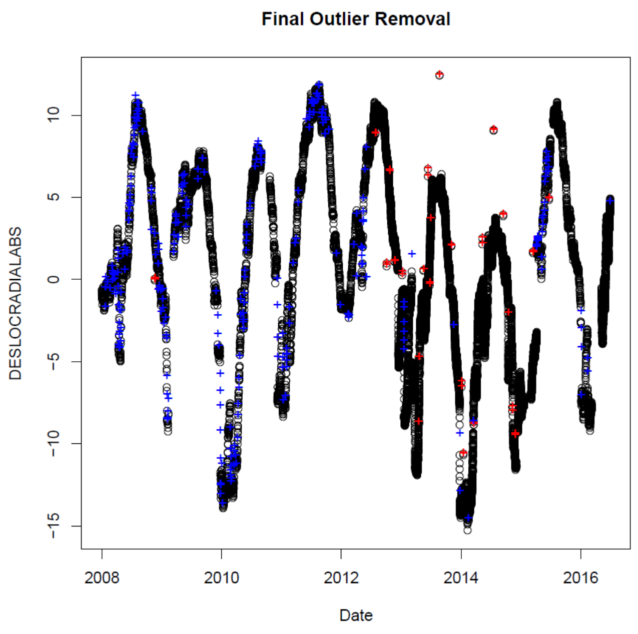

Figure 5.2: Final result in the dataset with 5% of outliers.

Table 5.5: Confusion Matrix Evolution

	MDAS Algorithm		DBSCAN		Standard Deviation	
TYPE	**FALSE**	**TRUE**	**FALSE**	**TRUE**	**FALSE**	**TRUE**
0	25999	0	25724	275	25681	318
1	273	0	17	256	17	256
2	274	0	18	256	18	256
3	274	0	26	248	12	262
4	274	0	27	247	0	274
5	9	264	0	273	0	273

In Table 5.5, we can see that the first step (MDAS Algorithm) was not able to detect most of the extreme outliers (missed 9 type 5 outliers and all type 4 outliers). Again DBSCAN proved to be the step responsible of catching most of the outliers, with a little detail: the algorithm detected with higher precision type 1 and 2 outliers (94%) than types 3 and 4 (90%).

Table 5.6: Results Metrics Step by Step (%)

	Percentage of Outlier Caught					**Classification Metrics**		
Step	Type 1	Type 2	Type 3	Type 4	Type 5	Recall (%)	Accuracy (%)	F2
MDAS	0	0	0	0	96.703	19,298	85.969	26,400
DBSCAN	93.773	93.431	90.511	90.146	100	93,567	88.676	89,495
SD	93.773	93.431	95.620	100	100	96,564	88.669	90,583
Final	93.773	93.431	95.620	100	100	96,564	88.669	90,583

With an higher number of outliers, the method was able to catch 96,5% of them, with a F2-Measure of 90,5. All of the extreme outliers were caught, and all other types got high percentage of detection (93% or more), as seen in Table 5.6.

5.2.3 Dataset with 10% of Outliers

The dataset with 10% outliers shows the same problem as the datasets from the subsections before: the results are dissimilar, having completely different results before and after 2012. In the end, we can observe that most of the outliers were removed. However we can also see that we have a lot of False Positives (removed points that are not outliers) before 2012, while the False Negatives (outliers not identified) appear only after the year 2012, as we can see in Figure 5.3.

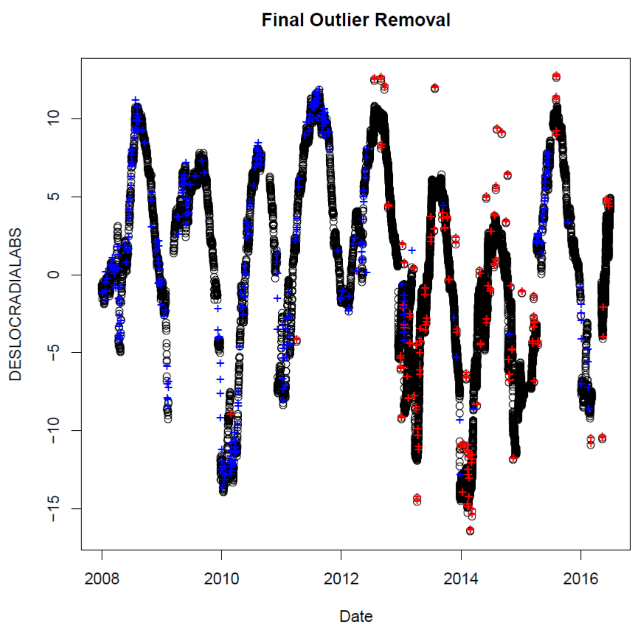

Figure 5.3: Final result in the dataset with 10% of outliers.

Table 5.7: Confusion Matrix Evolution

	MDAS Algorithm		DBSCAN		Standard Deviation	
TYPE	**FALSE**	**TRUE**	**FALSE**	**TRUE**	**FALSE**	**TRUE**
0	24631	0	24316	315	24274	357
1	547	0	92	455	92	455
2	548	0	69	479	63	485
3	547	0	67	480	30	517
4	369	178	44	503	0	547
5	0	547	0	547	0	547

In Table 5.7, we can observe that the MDAS algorithm was able to identify all of type 5 outliers, however it missed most of the type 4. In the end, the method was able to detect all type 4 and 5, and most of the other types.

Table 5.8: Results Metrics Step by Step (%)

	Percentage of Outlier Caught					Classification Metrics		
Step	Type 1	Type 2	Type 3	Type 4	Type 5	Recall (%)	Accuracy (%)	F2
MDAS	0	0	0	32.541	100	26,499	72.661	35,098
DBSCAN	83.181	87.409	87.751	91.956	100	90,058	77.864	89,589
SD	83.181	88.504	94.516	100	100	93,238	78.028	91,325
Final	83.181	88.504	94.516	100	100	93,238	78.028	91,325

The method was able to classify correctly 93% of the outliers, with a F2-Measure of 91.3 (as seen in Table 5.8). All of extreme outliers (type 4 and 5) were caught, 94.5% of type 3 but was not capable to get over than 90% of type 1 and 2 outliers caught. As we can see in 5.3, most of outliers remaining are not distinguishable from the remaining points. Besides the type 1 and 2, we still have almost 30 type 3 outliers, visible in the figure.

5.3 Performance Comparison

In this section we will compare the performance of both the Baseline Method analyzed in Section 5.1 and the Outlier Detection method analyzed in Section 5.2. We will use as the main source of comparison the F2-Measure for each dataset.

Table 5.9: Comparison between F2-Measure

Dataset	MDAS	DBSCAN	SD	Final	Baseline
1	37,079	90,315	90,315	90,315	59,302
5	26,4	89,49	90,583	90,583	61,203
10	35,098	89,589	91,325	91,325	58,819

In Table 5.9 we can observe the F2-Measure values achieved in each step for each dataset. In the end we also can see the comparison between the Final results obtained by our method against the Baseline Method. Despite having three different scenarios (datasets with different percentage of outliers), the results were very similar in all of them, with our method having F2-Measure values near 90 against the 60 from the Baseline Method. Below we have two plots that allow us to see the evolution of both the F2-Measure (Figure 5.4) and the Recall (Figure 5.5)in the three datasets during our Method's steps, and compare it to the Baseline mean value for those metrics.

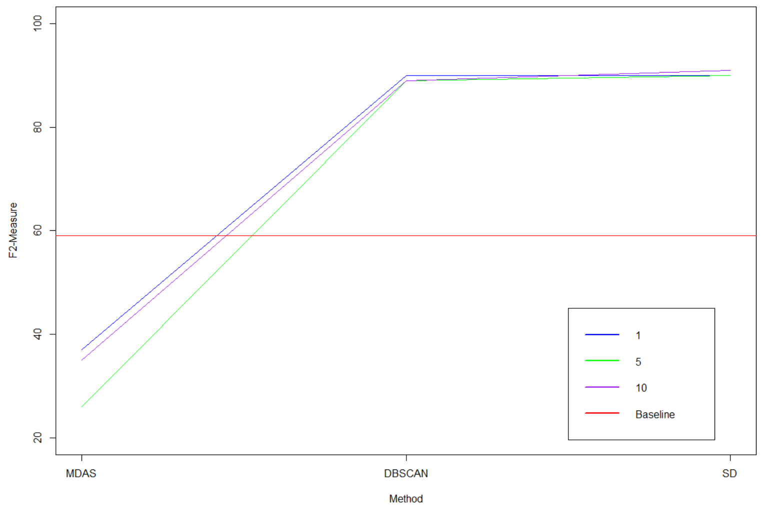

Figure 5.4: F2-Measure Evolution during the Method in the 3 datasets. The Yellow line represents the average Baseline Result

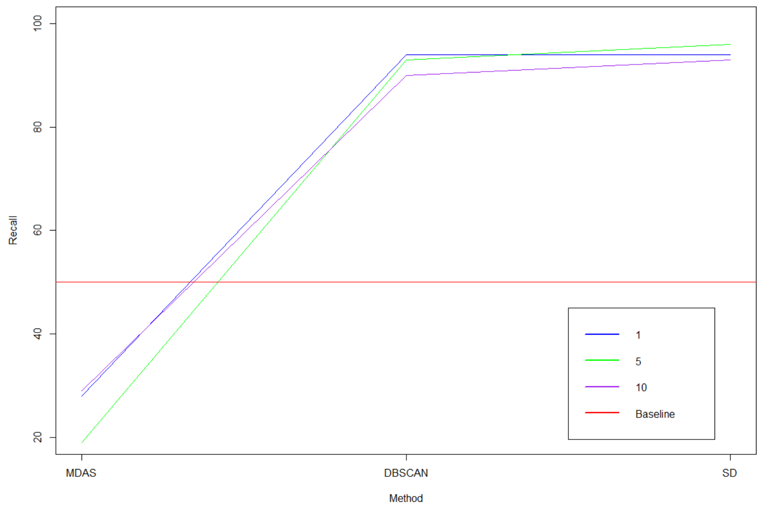

Figure 5.5: Recall Evolution during the Method in the 3 datasets. The Yellow line represents the average Baseline Result

Looking at Figures 5.4 and 5.5 and Table 5.9 we can analyze the following:

- DBSCAN is the more impactful step in the Method. We can observe that in the DBSCAN step we already have near 90 F2-Measure values, up from the 20/30/40 values from the MDAS algorithm.
- MDAS algorithm alone did not perform better than the Baseline Method in any dataset. Comparing the F2-Measures and both plots we can observe that MDAS algoritm performance was always worst than the Baseline mean values.
- The Standard Deviation does improve the overall performance of the method, although it is by a very small amount. With the exception of dataset with 1% of outliers, the SD step slightly improved the methods' performance.

Chapter 6

Conclusion

Outlier Detection and treatment is a fundamental process in the predictive and statistical analysis of sensor data. Due to Big Data chalenges it is necessary to develop automated outlier detection mechanisms to this costly and error prone objective. Paired with Big Data Analytics, Data Mining and Machine Learning introduced tools that allow this process not only to be automated and faster, but also to be more accurate when identifying outliers. In this thesis, we utilized real datasets from a dam in Portugal to understand the available tools to detect and treat outliers. Baseline techniques included Whiskers Box Plots and Standard Deviation thresholds, while Machine Learning provided clustering techniques (since we did not have labeled datasets, we used unsupervised algorithms), like K-Means, DBSCAN and LOF. Due to the quality of Manual Measurements, we were able to introduce an algorithm which makes use of these measurements. Since MDAS represents the variable behavior, they also represent the expected behavior for the ADAS measurements, so we may use them as a control to ensure ADAS measurements' quality.

We introduced a method that made use of the MDAS focused algorithm, to remove most or all of the extreme outliers, followed by a clustering algorithm (DBSCAN proved to be very useful). In the end we refined the results with a Baseline technique (Standard Deviations thresholds) that most of the times slightly improved the performance. To demonstrate and evaluate the methods, and to be able to prove that it performed better than the previous techniques, we created, with the help of an expert, three different labeled datasets with different percentages of outliers (1, 5 and 10%), and five different types of outliers (representing extreme and multi-variable outliers). Then we used the Baseline techniques and the proposed method to try to detect and remove the outliers from the datasets, gathering information about several classification metrics like Recall and F2-Measure.

6.1 Analysis of Research Questions

The main objective of this research was to be able to detect outliers in sensor data from engineering structures. We focused on sensors from dams, and with this thesis, tried to answer the following questions:

RQ1. Can we use clustering algorithms to detect outliers in sensor data?

RQ2. Is the information from other sensors useful when detecting outliers?

RQ3. Can we find a method that detects all or most of the outliers in a dataset? Does it perform better than baseline methods?

During the design and development phase (Chapter 3), we tried to use several clustering algorithms to detect outliers in dam's sensor data. From the three algorithms we used (K-Means, DBSCAN and LOF), two of them proved to be very useful and capable of identifying outliers in the several different kinds of variables from the sensors (see Section 3.4.4). In the end, we opted for using DBSCAN in the developed method (as seen in Section 4.3.3), which proved to be a extremely important step of the method, since it was capable to detect most of the outliers (we were able to obtain near 90 value for F2-Measure and Recalls above 90%, as seen in Chapter 5). Note that not all of the outliers were in place to be detected, since type 1 outliers offsets can be even smaller than the sensor error.

While most of the algorithms used did not use the information of other sensors, like Temperature, Water Level, etc, the clustering algorithms allow us to input this information, increasing the performance of the algorithms. Before applying our method, we did some Preliminary Tests (Section 4.3.1), through which we analyzed over 50000 different combinations of parameters in batch on the labeled datasets. One of the variations we tried was inputing different information on the clustering algorithm (DBSCAN), testing 12 different combinations of predictors (information extracted from other sensors (Water Level for example), or representing other sensors, as is the case of the Temperature). While finding the perfect combination of parameters was not the objective of our study (since these combinations will vary depending on the target variable, number and density of outliers, etc.), we wanted to be able to differentiate outliers from structural problems and therefore we needed the information of other variables. We proved that using environmental information allows better performance of the DBSCAN algorithm, as seen in Chapter 4 - Table 4.4. Only with the Water Level information, the mean results of the algorithm improved from a value of 0.406 to 0.442 in the F2-Measure (maxing at an average value of 0.522 utilizing Water Level and Temperature). In the end, and for these datasets, every single variation with external values obtained, in average, better results than the variation with none (just the target variable).

To be able to detect outliers in the datasets, we introduced an algorithm that made use of MDAS measurements (Section 3.5). While the algorithm does work to detect extreme outliers, it is insufficient by itself when the objective is to identify all or most of the outliers. So in Section 3.6 a method was introduced that involved the previous algorithm, followed by the clustering algorithm and refined with a Baseline Method. As demonstrated in Section 4.3, the method fulfills its purpose, identifying and removing most of the outliers. Below, in Figure 6.1, 6.2 and 6.3, we can see the final results after applying the method on the labeled datasets.

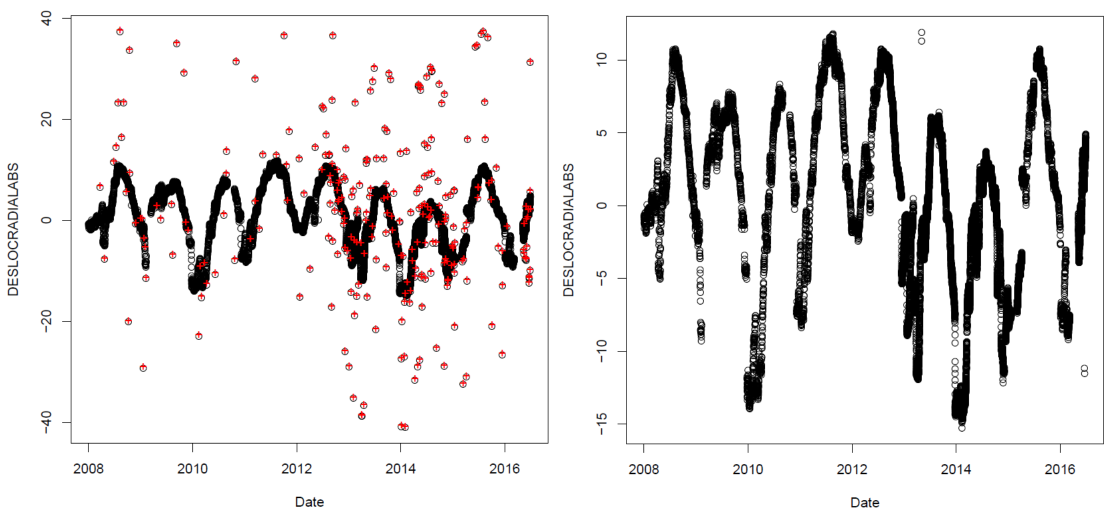

Figure 6.1: Impact of the Method on the Dataset with 1% of Outliers. Left - Original Dataset; Right - Final Result after outlier removal

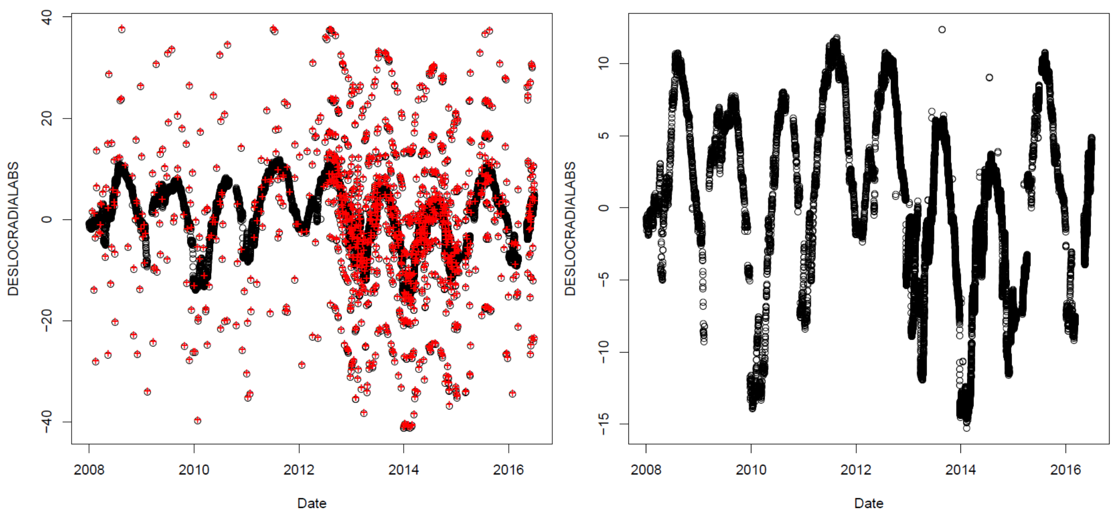

Figure 6.2: Impact of the Method on the Dataset with 5% of Outliers. Left - Original Dataset; Right - Final Result after outlier removal

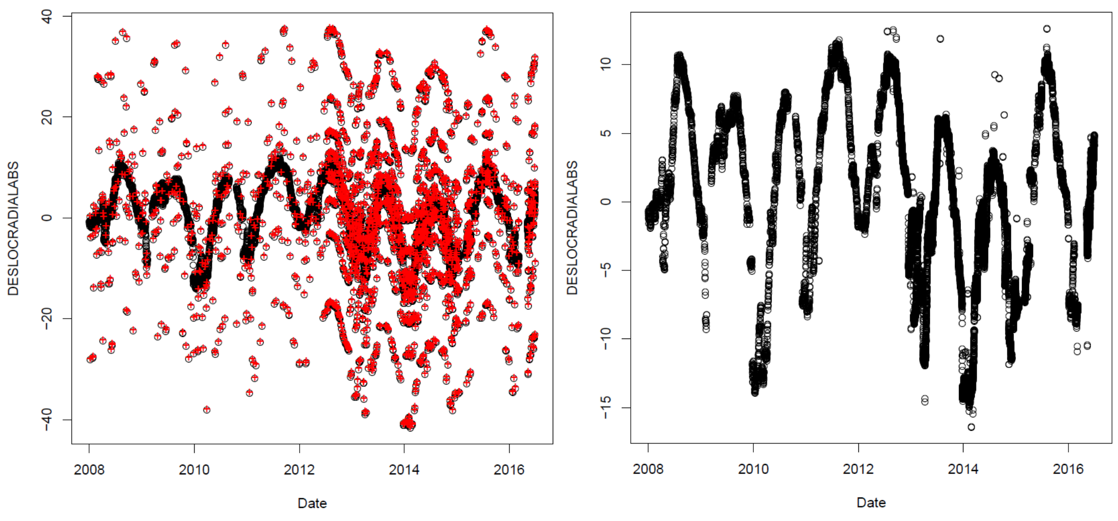

Figure 6.3: Impact of the Method on the Dataset with 10% of Outliers. Left - Original Dataset; Right - Final Result after outlier removal

In Section 5.3, we were able to compare the classification metrics from both the results showed above and the Baseline method. In Table 6.1, we can observe that the final results of our Method are better than the Baseline values in every dataset, reaching F2-Measure values above 90 (against values averaging 60 in the Baseline), and Recall values over 93 as well (against values averaging 50 in the Baseline).

Table 6.1: Comparison between F2-Measure

	Method Metrics		**Baseline Metrics**	
Dataset	F2	Recall	F2	Recall
1	90,315	94,505	59,302	49,817
5	90,583	96,564	61,203	51,316
10	91,325	93,238	58,819	48,794

We have enough information to answer our Research Questions:

RQ1. As shown in the Demonstration (Chapter 4), we can use clustering algorithms to detect outliers. Clustering algorithms, like K-Means and DBSCAN, proved to be very helpful when detecting the several types of outliers in the datasets;

RQ2. Information about other sensors proved to be very useful when detecting outliers. Water Level and Temperature (simulated) measurements were used by the clustering algorithms during our research, improving the overall performance;

RQ3. A Method for Outlier Detection and Treatment was design and developed and was able to detect between 93% to 96% of the outlier in our demonstration. In the end, this method's performance was better than the Baseline Methods, as we can see in Table 6.1.

6.2 Limitations

During this investigation we encountered some problems and limitations related to our method. While we were able to satisfy our main objectives (detect all or most of the outliers), we still have some visible outliers in the final result of all datasets. The main limitations of our method are summed up below.

As expected, Type 1 and 2 outliers are the most difficult to detect. Since they have a very small offset from the original point, they can be considered non-outliers, most of the times due to being inside the expected sensor's error threshold. We did not calibrate our method to weight more the remaining types, probably causing a lower Precision while trying to obtain a higher Recall, that is, when trying to detect the small outliers, we have a higher chance of obtaining False Positives, and therefore lowering the Precision.

Since DBSCAN is a density based algorithm, we had a problem with our datasets. As we can observe in the final results (see Figure 6.1 to 6.3), most of the remaining observable outliers appear after the year 2012. With a deep analysis, we can understand why: from 2008 until 2012 (including), so during a 5 year spawn, we have 12800 observations. But afterward, from 2013 to 2016, during 4 years, we have almost 310000 (increased acquisition frequency). Hence, we have a larger density of points after the year 2012, but we ran the algorithm with the same parameters for all the dataset.

The proposed algorithm was not in any case better that the Baseline Method, but the two are suited for extreme outlier detection. The Baseline Method however is very dependable on the predictive model. If the predictive model does not provide a good fit to the data, the Baseline Method will not perform well.

We were not able to separate an outlier from a structural problem. While we were able to detect both, we did not have any way to classify the outlier as a structural problem without looking at the neighborhood of the sensor.

Starting with the Baseline Method, one of the improvements we can make is utilizing it instead of the introduced algorithm (MDAS) in our Method, since we have seen that, in this dataset, it performed better. Another improvement will come from the utilization of a better Predictive Algorithm, or improvement of the current parameters. With a better predictive model, we will have a better fit of the dataset, and therefore a more accurate threshold.

Since DBSCAN was proven to be the main step of the presented Method, our focus should be on improving it. To do so, we can either treat the dataset as two or more separated frames or refine the detection by running DBSCAN more than once with different parameters in the same dataset. The objective of both solutions is the same: adapt the algorithm to different types of densities, and therefore, increase its general performance.

6.3 Future Work

Now that we have proved that we are able to detect and treat outliers in different datasets with the same target variables, we should expand our tests to different types of target variables. With different datasets, we need to do a deeper study on the clustering algorithm used (in this case DBSCAN), better understanding the impact of each parameter on the performance and provide better fitting depending on the dataset. This is true not only for the clustering algorithm, but also the predictive algorithm used on the Baseline Method (Standard Deviation).

As seen during the design and development phase, most of the datasets from the CaseStudy Dam have some extreme outliers. When asked about this problem, the Dam Expert explained that most of the datasets available contain a small number of outliers (either from the installation and calibration phase of the sensor, or from posterior errors). One of the things that should be done in batch mode, probably with the approval of an expert, is to either use the Baseline Method or the MDAS algorithm, allowing a first pass through the data and removing the extreme outliers. After that, we should do a more in depth study for each of the datasets and remove the remaining outliers with our Method.

During our research, we opted to remove outliers from the datasets after they were identified. However, in a real life situation information should not be deleted, but labeled instead. After an observation is signaled as an outlier, by our method, we still do not know if it is an anomaly or a structural problem. After the alert is given, a system should be in place that allows the labeling of observations, either as a normal point, anomaly or a problem. This annotation can be executed either by an automatic algorithm (extreme values should be always anomalies) or by an expert (as seen in Figure 6.4). To correctly label an observation as a structural problem, the expert should have relevant information about not only the external/environmental inputs but also the neighborhood of the sensor. Most of the structural problem should correspond to a value deviation from the variable both in the sensor and the sensors nearby. This information can be delivered to the expert through data visualization techniques and algorithms, such as graph networks.

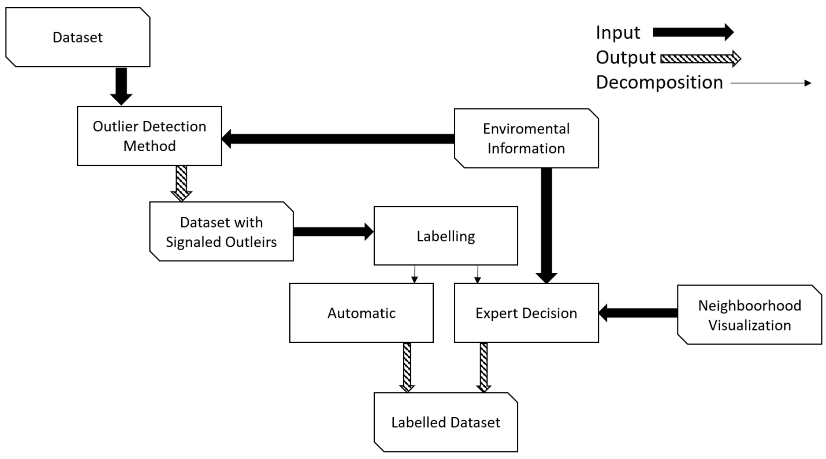

Figure 6.4: Outlier Detection and Treatment System

In this case study, the datasets were not labeled which lead to the choice of unsupervised methods (we had labeled datasets created in order to extract classification metrics to assess the method performance). If the system presented in Figure 6.4 is in place, we can use it to obtain labeled datasets, which will allow the use of supervised learning algorithms, like SVM and KNN. In the end, our main objective is to gather enough information and knowledge in order to anticipate and react to real problems in engineering structures. To do so, we should be able to have a real-time response to sensor data. Once the information is received, we want to classify it as either an outlier (error or structural problem) or a non-outlier, and depending on the classification be able to alert the experts about existent structural problems.

Bibliography

Abbott, D. (2014). *Applied predictive analytics: Principles and techniques for the professional data analyst.* John Wiley & Sons.

Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, *2*(4), 433–459.

Aggarwal, C. C. (2013). *Outlier analysis.* Springer.

Berry, M. J., & Linoff, G. (1997). *Data mining techniques: for marketing, sales, and customer support.* John Wiley & Sons, Inc.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). Lof: identifying density-based local outliers. In *Acm sigmod record* (Vol. 29, pp. 93–104).

Burton, B., Geishecker, L., Hostmann, B., Austin, T., Herschel, G., Soejarto, A., & Rayner, N. (2006). Business intelligence focus shifts from tactical to strategic. *Gartner Research, Stamford, CT*.

Castro, A. T., & Barateiro, J. (2015). Sistemas de informação no controlo de segurança de barragens de betão. *Construção Magazine*(70), 16–21.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, *41*(3), 15.

Chen, M., Mao, S., Zhang, Y., & Leung, V. C. (2014). *Big data: related technologies, challenges and future prospects.* Springer.

Cheng, L., & Zheng, D. (2013). Two online dam safety monitoring models based on the process of extracting environmental effect. *Advances in Engineering Software*, *57*, 48–56.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (Vol. 96, pp. 226–231).

Evans, D. (2011). The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, *1*(2011), 1–11.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996a). From data mining to knowledge discovery in databases. *AI magazine*, *17*(3), 37.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996b). The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, *39*(11), 27–34.

Few, S. (2006). Information dashboard design.

Friendly, M., & Denis, D. (2005). The early origins and development of the scatterplot. *Journal of the History of the Behavioral Sciences*, *41*(2), 103–130.

GlobalDataStrategy. (2017). *Transform data into valuable business insights.* `https://globaldatastrategy.com/our-services/business-intelligence-bi-big-data-analytics/`. (Accessed: 2017-09)

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques.* Elsevier.

Hartigan, J. A., & Hartigan, J. (1975). *Clustering algorithms* (Vol. 209). Wiley New York.

Hawkins, D. M. (1980). *Identification of outliers* (Vol. 11). Springer.

Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial intelligence review*, *22*(2), 85–126.

IBM. (2015). *Extracting business value from the 4 v's of big data.* `http://www.ibmbigdatahub.com/infographic/extracting-business-value-4-vs-big-data/`. (Accessed: 2017-09)

Jain, N., & Srivastava, V. (2013). Data mining techniques: a survey paper. *IJRET: International Journal of Research in Engineering and Technology*, *2*(11), 2319–1163.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning, volume 103 of.* Springer Texts in Statistics.

Keim, D. A. (2002). Information visualization and visual data mining. *IEEE transactions on Visualization and Computer Graphics*, *8*(1), 1–8.

Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling.* John Wiley & Sons.

Kohavi, R., et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai* (Vol. 14, pp. 1137–1145).

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of massive datasets.* Cambridge University Press.

Li, N., Li, P., Shi, X., Yan, K., & Ren, W. (2010). Outlier identify based on bp neural network in dam safety monitoring. In *Informatics in control, automation and robotics (car), 2010 2nd international asia conference on* (Vol. 2, pp. 210–214).

Ljunggren, M., Logan, T., & Campbell, P. (2013). Is your dam as safe as your data suggest. In *Proceedings of the nzsold/ancold conference* (Vol. 1).

Marr, B. (2015). *Why only one of the 5 vs of big data really matters.* `http://www.ibmbigdatahub.com/blog/why-only-one-5-vs-big-data-really-matters/`. (Accessed: 2017-09)

Mata, J. (2011). Interpretation of concrete dam behaviour with artificial neural network and multiple linear regression models. *Engineering Structures*, *33*(3), 903–910.

Mata, J., & Tavares de Castro, A. (2015). Assessment of stored automated measurements in concrete dams.

McAfee, A., Brynjolfsson, E., et al. (2012). Big data: the management revolution. *Harvard business review*, *90*(10), 60–68.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). *Introduction to linear regression analysis* (Vol. 821). John Wiley & Sons.

Montgomery, D. C., & Runger, G. C. (2010). *Applied statistics and probability for engineers.* John Wiley & Sons.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, *24*(3), 45–77.

Portela, E., Pina dos Santos, C., Silva, A., Galhardas, H., & Barateiro, J. (2005). A modernizaçao dos sistemas de informaçao de barragens: o sistema gest-barragens.

Ray, S. (2016). *A comprehensive guide to data exploration.* `https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/`. (Accessed: 2016-12)

Analytics IBM. (2017). *Big data analytics.* `https://www.ibm.com/analytics/us/en/big-data/`. (Accessed: 2017-09)

Insights SAS. (2017). *Big data. what it is and why it matters.* `https://www.sas.com/en_us/insights/big-data/what-is-big-data.html`. (Accessed: 2017-09)

Salazar, F., Morán, R., Toledo, M. Á., & Oñate, E. (2017). Data-based models for the prediction of dam behaviour: a review and some methodological considerations. *Archives of Computational Methods in Engineering*, *24*(1), 1–21.

Shearer, C. (2000). The crisp-dm model: the new blueprint for data mining. *Journal of data warehousing*, *5*(4), 13–22.

Singh, K., & Upadhyaya, S. (2012). Outlier detection: applications and techniques. *International Journal of Computer Science Issues*, *9*(1), 307–323.

Sun, Z., Zou, H., & Strang, K. (2015). Big data analytics as a service for business intelligence. In *Conference on e-business, e-services and e-society* (pp. 200–211).

Tayfur, G., Swiatek, D., Wita, A., & Singh, V. P. (2005). Case study: Finite element method and artificial neural network models for flow through jeziorsko earthfill dam in poland. *Journal of Hydraulic Engineering*, *131*(6), 431–440.

Wirth, R., & Hipp, J. (2000). Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining* (pp. 29–39).

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, *2*(1-3), 37–52.

Yao, Y., Sharma, A., Golubchik, L., & Govindan, R. (2010). Online anomaly detection for sensor systems: A simple and efficient approach. *Performance Evaluation*, *67*(11), 1059–1075.

Yu, H., Wu, Z., Bao, T., & Zhang, L. (2010). Multivariate analysis in dam monitoring data with pca. *Science China Technological Sciences*, *53*(4), 1088–1097.

Zhang, S., Zhang, C., & Yang, Q. (2003). Data preparation for data mining. *Applied Artificial Intelligence*, *17*(5-6), 375–381.