IMPROVING QUALITY CONTROL EFFICIENCY WITH AN UNSUPERVISED ANOMAL DETECTION APPLICATION USING LIGHT SENSORS

TÍTOL DEL TFG: Improving quality control efficiency with an unsupervised anomaly detection application using light sensors

TITULACIÓ: Grau en Enginyeria Telemàtica

AUTOR: Sergio Gras López

DIRECTOR: Roc Meseguer Pallares

DATA: 13 de febrer del 2023

Title: Improving quality control efficiency with an unsupervised anomaly detection application using light sensors

Author: Sergio Gras López

Director: Roc Meseguer Pallares

Date: February 13, 2023

Overview

This project aims to investigate and create a solution to a problem that the company Giesecke & Devrient has in one of the quality control processes. The base of one of its products is made up of a metallic foil of a certain color that has to meet quality standards, but producing it requires a complex process where errors appear.

The project aims to use data collected from 11 sensors that measure the foil wavelength to identify these errors. However, there is a challenge as there are no established values for what is considered acceptable, and the sensors used do not work in optimal conditions, which generates measurement errors in the data set.

To tackle this challenge, extensive data analysis was performed to understand the system. Different techniques and principles have also been applied for the detection of anomalies, analyzing and comparing the results. Ultimately, a prototype tool was developed to integrate these techniques and improve the quality control process.

Título: Mejora de la eficiencia del control de calidad con una aplicación de detección de anomalías no supervisada mediante sensores de luz

Autor: Sergio Gras López

Director: Roc Meseguer Pallares

Fecha: 13 de febrero del 2023

Resumen

Este proyecto tiene como objetivo investigar y crear una solución a un problema que tiene la empresa Giesecke & Devrient en uno de los procesos de control de calidad. La base de uno de sus productos está formada por una lámina metálica de un determinado color que tiene que cumplir unos estándares de calidad, pero su elaboración requiere un proceso complejo en el que aparecen errores.

El proyecto tiene como objetivo utilizar los datos recopilados de 11 sensores que miden la longitud de onda de la lámina para identificar estos errores. Sin embargo, existe un desafío ya que no existen valores establecidos para lo que se considera aceptable y los sensores utilizados no funcionan en condiciones óptimas, lo que genera errores de medición en el conjunto de datos.

Para hacer frente a este desafío, se realizó un extenso análisis de datos para comprender el sistema. También se han aplicado diferentes técnicas y principios para la detección de anomalías, analizando y comparando los resultados. Finalmente, se desarrolló una herramienta prototipo para integrar estas técnicas y mejorar el proceso de control de calidad.

ACKNOWLEDGMENT

I am deeply grateful to the company Giesecke & Devrient for offering me the chance to tackle one of their challenges and provide a solution. A special mention goes to Marta Couto, who has provided me with the resources and time that I have needed to be able to do this work.

My heartfelt thanks also go to my mentor, Roc Meseguer, who has been a constant source of guidance and support. Their invaluable advice and commitment to tutoring my project, even during holidays, was essential to my success.

Finally, I would like to express my gratitude to my girlfriend, friends, and family. Although they were not directly involved in this project, their unwavering support and motivation were instrumental in my journey. I could not have completed this work without their love and encouragement.

INDEX

CHAPTER 1. INTRODUCTION 1

1.1. Giesecke & devrient 1

1.2. Problem 2

1.3. Goals 2

CHAPTER 2. DATA ANALYSIS 3

2.1. The data 3
- 2.1.1. Data source 3
- 2.1.2. Credibility and integrity 5

2.2. Prepare and process 5
- 2.2.1. Data cleaning and manipulation 6
- 2.2.2. Light spectrum 8

2.3. Analysis 10
- 2.3.1. Data visualization 10
- 2.3.2. Descriptive statistic 11
- 2.3.3. Distribution 13
- 2.3.4. Correlation 15

CHAPTER 3. ANOMALY LITERATURE 17

3.1. Types of Anomalies 17
- 3.1.1. Point Anomalies 17
- 3.1.2. Contextual anomalies 18
- 3.1.3. Collective anomalies 19

3.2. Data labels 19
- 3.2.1. Supervised anomaly detection 19
- 3.2.2. Semi-Supervised anomaly detection 20
- 3.2.3. Unsupervised anomaly detection 20

3.3. Output of Anomaly Detection 20
- 3.3.1. Scores 20
- 3.3.2. Labels 20

CHAPTER 4. APPLYING ANOMALY DETECTION TECHNIQUES 21

4.1. Statistical techniques 21
- 4.1.1. The 3σ-Rule 22
- 4.1.2. Interquartile Range Rule 26
- 4.1.3. Gaussian mixture model 30
- 4.1.4. Conclusions from the statistical techniques 37

4.2. Nearest neighbor techniques 38
- 4.2.1. LOF 38
- 4.2.2. Conclusions from the nearest neighbor techniques 42

4.3. Linear models techniques 43

INDEX

4.3.1. Autoencoders 43
4.3.2. Vector Autoregression 48
4.3.3. Conclusions from the linear model techniques 55

CHAPTER 5. WEB APP DEVELOPMENT 56

5.1. Motivations and requirements 56

5.2. Technologies 57
5.2.1. Database 57
5.2.2. Backend and frontend 57

5.3. Design 58
5.3.1. Wireframe and user experience 58
5.3.2. Mockup 59

5.4. Data diagram 60

5.5. Development 61
5.5.1. Development steps of a Django web application 61
5.5.2. Architecture 62

5.6. Result 63

5.7. Testing 64

CHAPTER 6. CONCLUSIONS 66

6.1. Future lines 67

6.2. Personal assessment 68

BIBLIOGRAPHY 69

CHAPTER 1. INTRODUCTION

G&D is a company that offers services in diverse industries such as connectivity and payments. In one of their production processes, they are encountering a challenge in their quality control system, where it is not accurately detecting instances of color shrinkage in the metallic foil they manufacture. To address this issue, we are embarking on a project to develop a solution that will detect process anomalies. Our goal is to help G&D improve the efficiency and reliability of its quality control process by providing a robust and scalable solution that can accurately identify anomalies. The project will involve the research and implementation of some leading techniques in the field of anomaly detection, as well as the development of a custom tool to support the detection process.

In this chapter, we will gain a deeper understanding of the company and the issue they are facing, and outline the objectives that this academic project is aimed at achieving.

1.1. Giesecke & devrient

To analyze the problem first we have to understand who G&D is and what products it offers. The G&D company is a German company based in Munich that offers security technologies, both in the physical and digital world.
Founded in 1852 by Hermann Giesecke and Alphonse Devrient, the firm initially specialized in high-quality printing, notably currency and securities printing. The firm has maintained this important role until today, being the world's second largest supplier of banknotes.

It currently focuses on 4 fields:

- Payment: Provides end-to-end solutions, products, technologies and services for the entire cash, card and digital payment cycle.
- Connectivity: supports mobile network operators and offers solutions such as eSIM. It also ensures the management of IoT device and provides secure services to the automotive industry.
- Identities: comprehensive solutions and services for legal identity. These range from paper to security printing, electrical chip components, and mobile identification solutions.
- Digital infrastructures: facilitates digital infrastructures for governments, companies and society with cybersecurity solutions in sectors such as health, industry, border control and defense.

1.2. Problem

The production of a product related to physical payment involves a colored foil with specific characteristics. The goal is to produce a large foil that meets quality standards, but the process can result in errors, some of which are on the millimeter scale, and others are on the metric scale.

It is imperative to detect any errors that occur during the production process in order to discard any affected sections of the foil and ensure that only high-quality products are utilized.

The process to generate this foil is complex, made up of more than 6 sub-modules that work together to generate this foil. The problem we are facing occurs in one of these submodules. In this particular sub-module, the colored foil passes through a vacuum chamber, and it is inside this chamber that a set of light sensors are taking measurements of the wavelength reflected from this foil.

These sensors, being located inside a vacuum chamber, can cause errors in the reading of the wavelength. This, in turn, hinders other types of readings such as barcode reading. This problem translates into a difficult mapping between other data collected by other modules and those collected by these sensors. That is why for this problem only the values obtained by these sensors are available. Additionally, we know that in a similar process up to 30% of the product can be discarded depending on the quality requirements.

Previously, the evaluation of the foil quality was conducted manually by an operator who would inspect its uniformity during a specific step in the production process. This method is inadequate for detecting shrinkage due to the nature of the errors, making it difficult and inefficient. In an effort to solve this challenge, G&D aims to utilize the data captured by 11 light sensors located inside the vacuum chamber in the production process. The goal is to analyze this data and develop a tool that can assist the operator in detecting foil shrinkage, thus improving the quality control process.

1.3. Goals

The primary goal of this final degree project is to develop a practical solution to a specific business problem by achieving the following three objectives:

- Acquire expertise in various data analysis techniques by applying them to a real-world dataset.
- Conduct a comprehensive literature review on the topics of anomalies and outliers, and identify key concepts that are relevant to the project's problem.
- Use the knowledge and skills gained throughout my academic studies to design and develop a prototype web application from scratch.

CHAPTER 2. DATA ANALYSIS

Once the problem is defined, the next step is to perform a thorough analysis of the data using essential data analysis techniques. Some of these steps are data collection, data cleaning, data preparation, and an Exploratory Data Analysis (EDA).

2.1. The data

The data used for this project comprises a collection of light sensor measurements taken over a period of time. These measurements consist of the average of the wavelengths that are captured by each sensor, with a precision that can reach several hundred nanometers. That is why we can classify this type of data collection as a time series of real values.

2.1.1. Data source

The data used in this project is derived from measurements made by a total of 11 light sensors, which are positioned along the 2-meter width of a foil. These sensors are placed inside a vacuum chamber, which presents one of the main challenges of this problem, as these sensors were not specifically designed to operate in such conditions. This leads to a considerable amount of noise in the data, which can potentially impact the accuracy of the analysis. Another significant challenge is that there is no established correlation or association between the shrinkage of the foil and the measurements made by the sensors. In other words, there is no reference sample that can be used to determine which wavelengths are considered acceptable or which deviate from the quality standards. This limitation makes it difficult to use certain data analysis methods, such as classifiers based on neural networks, which require labeled data.

The following image illustrates an overview of the data collection process. The process involves a foil that moves at a speed of 1.3 meters per second and is 2 meters wide, where 11 sensors are strategically placed across the width of the foil to measure the wavelength at specified intervals. The image also shows some examples of errors that are expected to occur in the foil, which manifest mainly as deviations in color over time and/or space.

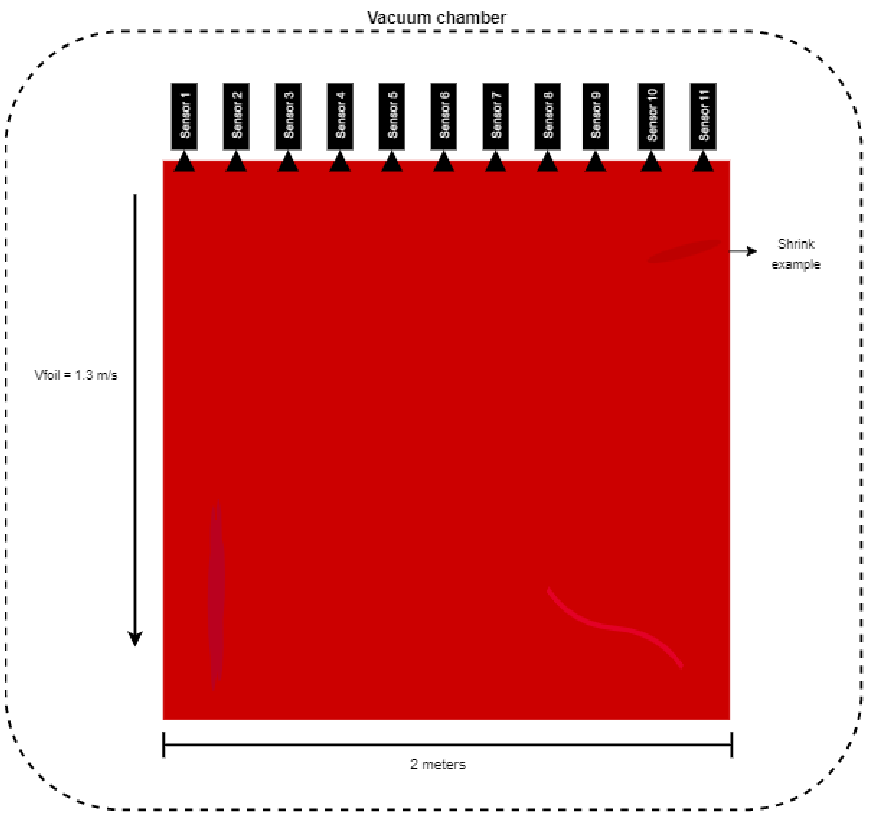


Fig.2.1 Diagram of the problem, where 11 sensors appear along a 2-meter-wide foil advancing at 1.3 meters per second inside a vacuum chamber.

Also, sensor data is characterized by varying sampling intervals between successive time points ($\Delta t = t(i) - t(i-1) \neq C$). Time series with varying sampling frequencies are called unevenly spaced time series.

The format with which the data comes is CSV, divided into more than 500 files. Each of these files contains more than 5000 samples.

Once we have seen where and how the data comes from, we will see the credibility and integrity of this data, a necessary point if we want to be realistic with the possibilities that this collection of data can offer us.

2.1.2. Credibility and integrity

To determine the credibility and integrity of the data, Google [1] designed a system composed of 5 parameters, called ROCCC by its acronym (Reliable, Originality, Comprehensive, Current, Cited). Once each of these parameters has been analyzed, we can have a clearer idea about the type of data with which we are working.

- Reliable. The data is not reliable. Data is not stored correctly due to measurement noise and variable sampling intervals. Also, the data is not labeled and does not have a reference to what the correct data looks like, so it is not complete. Despite having a large amount of data to analyze, there is no additional information that could help the study, such as error rates.
- Originality. The data is original, since it is provided directly by the company and nobody and nothing has processed or manipulated the data.
- Comprehensive. The entire data set is not comprehensive, since there are some fields like the barcode that are not read correctly. The only relevant information is the measurements from the sensors and the time taken, but even this data is neither precise nor accurate.
- Current. These data are current. The data is from 2020 but since the process described is not used at the time of this thesis, they are the most recent data on the problem.
- Cited. The data is cited. Each piece of data is labeled with the date it was taken and a unique number to identify it. The distributor is the company itself, which makes it a credible fount.

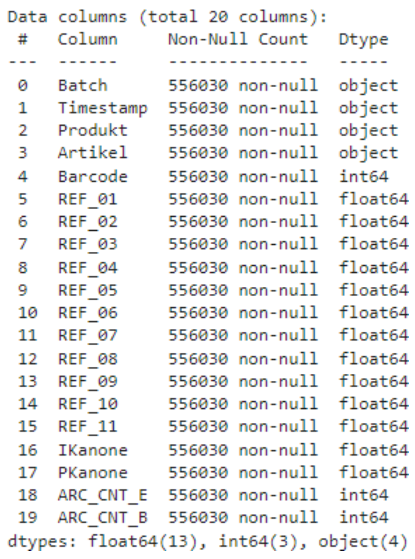
2.2. Prepare and process

Cleaning and preparing the data before performing data exploration is crucial to gaining accurate and meaningful insights from the analysis. The data cleaning process involves identifying and correcting errors, missing values, and any other inconsistencies in the data. This step is essential to ensure that the data is accurate and reliable, and that the conclusions drawn from the analysis are not biased or distorted by errors in the data.

Also, in this specific case where the data is generated in a vacuum chamber, it is important to eliminate noise generated by the sensor and chamber conditions, as this can have a large impact on the accuracy of the data and the results.

2.2.1. Data cleaning and manipulation

The following image shows all the fields that make up the data set



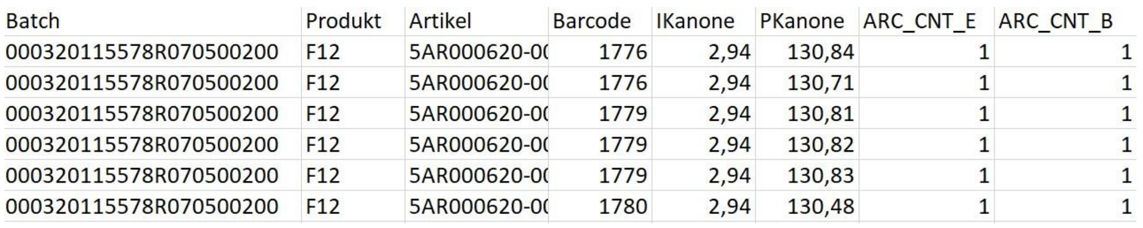
```
Data columns (total 20 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Batch      556030 non-null  object
 1   Timestamp  556030 non-null  object
 2   Produkt    556030 non-null  object
 3   Artikel    556030 non-null  object
 4   Barcode    556030 non-null  int64
 5   REF_01     556030 non-null  float64
 6   REF_02     556030 non-null  float64
 7   REF_03     556030 non-null  float64
 8   REF_04     556030 non-null  float64
 9   REF_05     556030 non-null  float64
 10  REF_06     556030 non-null  float64
 11  REF_07     556030 non-null  float64
 12  REF_08     556030 non-null  float64
 13  REF_09     556030 non-null  float64
 14  REF_10     556030 non-null  float64
 15  REF_11     556030 non-null  float64
 16  IKanone    556030 non-null  float64
 17  PKanone    556030 non-null  float64
 18  ARC_CNT_E  556030 non-null  int64
 19  ARC_CNT_B  556030 non-null  int64
dtypes: float64(13), int64(3), object(4)
```

Fig.2.2 Columns of the original data

As can be seen in the image, the data is made up of 20 fields, but not all of them are useful to us. The following fields that list are constant values or text that does not provide any information:

- Batch
- Produkt
- Artikel
- IKanone
- Pkanone
- ARC_CNT_E
- ARC_CNT_B

In the following image I show an example of the values of these fields are not useful to us.



Batch	Produkt	Artikel	Barcode	IKanone	PKanone	ARC_CNT_E	ARC_CNT_B
000320115578R070500200	F12	5AR000620-0(	1776	2,94	130,84	1	1
000320115578R070500200	F12	5AR000620-0(	1776	2,94	130,71	1	1
000320115578R070500200	F12	5AR000620-0(	1779	2,94	130,81	1	1
000320115578R070500200	F12	5AR000620-0(	1779	2,94	130,82	1	1
000320115578R070500200	F12	5AR000620-0(	1779	2,94	130,83	1	1
000320115578R070500200	F12	5AR000620-0(	1780	2,94	130,48	1	1

Fig.2.3 Example of useless field values

As I mentioned before in the comprehensibility study of section 2.1.2 , the barcode is not read correctly, probably for the same reason that causes the errors in the sensor measurements. The purpose of this field is to identify a section of the sheet line, so it is highly correlated with time. For this reason, this field will not be used in the study either.

With all this we will only be left with the fields of each sensor, labeled as ‘REF_XX’, and the moment in which each measurement was taken, called Timestamp

The following manipulation has been to correctly format the Timestamp field for its subsequent analysis with python libraries such as Pandas [2].

The next step was to obtain a relevant number of samples for the study. For this I combined more than 250 consecutive files to achieve a total of more than 1 million samples. This number of samples is equivalent to several observation periods spread over the year. In order to visualize it, all the samples of sensor 1 have been set to 0 and a scatter graph has been created where the index of each sample is the time it was taken. In this way we can see that the first sample was taken in January 2020 and the last sample at the end of the year. However, the sensors were not taking measurements uninterruptedly throughout the year, but rather there were several periods.

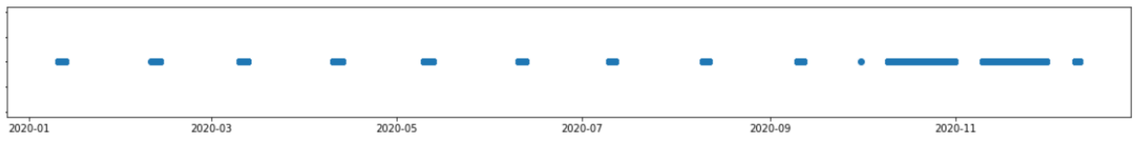
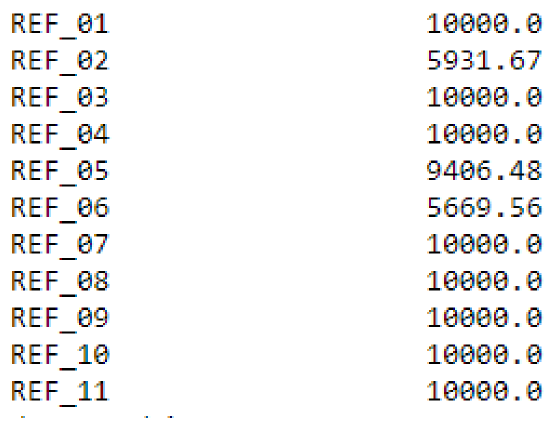


Fig.2.4 Distribution of the data set over time, using a scatter plot of sensor 1, where all values were equalized for improved viewing.

However, dealing with such large CSV files is complicated and time-consuming. That is why I made use of the Pickle[3] module, where a Python object is serialized to a byte string and save it to a file. In this way we achieve greater efficiency every time we have to consult the data.

2.2.2. Light spectrum

Before going into depth in the analysis of the data we have to try to do the analysis using quality information. As I mentioned before, the sensors are working in non-optimal conditions and probably due to this, very rare and extreme values appear.

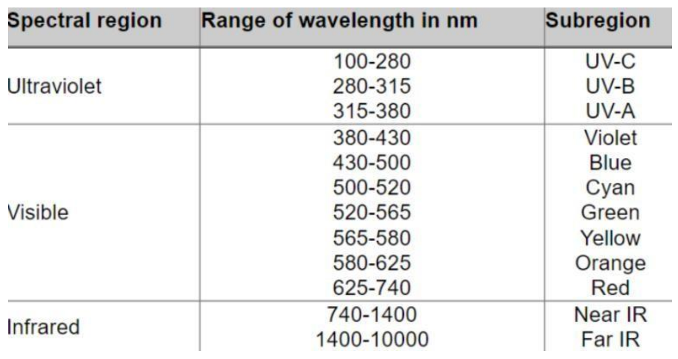


```
REF_01          10000.0
REF_02          5931.67
REF_03          10000.0
REF_04          10000.0
REF_05          9406.48
REF_06          5669.56
REF_07          10000.0
REF_08          10000.0
REF_09          10000.0
REF_10          10000.0
REF_11          10000.0
```

Fig.2.5 Maximum values of each sensor

As we see in the image above, 8 out of 11 sensors have suspiciously the same and very large value, 10000, but also the rest have extreme maximum values.

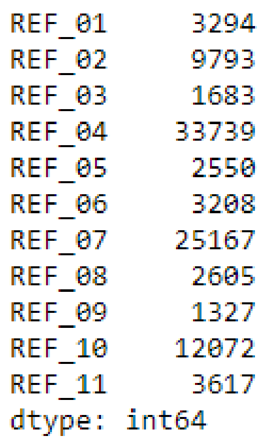
Here a hypothesis is needed to discard these values in order to use accurate and useful information for the future analysis. For them I have first studied how many samples are outside the range of the light spectrum, particularly below 380nm and above 740nm.



Spectral region	Range of wavelength in nm	Subregion
Ultraviolet	100-280	UV-C
	280-315	UV-B
	315-380	UV-A
Visible	380-430	Violet
	430-500	Blue
	500-520	Cyan
	520-565	Green
	565-580	Yellow
	580-625	Orange
	625-740	Red
Infrared	740-1400	Near IR
	1400-10000	Far IR

Fig.2.6 Range of wavelength in each spectral region

The results indicated that of 80041 samples at least 1 of its sensors measured a value outside the visible light spectrum, this represents 7.14% of the total data set. If we search specifically for each sensor, we obtain the following values:



```
REF_01     3294
REF_02     9793
REF_03     1683
REF_04    33739
REF_05     2550
REF_06     3208
REF_07    25167
REF_08     2605
REF_09     1327
REF_10    12072
REF_11     3617
dtype: int64
```

Fig.2.7 Number of samples outside the light spectrum for each sensor

Approximately 10% of the data set consists of values that are outside the visible light spectrum. Removing or replacing these values may be an option as the primary objective is to measure foil color for quality control purposes, and values outside the visible spectrum may indicate errors in reading. However, this approach is considered drastic as it discards a significant amount of potential data. Additionally, sensors used for this purpose typically have a wider detection range, with some sensors capable of detecting between 350 and 800 nanometers, including a small portion of the ultraviolet and infrared spectrum.

Fig.2.8 Example of the sensors used in our scenario, called spectrometers.
If we analyze the portion of samples that contain some measurement outside the working wavelength range of these sensors, we obtain 1198 samples, which represent 0.1%. We see that they are distributed as follows:

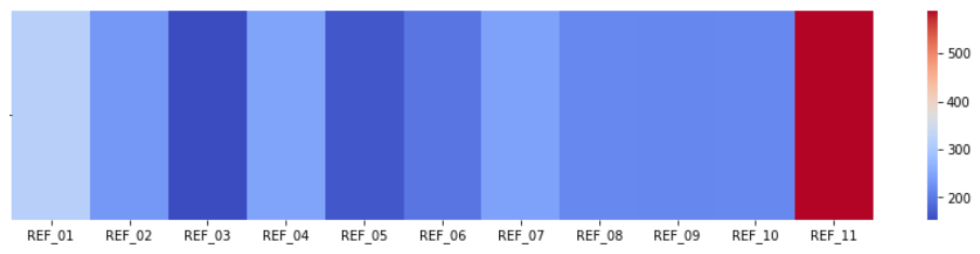


Fig.2.9 Heat map of the samples outside the working wavelength range of each sensor

In the previous image I show a heat map that goes from blue to red according to the number of samples that are outside the working range in each sensor. We can clearly notice that the sensor 11 differs from the rest of the sensors. Likewise, at the other end of the sheet, sensor 1, is the second with the highest value of samples outside the working range, perhaps indicating that the ends of the foil are prone to measurement errors. Additionally, we can see a homogeneous area between sensor 7 and 11.

As previously stated in section 2.1, the data set covers multiple time periods over the course of a year. Upon examination, it was determined that the data that falls outside the sensors' operating range is evenly distributed across these time periods and not concentrated in any specific time frame. Additionally, it was found that the majority of the samples affected by measurement errors were from only 1 or 2 sensors. Removing this subset of data would result in the loss of some information. Therefore, it has been decided to replace the value of each sensor by its average, thus for each of the sensors of this total of 1198 samples that meet the condition.

At this point we already have a significantly large data set ready to be analyzed.

2.3. Analysis

Data analysis is a crucial step in understanding the underlying patterns and relationships in a given dataset. This section will provide a comprehensive overview of the data, and a useful starting point for comprehending data is to visualize it

2.3.1. Data visualization

The following images help to better understand this type of data.

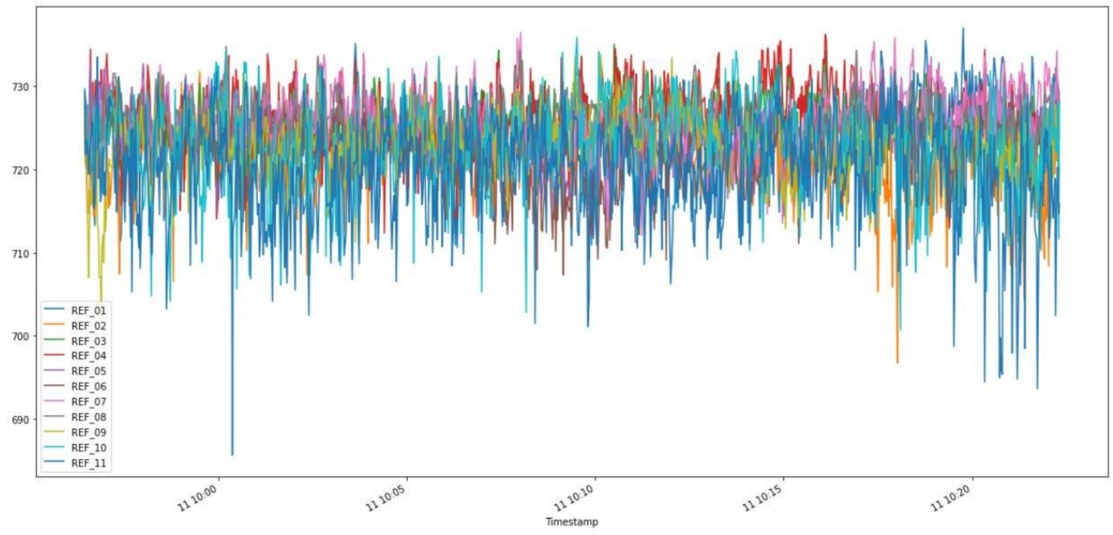


Fig.2.10 Data representation over 30 minutes of time

The image above illustrates the wavelength values in nanometer scale, captured by all 11 sensors over a period of 30 minutes. The values are presented in a continuous representation, meaning that they are plotted as a function of time.

We can notice that the values move in a considerable range of values, taking into account that the system is designed to produce a foil of uniform color, so in an ideal scenario we would expect constant values.

The image below is the same representation but in a smaller window of time, 3 minutes in this case. In it we can see that despite the fact that the 11 values are measuring the same foil, each of the sensors has a different behavior. Another point to note is that between consecutive measurements there are very abrupt jumps.

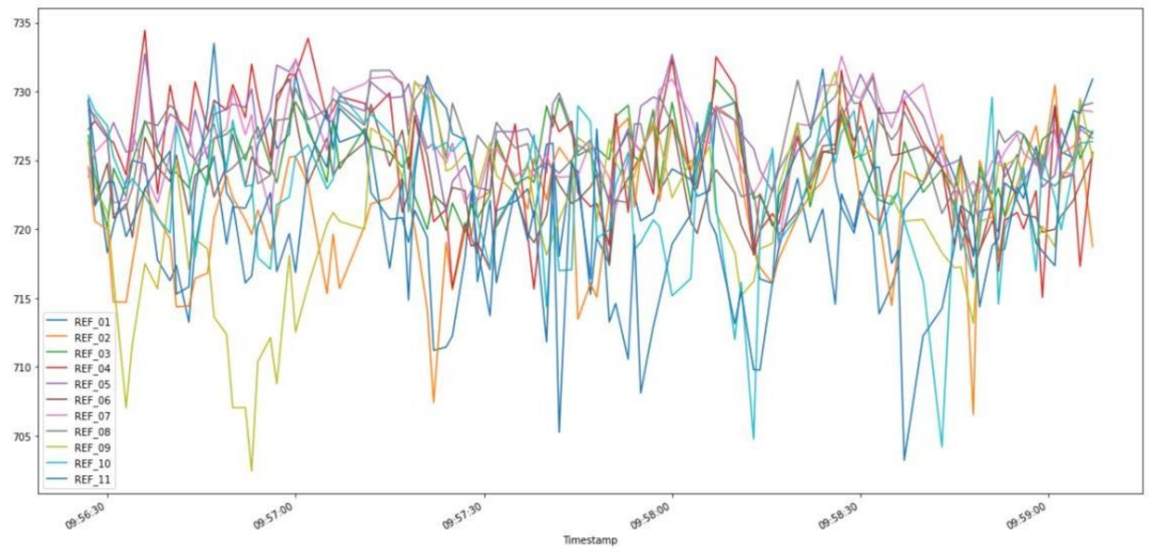
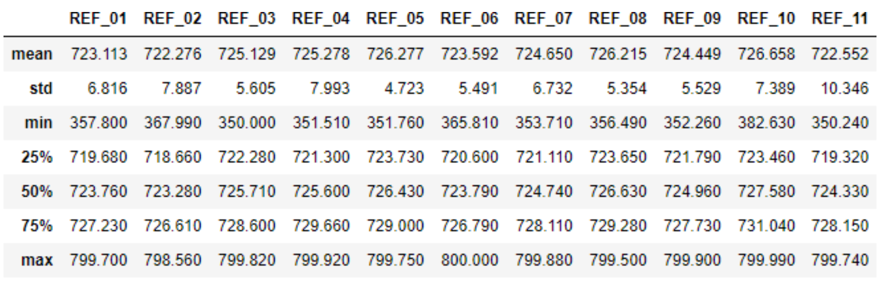


Fig.2.11 Data representation over 3 minutes of time

Descriptive statistics are an important aspect of the exploratory data analysis step because they provide a quick and easy way to understand the general characteristics and properties of a dataset. Descriptive statistics provide a summary of the distribution of the data, such the mean and standard deviation, which give an idea of the central tendency and spread of the data.

2.3.2. Descriptive statistic

This table also provides maximum values and quartiles of a total number of 1120772 samples:



	REF_01	REF_02	REF_03	REF_04	REF_05	REF_06	REF_07	REF_08	REF_09	REF_10	REF_11
mean	723.113	722.276	725.129	725.278	726.277	723.592	724.650	726.215	724.449	726.658	722.552
std	6.816	7.887	5.605	7.993	4.723	5.491	6.732	5.354	5.529	7.389	10.346
min	357.800	367.990	350.000	351.510	351.760	365.810	353.710	356.490	352.260	382.630	350.240
25%	719.680	718.660	722.280	721.300	723.730	720.600	721.110	723.650	721.790	723.460	719.320
50%	723.760	723.280	725.710	725.600	726.430	723.790	724.740	726.630	724.960	727.580	724.330
75%	727.230	726.610	728.600	729.660	729.000	726.790	728.110	729.280	727.730	731.040	728.150
max	799.700	798.560	799.820	799.920	799.750	800.000	799.880	799.500	799.900	799.990	799.740

Fig.2.12 Table of the descriptive statistics of each sensor

Looking at this table we can see that the minimum and maximum values are quite close between each sensor, this is because we have filtered the data to be between 350 and 800 nanometers.

If we look at the average of each sensor, we see that the values are between 722 and 727 nanometers.

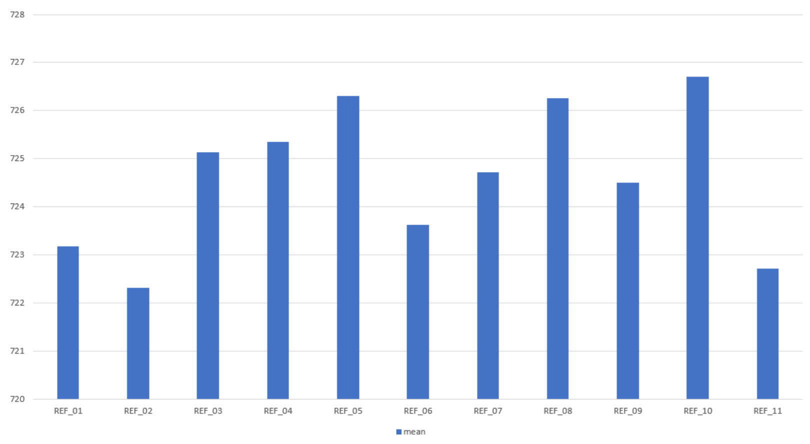
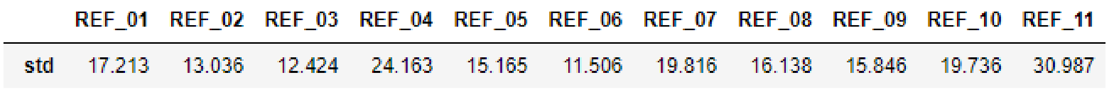


Fig.2.13 Bar graph of the average of each sensor

Based on the distribution of the means, no clear pattern can be identified. However, there seems to be a correlation between the averages and the results obtained when analyzing samples that fall outside the working wavelength range of the sensors, as seen in Figure 2.5. In this figure it can be seen that the sensors at the extremes have a large value, and in this case, these same sensors have a lower mean.

As I have commented, the descriptive statistics table above has been taken after handling wavelength working range anomalies. The following table shows the average of each sensor before this step:



	REF_01	REF_02	REF_03	REF_04	REF_05	REF_06	REF_07	REF_08	REF_09	REF_10	REF_11
std	17.213	13.036	12.424	24.163	15.165	11.506	19.816	16.138	15.846	19.736	30.987

Fig.2.14 Table of the descriptive statistics of each sensor

As we can see from the table above, even though only 0.1% of the data set was affected, the effect of replacing them has a large impact on the sensor mean. In some cases, reducing it 3 times.

On the other hand, if we analyze the standard deviation, we can see that values are between 5 and 10 nanometers. With the following graph it can be seen how, due to the fact that sensor 11 concentrates the largest number of samples outside the operating range of the sensors, it obtains the highest standard deviation.

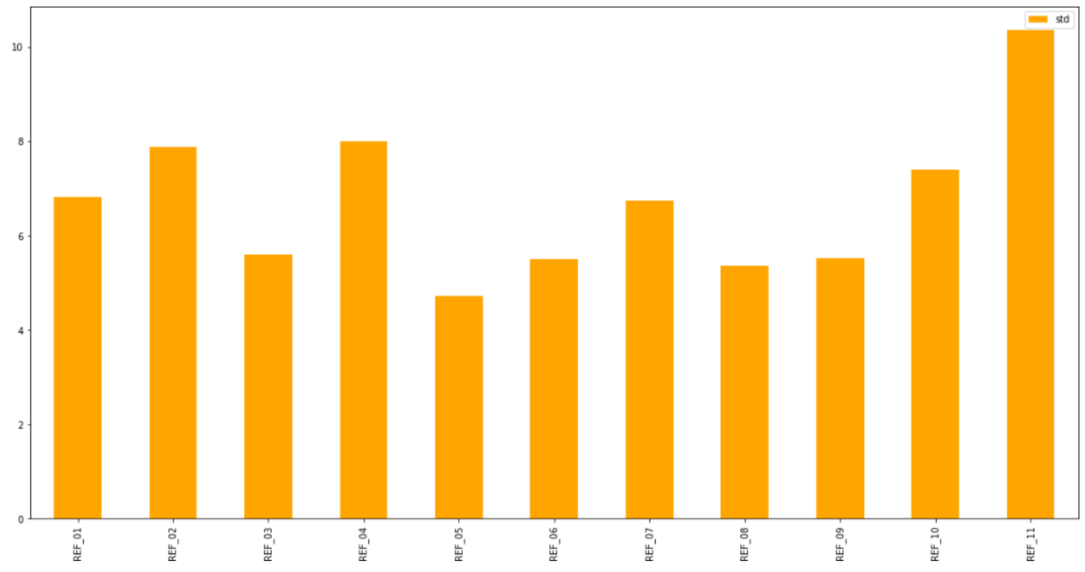


Fig.2.15 Bar graph of the standard deviation of each sensor

2.3.3. Distribution

One of the most important techniques in any data study is the distribution of observations. This can provide us with valuable information about the process that generates this data, as well as see trends and behaviors.

For the following graphs where the histograms of each sensor are shown, the same magnitude has been used on the Y axis, so all the histograms are to scale. For a better representation, the X axis has been limited between 600 and 800 nanometers. Although there is no correct number of bins to represent a histogram, it has been decided to use 900 since I wanted to group each of the 450 nanometers that we have (350-800) into two groups. It is also close to the square root of the total number of samples ($\sqrt{1120772} = 1058.66$), a rule generally used to choose the size of bins. I've also tried different numbers of bins and while they all reveal different information, the selected number seems to pick up the information for most of these combinations.

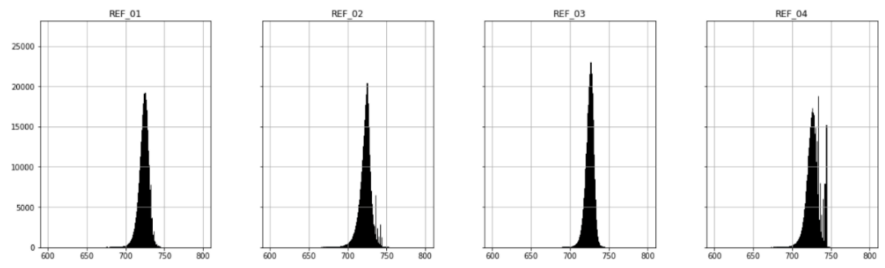


Fig.2.16 Data distribution from sensors 1, 2 ,3 and 4

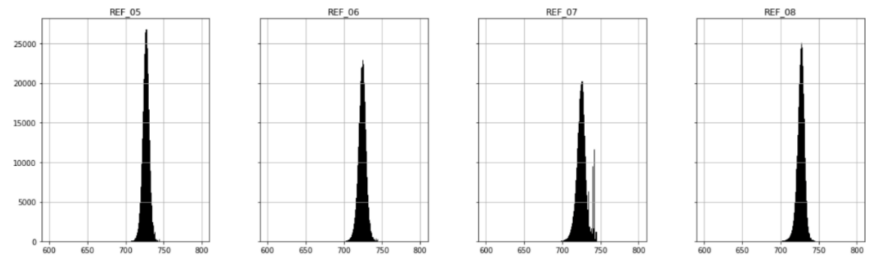


Fig.2.17 Data distribution from sensors 5, 6, 7 and 8

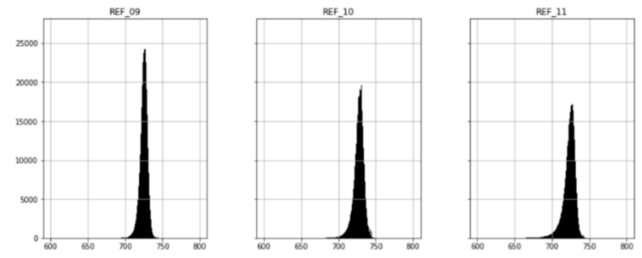


Fig.2.18 Data distribution from sensors 9, 10 and 11

The first thing we can notice about the images above is that the data appear to have the basic characteristics of a normal distribution. However, not all are the same, which confirms that each sensor has certain behavior and characteristics. It is important to note that the mean coincides with the mode in most of the histograms, which is necessary for the application of some anomaly detection techniques. However, an exception was found in sensor 4, where the first half of the values between 733 and 734 nanometers appear with a higher frequency than expected, even higher than average. This behavior is also observed in sensors 2 and 7, where certain values, all of which are above the mean, deviate from the typical curve and symmetry of a normal distribution. Given that these atypical values fall within the operating range of the sensors, it is believed that this behavior is a result of the system itself, which may be favoring the appearance of certain values in certain areas of the foil.
Another interesting aspect to highlight is that in some sensors a greater number of measurements appears below 700 nanometers, which does not occur in the same way above 800 nanometers.

2.3.4. Correlation

In order to gain a deeper understanding of the behavior of the sensors, a correlation analysis was conducted. The Pearson coefficient was used to measure the strength of the linear relationship between the data samples collected by the sensors. The Pearson coefficient ranges from -1 to 1, with -1 indicating a completely negative correlation, 1 indicating a completely positive correlation, and 0 indicating no correlation. A value below -0.5 or above 0.5 typically indicates a strong correlation. The coefficient was calculated for all pairs of sensors and a diagonal correlation matrix was used to visualize the results of the analysis.

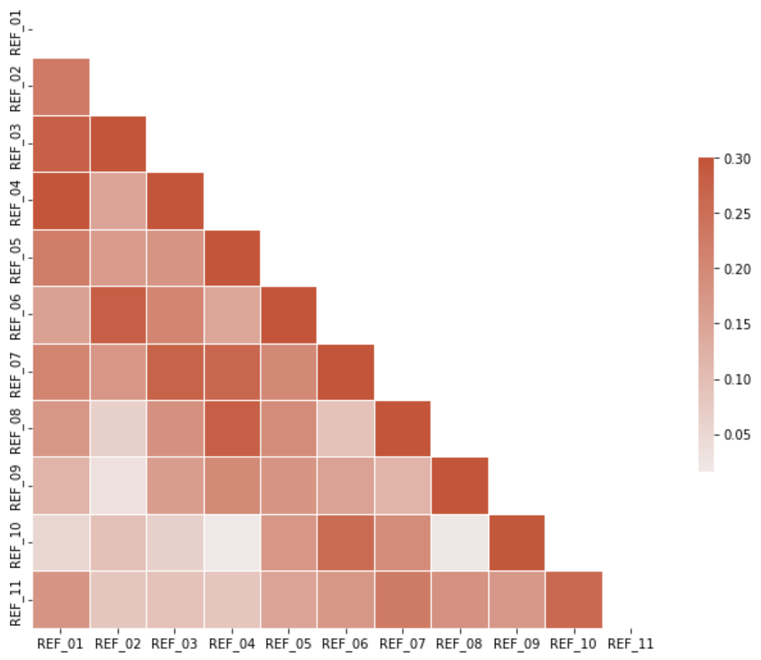


Fig.2.19 Diagonal correlation matrix using Pearson coefficient

The previous image of the correlation matrix shows some key characteristics. First, we can see that the correlation coefficients range from 0 to 0.3, indicating a weak positive correlation between the sensors. Additionally, it can be observed that the neighboring sensors have the highest correlation coefficients, with the highest value being between sensors 5 and 6.

This is expected as sensors that are physically closer to each other have a higher probability of measuring similar values and therefore have a stronger correlation. On the other hand, the lowest correlation is found between the extreme sensors, where we can notice that sensor 10 has a very weak correlation with sensors 1, 2, 3 and 4.

In the matrix of figure 2.19, we can also notice a value that moves away from what has been established up to now, since we can notice a usually low value for the pair of sensors 10 and 8, where despite being close to each other, it has the lowest coefficient of all.

In conclusion, the data analysis chapter aimed to understand the characteristics and limitations of the data collected by the 11 light sensors. We evaluated the credibility and completeness of the data set, established limitations, and performed necessary preparation and processing steps. We visualized the data to get a general idea of its appearance and studied descriptive statistics such as the mean and standard deviation. Additionally, we analyzed the distribution of each sensor and the correlation between sensors using the Pearson coefficient. This comprehensive data analysis allowed us to get a better insight into the behaviors of the system, which will be useful in our further study and solution to the problem.

It is important to study the anomaly literature before applying any techniques for anomaly detection as it provides a deep understanding of the problem and the various approaches that have been taken to solve it. This helps to choose the most appropriate method for the specific problem, taking into account the characteristics of the data, the goals, and the limitations of each technique. In addition, the study of the literature on anomalies makes it possible to take advantage of existing knowledge and make contributions to the field, through the development of new techniques or new approaches.

CHAPTER 3. ANOMALY LITERATURE

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior. These non-conforming patterns are often referred to as anomalies, outliers, discordant observations, exceptions, surprises, peculiarities or contaminants in different application domains. Of these, anomalies and outliers are two terms used most commonly in the context of anomaly detection; sometimes interchangeably.

3.1. Types of Anomalies

Despite the fact that Charau Aggarwal, author of the book "Outlier Analysis", defined anomalies and outliers as synonyms, Varun Chandola tries to give a more precise meaning to the term "anomaly", classifying it into the following three categories

3.1.1. Point Anomalies

If an individual data instance can be considered as anomalous with respect to the rest of data, then the instance is termed as a point anomaly. This is the simplest type of anomaly and is the focus of majority of research on anomaly detection.

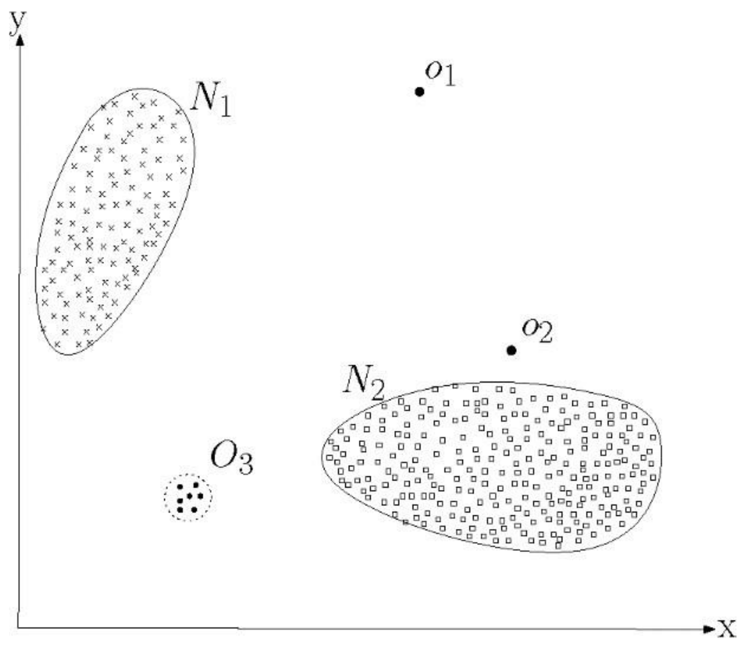


Fig.3.1 Example of anomalies in a 2-dimensinal data set

For example, in Figure 3.1, points O1 and O2 as well as points in region O3 lie outside the boundary of the normal regions, and hence are point anomalies since they are different from normal data points.

3.1.2. Contextual anomalies

If a data instance is anomalous in a specific context but not otherwise, then it is termed as a contextual anomaly (also referred to as conditional anomaly).
The notion of a context is determined by the structure in the data set. Each data instance is defined using following two sets of attributes:

- Contextual attributes. The contextual attributes are used to determine the context, or neighborhood, for that instance. For example, in spatial data sets, the longitude and latitude of a location are the contextual attributes. In time-series data, time is a contextual attribute which determines the position of an instance on the entire sequence.
- Behavioral attributes. The behavioral attributes define the non-contextual characteristics of an instance. For example, in a spatial data set describing the average rainfall of the entire world, the amount of rainfall at any location is a behavioral attribute.

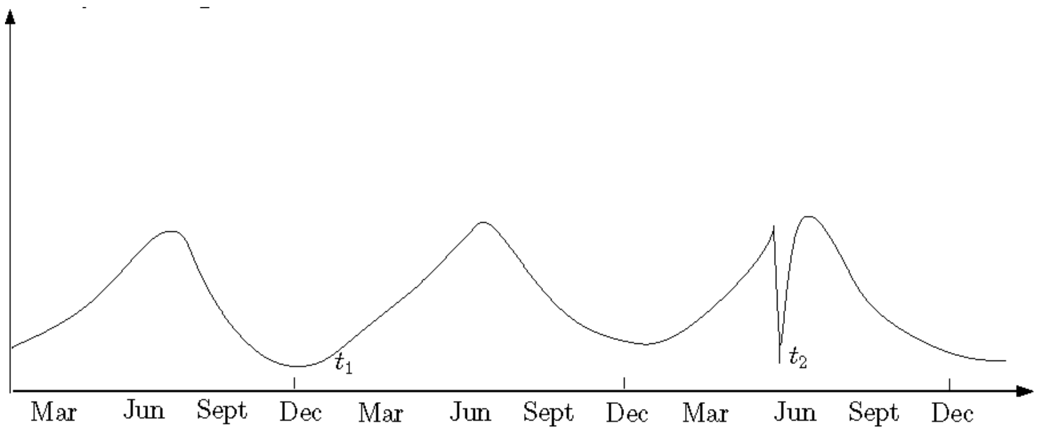


Fig.3.2 Example of anomalies in a 2-dimensinal data set

The example above shows the monthly temperature. As you can see, the t1 and t2 values are more or less the same, but t2 occurs in a different context. Although the value of t2 is in the range of normal values, this temperature is unusual in June.

Some of the ways in which contextual attributes can be defined are:

- Spatial: The data has spatial attributes, which define the location of a data instance and hence a spatial neighborhood.
- Graphs: The edges that connect nodes (data instances) define neighborhood for each node.
- Sequential: The data is sequential, i.e., the contextual attributes of a data instance are its position in the sequence. Time-series data has been extensively explored in the contextual anomaly detection category.
- Profile: Often times the data might not have an explicit spatial or sequential structure, but can still be segmented or clustered into components using a set of contextual attributes. These attributes are typically used to profile and group users in activity monitoring systems, such as cell-phone fraud

detection and credit-card fraud detection. The users are then analyzed within their group for anomalies.

3.1.3. Collective anomalies

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

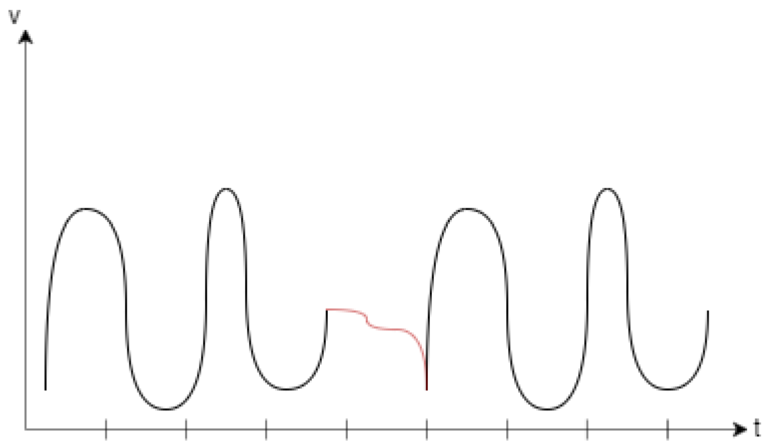


Fig.3.3 Example of anomalies in a 2-dimensinal data set

In the following table I want to display a signal over time. In this you can notice an area where it is anomalous that these values remain at that level for so long.

It should be noted that while point anomalies can occur in any data set, collective anomalies can occur only in data sets in which data instances are related. In contrast, occurrence of contextual anomalies depends on the availability of context attributes in the data.

3.2. Data labels

The labels associated with a data instance denote if that instance is normal or Anomalous. Labeling is often done manually by a human expert and hence requires substantial effort to obtain the labeled training data set. Typically, getting a labeled set of anomalous data instances which cover all possible type of anomalous behavior is more difficult than getting labels for normal behavior.
In certain cases, such as air traffic safety, anomalous instances would translate to catastrophic events, and hence will be very rare.
Based on the extent to which the labels are available, anomaly detection techniques can operate in one of the following three modes:

3.2.1. Supervised anomaly detection

Techniques trained in supervised mode assume the availability of a training data set which has labeled instances for normal as well as anomaly class. Typical

approach in such cases is to build a predictive model for normal vs. anomaly classes.
One thing to watch out for is unbalanced data, as anomalous instances are much less compared to normal instances in the training data.

3.2.2. Semi-Supervised anomaly detection

Techniques that operate in a semi-supervised mode, assume that the training data has labeled instances for only the normal class.
The typical approach used in such techniques is to build a model for the class corresponding to normal behavior, and use the model to identify anomalies in the test data.

3.2.3. Unsupervised anomaly detection

Techniques that operate in unsupervised mode do not require training data, and thus are most widely applicable. The techniques in this category make the implicit assumption that normal instances are far more frequent than anomalies in the test data. If this assumption is not true then such techniques suffer from high false alarm rate.

3.3. Output of Anomaly Detection

An important aspect for any anomaly detection technique is the manner in which the anomalies are reported. Typically, the outputs produced by anomaly detection techniques are one of the following two types:

3.3.1. Scores

Scoring techniques assign an anomaly score to each instance in the test data depending on the degree to which that instance is considered an anomaly. Thus, the output of such techniques is a ranked list of anomalies. An analyst may choose to either analyze top few anomalies or use a cut-off threshold to select the anomalies.

3.3.2. Labels

Techniques in this category assign a label (normal or anomalous) to each test instance.
Techniques that provide binary labels to the test instances do not directly allow the analysts to make a choice based on a specific domain, though this can be controlled indirectly through parameter choices within each technique.

CHAPTER 4. APPLYING ANOMALY DETECTION TECHNIQUES

After gaining an understanding of the various types of anomalies and their characteristics, the next step in the process is to implement various techniques for detecting them.

In previous chapters we have studied the data set and extracted certain features. This extracted information can help us choose which techniques are best suited to our problem.

These techniques have been classified into several groups depending on the fundamental principle upon which they are founded.

4.1. Statistical techniques

The earliest methods for outlier detection were rooted in probabilistic and statistical models and date back to the 19th century [4].

The fundamental principle behind these statistical anomaly detection techniques is that anomalies are observations that deviate significantly from the expected pattern or model of the data: “An anomaly is an observation which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed" [5].

Statistical anomaly detection techniques are based on the assumption that normal data instances occur in high-probability regions of a stochastic model, while anomalies occur in low-probability regions.

Stochastic models are mathematical models that describe systems that behave in a random or uncertain manner, like our system. These models are used to study and make predictions about systems where the outcomes are influenced by random variables. For example, stock prices, weather patterns, or even human behavior, can all be modeled using stochastic models. These models are important because they help us to understand and analyze complex systems that are subject to uncertainty, and provide us with a framework for making informed predictions about the future behavior of these systems.

In this category of anomaly detection techniques, I will study the following techniques:

- 3-sigma-rule
- Interquartile range rule
- Gaussian mixture model

Such techniques assume that the data is generated from a Gaussian distribution.

Based on the evaluations conducted in sections 2.2.2 and 2.2.3, it has been determined that the most appropriate approach for identifying anomalies in our data is to assume that they are contextual anomalies. As a result, the techniques chosen for this analysis have been implemented with the consideration of separating the data into distinct profiles, each corresponding to a specific sensor in our system.

4.1.1. The 3σ-Rule

The 3-sigma rule, also known as the Shewhart's rule, empirical rule or 68-95-99.7 rule, is a straightforward method for identifying outliers frequently used in the process quality control domain [6].

This rule is a statistical guideline that states that for a normal distribution, 68% of the data falls within one standard deviation (σ) of the mean (μ), 95% of the data falls within two standard deviations of the mean, and 99.7% of the data falls within three standard deviations of the mean. In other words, the 3-sigma rule states that almost all (99.7%) of the data in a normal distribution falls within three standard deviations from the mean. This rule is useful in detecting outliers or anomalies in the data, as values that fall outside of the 3-sigma range can be considered unusual or unexpected.

This technique is based on the idea that data points that fall outside of a range defined by 3 standard deviations (σ) from the mean (μ) of a distribution should be considered as outliers, as they are unusual or unexpected.

As seen in the graph below, in a normal distribution, the region defined by μ ± 3σ contains approximately 99.7% of all data points.

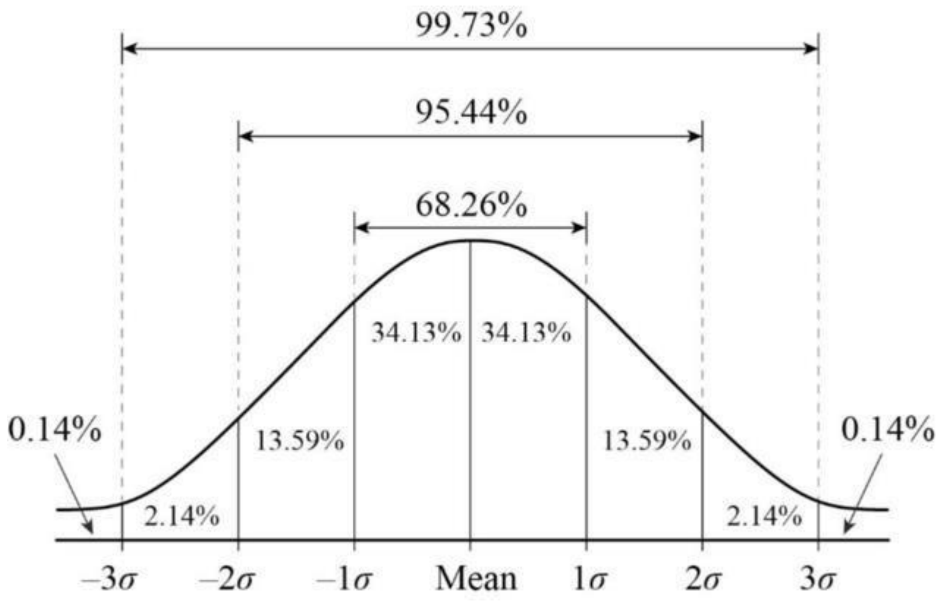
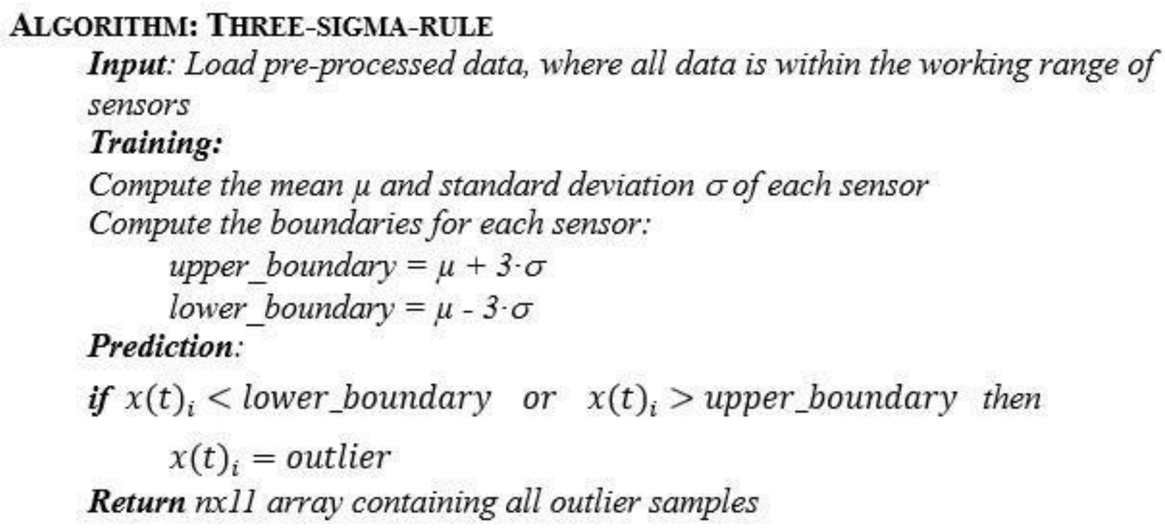


Fig.4.1 Density function that follows a normal distribution and displays the amount of data contained by the number of standard deviations from the mean.

4.1.1.1. Methodology

To apply the technique described, the following algorithm has been followed:



```
ALGORITHM: THREE-SIGMA-RULE
    Input: Load pre-processed data, where all data is within the working range of sensors
    Training:
    Compute the mean $\mu$ and standard deviation $\sigma$ of each sensor
    Compute the boundaries for each sensor:
        upper_boundary = $\mu + 3 \cdot \sigma$
        lower_boundary = $\mu - 3 \cdot \sigma$
    Prediction:
    if $x(t)_i < lower\_boundary$ or $x(t)_i > upper\_boundary$ then
        $x(t)_i = outlier$
    Return nx11 array containing all outlier samples
```

Fig.4.2 Pseudocode of the three-sigma-rule algorithm

4.1.1.2. Results

The limits found after applying the described algorithm are in the following table:

Sensor	Lower boundary	Upper boundary
1	703.31	743.56
2	699.62	745.93
3	708.89	741.94
4	701.62	749.25
5	712.26	740.44
6	707.31	740.06
7	704.54	744.84
8	710.56	742.27
9	708.37	741.03
10	705.41	748.82
11	693.29	753.59

Fig.4.3 Table of the 3-sigma-rule boundaries

By counting the measurements from each sensor that fulfill the criteria in relation to the utilized dataset, the following percentages were obtained.

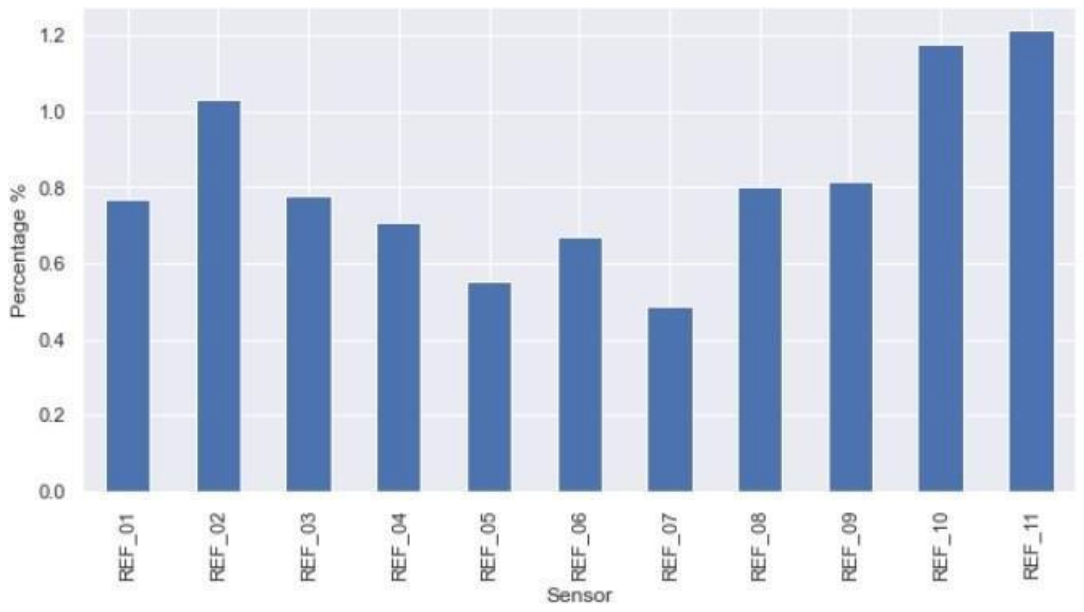


Fig.4.4 Percentage of measurements that meet the condition of the 3-sigma rule for each sensor

With the analysis of these results, we are able to understand the pattern of errors in relation to the location of the sensors. It appears that errors are less frequent in the middle area of the foil, while the extreme area where sensors 10 and 11 are located have a higher rate of errors.

Number of sensors affected	Number of samples	Percentage
1	66528	5.93%
2	8651	0.77%
3	2177	0.19%
4	802	0.07%
5	363	0.03%
6	254	0.02%
7	156	0.01%
8	100	<0.01%
9	83	<0.01%
10	63	<0.01%
All	69	<0.01%

Fig.4.5 Table showing the number and percentage of samples according to the number of sensors that meet the condition of the 3-sigma rule

On the other hand, we can study how many errors occur in relation to time. The table above shows the number of samples affected by at least 1 sensor.

These results show us that almost 6% of the data set contains at least 1 sensor that is classified as outlier considering the 3-sigma rule. These values change dramatically in the case of more than 1 sensor involved, decreasing to less than 1%.

Another observation is that the scenario where all the sensors fulfill the condition of the 3-sigma rule (0.00615%) is more frequent than when only 10 of them fulfill it (0.0056%).

4.1.1.3. Interpretation

The results obtained with this technique appear to be quite reasonable. The percentage of anomalies per sensor is generally below 1%, with the exception of sensors 10 and 11. This suggests that there is a specific area on the foil where unexpected values tend to occur. Possible causes for this could include wear and tear on these sensors, resulting in reduced reliability and accuracy.

Another reason is that there are some asymmetries in the environment. Small defects in the vacuum chamber where the sensors are located may be producing certain reflections that interfere with the sensors, resulting in anomalous values.

The other reason may come from the process itself, where at some point in the production chain there is a difference in the processes, producing an asymmetry in the result obtained.

The 3-sigma deviation rule can be a good technique for detecting unsupervised anomalies as is a well-established method and has been widely used for many years, making it a reliable and trusted. Also is easy to understand and implement, and can handle large amounts of data effectively.

However, the 3-sigma deviation rule assumes that the data follows a normal distribution, which may not always be the case. Also uses a fixed threshold, which may not be appropriate for all datasets. This can lead to false positives or negatives if the threshold is not set correctly. In some cases, the 3-sigma deviation rule may not be accurate in detecting anomalies, especially if the data has complex patterns. This is because it only considers the mean and standard deviation of the data.

4.1.2. Interquartile Range Rule

The box plot rule or Interquartile range rule is one of the simplest statistical techniques that has been applied to detect univariate and multivariate anomalies in some fields like medical domain [7] or turbine rotors data [8]

A boxplot is a standardized way of displaying the dataset based on the five-number summary: the minimum, the maximum, the sample median, and the first and third quartiles.

- Minimum (Q0 or 0th percentile): the lowest data point in the data set excluding any outliers.
- Maximum (Q4 or 100th percentile): the highest data point in the data set excluding any outliers.
  Median (Q2 or 50th percentile): the middle value in the data set.
- First quartile (Q1 or 25th percentile): the median of the lower half of the dataset. Is the number that marks one quarter of the ordered data set. In other words, there are exactly 25% of the elements that are less than the first quartile and exactly 75% of the elements that are greater than it.
- Third quartile (Q3 or 75th percentile): the median of the upper half of the dataset. There are exactly 75% of the elements that are less than the third quartile and 25% of the elements that are greater than it

The quantity Q3-Q1 is called the Inter Quartile Range (IQR). The box plots also indicates the limits beyond which any observation will be treated as an anomaly. A data instance that lies more than 1.5 * IQR lower than Q1 or 1.5 * IQR higher than Q3 is declared as an anomaly. The region between Q1-1.5IQR and Q3 +1.5IQR contains 99:3% of observations, and hence the choice of 1.5IQR boundary makes the box plot rule equivalent to the 3-sigma technique for Gaussian data.

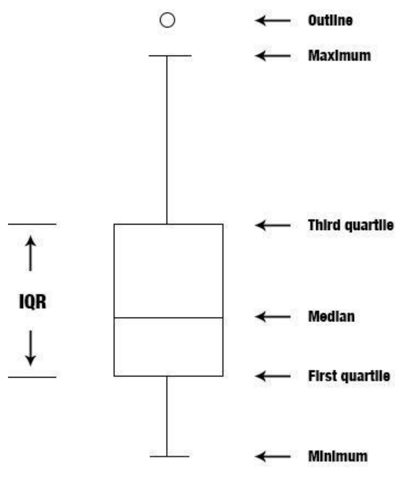
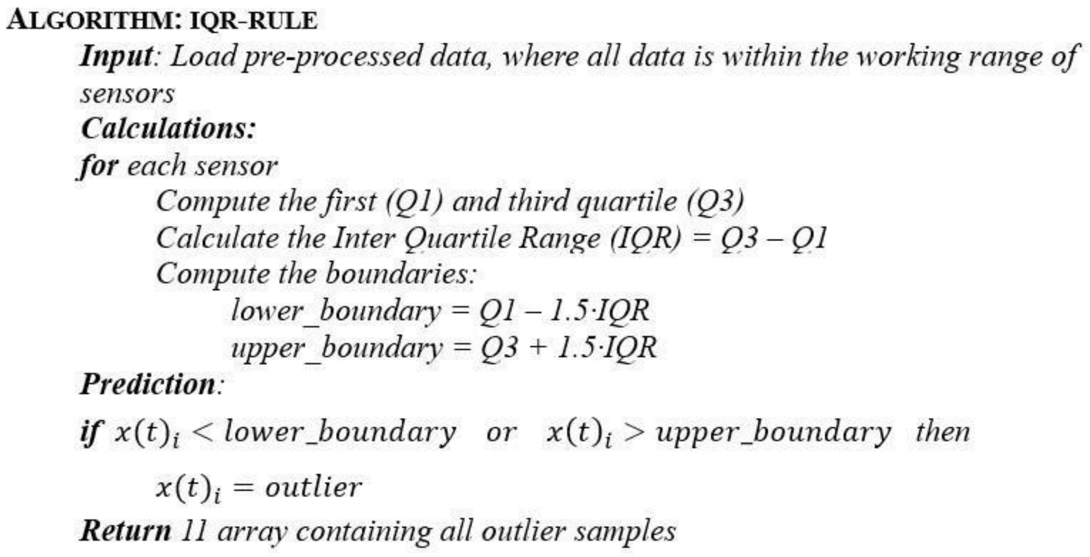


Fig.4.6 Box plot description

4.1.2.1. Methodology

To apply the technique described, the following algorithm has been followed:



```
ALGORITHM: IQR-RULE
    Input: Load pre-processed data, where all data is within the working range of
    sensors
    Calculations:
    for each sensor
        Compute the first (Q1) and third quartile (Q3)
        Calculate the Inter Quartile Range (IQR) = Q3 – Q1
        Compute the boundaries:
            lower_boundary = Q1 – 1.5·IQR
            upper_boundary = Q3 + 1.5·IQR
    Prediction:
    if $x(t)_i < lower\_boundary$ or $x(t)_i > upper\_boundary$ then
        $x(t)_i = outlier$
    Return 11 array containing all outlier samples
```

Fig.4.7 Pseudocode of the Interquartile Range rule

4.1.2.2. Results

The limits found after applying the described algorithm are the following:

Sensor	Lower boundary	Upper boundary
1	708.35	738.55
2	706.73	738.53
3	712.8	738.08
4	708.76	742.2
5	715.82	736.9
6	711.31	736.07
7	710.61	738.61
8	715.2	737.72
9	712.87	736.64
10	712.09	742.4
11	706.07	741.39

Fig.4.8 Table of the IQR-rule boundaries

The results obtained are in line with what was predicted, as we are obtaining more stringent limits than when using the 3-sigma rule. This is due to the fact that the region between Q1-1.5IQR and Q3+1.5IQR captures 99.3% of the data, as opposed to 99.73% for the μ ± 3σ region of the 3-sigma rule. Having a smaller percentage of data in this region means that there will be additional outliers, thus resulting in stricter limits.

By counting the measurements from each sensor that are outside the boundaries in relation to the utilized dataset, the following percentages were obtained.

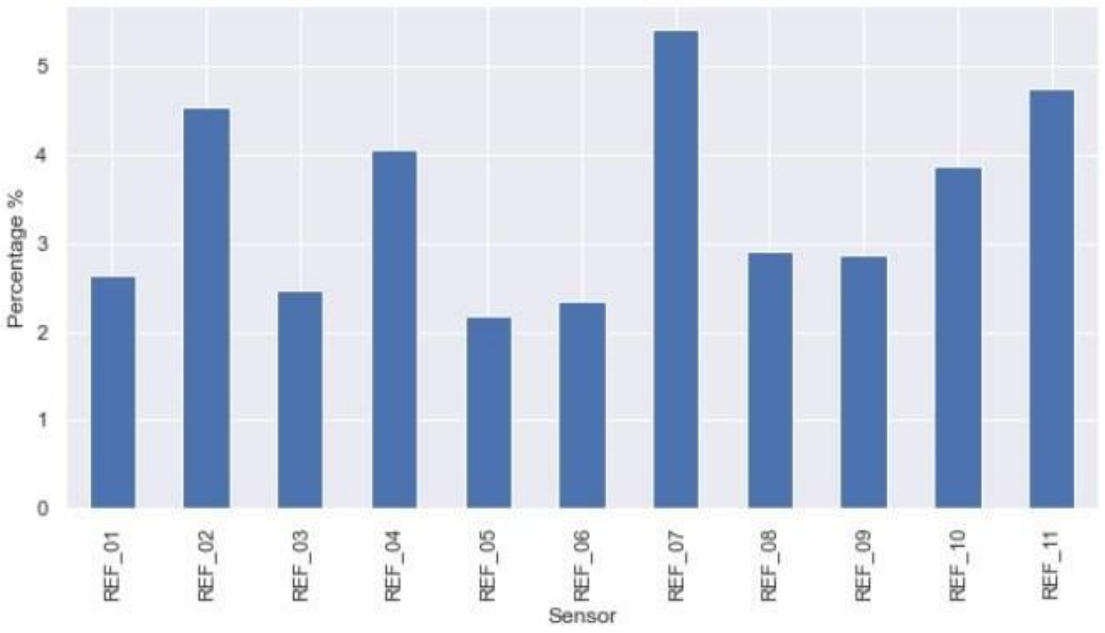


Fig.4.9 Percentage of measurements that meet the condition of the 1.5iqr-rule for each sensor

Looking at these results you can easily see a notable increase in the number of samples considered as anomalies, if we compare it with the 3-sigma-rule.
If we make this same comparison looking at the bar graph, we can see some different and curious behaviors. On the one hand we have that sensor 4 has gone from being the 4t with the lowest percentage of anomalous samples to being the 4t with the highest number. Along these same lines, sensor 7 has gone from being the sensor with the lowest percentage of anomalous samples to being the one with the most, this time leaving sensor 11 in second place.

These sudden changes in behavior can be attributed to the fact that sensors 4 and sensor 7 have certain values that occur at a much higher frequency than expected. With stricter limits in place, these values are now being classified as anomalies.

If we study the number of samples over time according to the number of affected sensors, we obtain:

Number of sensors affected	Number of samples	Percentage
1	231998	20.7%
2	55591	4.96%
3	14121	1.26%
4	4819	0.43%
5	1905	0.17%
6	673	0.06%
7	336	0.03%
8	168	0.015%
9	135	0.012%
10	101	0.009%
All	123	0.011%

Fig.4.10 Table showing the number and percentage of samples according to the number of sensors that meet the condition of the IQR rule

When comparing the results obtained with the 3-sigma rule, it is observed that there is an increase between 3 and 6 times greater in the number of anomalous samples, depending on the number of affected sensors. In particular, when a single sensor is affected, 20.7% of the samples are considered anomalous, which from a business point of view may be considered an inefficiently large number. This means that out of every 5 samples, 1 would need to be reviewed.

The image below displays a boxplot representation of the data collected from each sensor. The boxplots are arranged horizontally for improved readability and were generated using the boxplot function from the Seaborn [9] Python data visualization library.

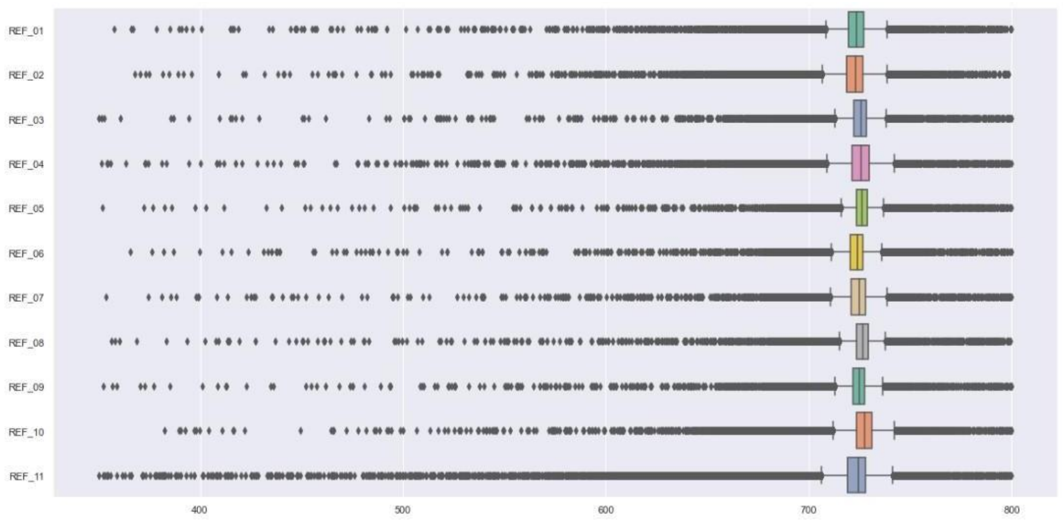


Fig.4.11 Horizontal boxplot representation of each sensor

Upon examination of the box plot graph, it is apparent that the lower "whisker" of the box is truncated more abruptly than the upper ones, where the number of anomalies decreases as the wavelength value decreases. This is a result of the preprocessing that was conducted on the data, where data points that fell outside of the operating wavelength range of the sensors were replaced.

4.1.2.3. Interpretation

The results obtained with this technique appear to be quite large in some cases, but are still within reasonable limits.

The most striking result is the percentage of samples affected by a sensor, which can reach up to 20%. This means that 1 out of every 5 samples needs to be reviewed, or in other words, a significant deviation is occurring every 10 seconds on one of the sensors. While this may seem high, it must be considered in the context of the problem and the industry. For example, in another process of the same company, up to 40% of samples may be discarded, depending on the desired quality for the client.

Another characteristic of this system extracted thanks to this technique is the characteristic behavior of the sensor 11, where it differs from the rest by the number of measurements it performs in the shortest wavelength range.

The interquartile range rule is fairly easy to implement and is resistant to outliers since it is based on the quartiles rather than the mean.

However, it may not be appropriate for data sets with non-normal distributions and provides limited information about the distribution of the data

4.1.3. Gaussian mixture model

A Gaussian mixture model (GMM) is a probabilistic model that assumes that the underlying data is generated by a mixture of multiple Gaussian distributions. Each Gaussian distribution represents a cluster of data points, and the mixture model estimates the parameters of each cluster and the probability of each data point belonging to each cluster.

The parameters of this model are estimated using the Expectation-Maximization (EM) algorithm which use the Maximum likelihood estimation (MLE) method [10]. The EM algorithm is a statistical method that finds local maximum likelihood [11] by iteratively updating the parameters of the model. This method involves identifying the parameters θ that yield the highest probability (evaluation function) of the hypothetical data generating function f(X,θ).

$$\theta_{MLE} = argmax\ p(X|\theta) \tag{4.1}$$

Since we are assuming that the data, we are working with was generated by an underlying Gaussian process we will work with the Gaussian likelihood function (L).

$$f = p(\boldsymbol{X}|\theta) = N(\boldsymbol{X}|\theta) = N(\boldsymbol{X}|\mu, \Sigma) \quad \textbf{(4.2)}$$

Therefore, for MLE of a Gaussian model, we will need to find good estimates of both parameters: the mean (μ) and the covariance matrix (Σ) [12]:

$$\mu_{MLE} = argmax_{\mu} N(\boldsymbol{X}|\mu, \Sigma) \quad \textbf{(4.3)}$$

$$\Sigma_{MLE} = argmax_{\Sigma} N(\boldsymbol{X}|\mu, \Sigma) \quad \textbf{(4.4)}$$

In summary, the process involves assuming multiple random components and determining the probability that each data point was generated by each component of the model. Then, the parameters are adjusted to increase the likelihood of the data given these assignments. By repeating this process, the algorithm is guaranteed to converge to a local optimum.

Gaussian mixture models have used to detect strains in airframe data [13], to detect anomalies in mammographic image analysis [14] and for network intrusion detection [15].

4.1.3.1. Methodology

The Gaussian Mixture Model (GMM) algorithm is commonly used as a multivariate classification model [16]. In our case, as the anomalies are assumed to be of the contextual type, this technique will be applied separately for each sensor, rather than applying it to each sample defined by the 11 variables (sensors).

The Mixture package from the Sklearn [17] library has been used to implement this algorithm. This package has different options to restrict the covariance: spherical, diagonal, tied or full covariance. The following image shows an example of how these options behave. The results come from applying to a dataset called 'Iris flower data set' [18].

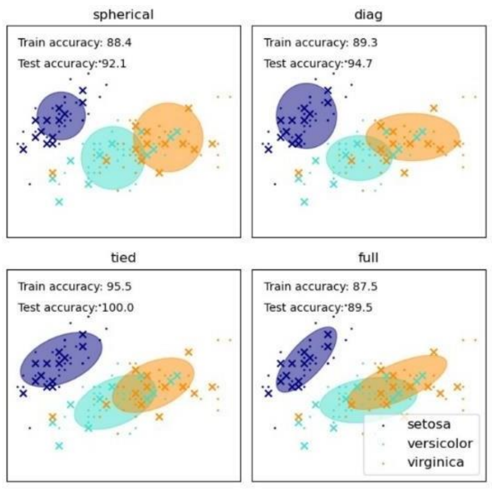
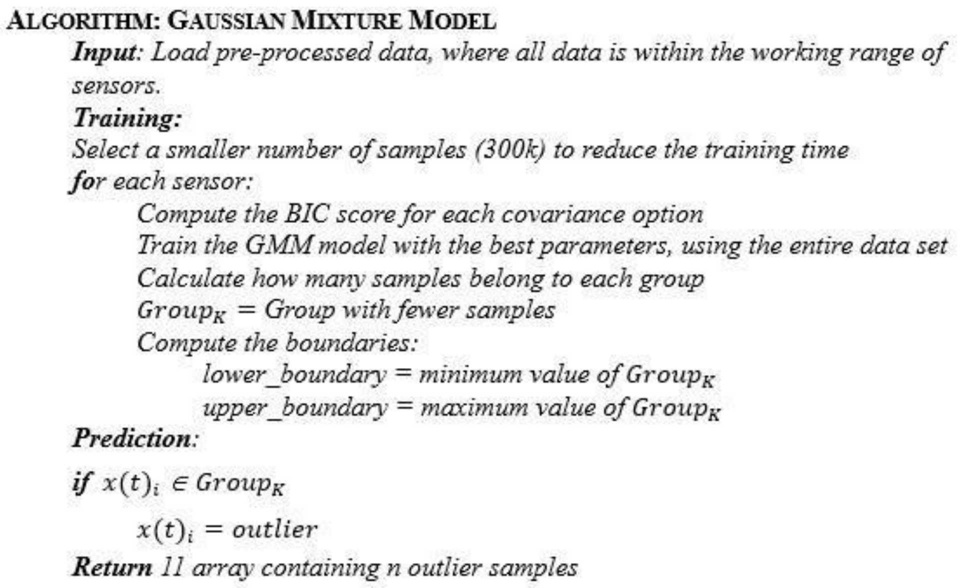


Fig.4.12 Covariance constraint options of the 'Mixture' package applied to the 'Iris flower' data set

The methodology followed to apply this technique is described in the following pseudocode:



```
ALGORITHM: GAUSSIAN MIXTURE MODEL
    Input: Load pre-processed data, where all data is within the working range of
    sensors.
    Training:
    Select a smaller number of samples (300k) to reduce the training time
    for each sensor:
        Compute the BIC score for each covariance option
        Train the GMM model with the best parameters, using the entire data set
        Calculate how many samples belong to each group
        $Group_K$ = Group with fewer samples
        Compute the boundaries:
            lower_boundary = minimum value of $Group_K$
            upper_boundary = maximum value of $Group_K$
    Prediction:
    if $x(t)_i \in Group_K$
        $x(t)_i = outlier$
    Return 1I array containing n outlier samples
```

Fig.4.13 Pseudocode of the Gaussian Mixture model

4.1.3.2. Results

Since this is an unsupervised problem, there is no way of knowing in advance the optimal number of components. Fortunately, being a probabilistic model, metrics such as Bayesian information criterion (BIC) [19] can be used to identify how well the observed data fit the model created, while controlling for excessive overfitting.

As demonstrated in the following images, each sensor will produce different scores. Additionally, even when using the same sensor with the same data set, the algorithm may converge to different values because it is only guaranteed to converge to a local minimum, not a global minimum as mentioned before.

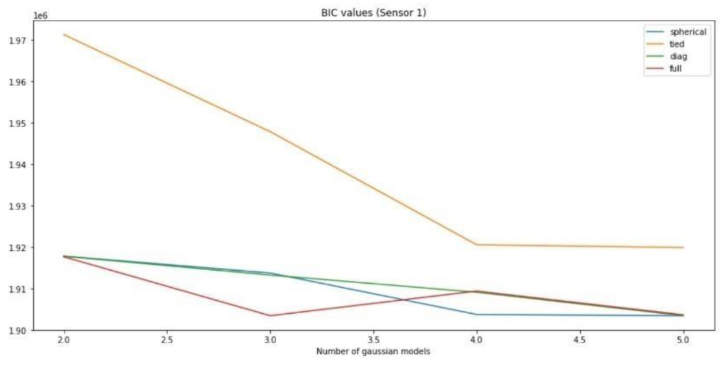


Fig.4.14 Graph showing the BIC score obtained for each covariance option according to the number of components (Gaussian models) for sensor 1.

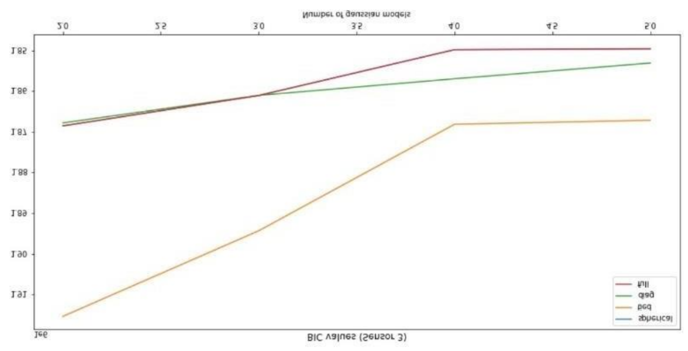


Fig.4.15 Graph showing the BIC score obtained for each covariance option according to the number of components (Gaussian models) for sensor 3

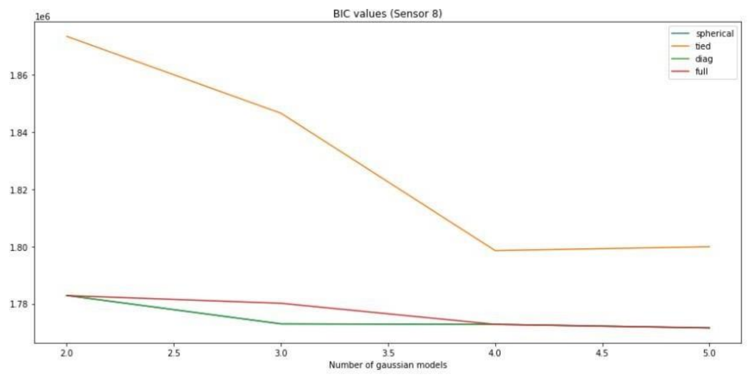


Fig.4.16 Graph showing the BIC score obtained for each covariance option according to the number of components (Gaussian models) for sensor 1.

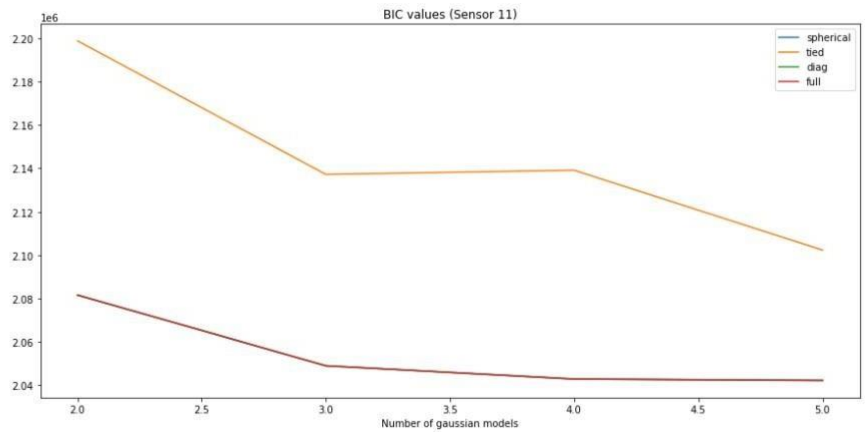


Fig.4.17 Graph showing the BIC score obtained for each covariance option according to the number of components (Gaussian models) for sensor 1.

As we can see in the previous images, each sensor has different parameters. The results of sensors 1,3,8 and 11 have been shown to obtain a general representation of the behavior of each foil zone.

In the case of sensor 1 the ideal model would be a full covariance model with 4 components or a spherical model with 4 components. If we add more components the results hardly improve.

In sensor 3 we see that the best result is obtained with full covariance with 4 components.

In the case of sensor 8, the diagonal covariance model obtains the best score, but it is equalized with the full covariance model by adding one more component.

Finally, we have sensor 11, where the BIC score stagnates at 4 components.

Choosing the best-case scenario for each of the sensors is not always a good option, as we run the risk of overfitting.

Overfitting is a common risk in machine learning models, where a model becomes too complex and fits the noise in the training data, rather than the underlying patterns. For this reason, it has been decided to choose the pair of parameters that works best in general, rather than choosing a specific one for each of the sensors. The result is the full covariance pair with 4 components.

The number of samples classified in each group for sensor 1 are as follows:

Group 1	545016
Group 2	441231
Group 3	128444
Group 4	6081

As we can see the least numerous group for the case of sensor 1 is group 4, with a total number of 6081, quite similar to the 8637 obtained from the 3-sigma-rule.

In the following table I show the number of samples in the group with the least number of samples for each sensor.

Sensor	Less numerous group (number of anomalies)	Percentage over total
1	6081	0.54%
2	34393	3.06%
3	32433	2.89%
4	3871	0.34%
5	26826	2.39%
6	42140	3.75%
7	45236	4.04%
8	5858	0.52%
9	27614	2.46%
10	32979	2.94%
11	5863	0.52%

Fig.4.18 The number of samples and the percentage of the total in the group with the least number of samples, per sensor.

Looking at the results we can notice big differences in the number of anomalies it is detecting, where for some sensors the values are very close to those obtained with the 3-sigma-rule, and others exceed the values obtained with the IQR-rule.

The results obtained may be a consequence of the generalization applied when selecting the parameters. In addition, the nature of the model itself may also be a contributing factor. Specifically, the Sklearn library model assigns a value to one of four groups based on the probability that it belongs to that group and selects the group with the highest probability. However, the model may not be sure to which group a value belongs and assign it to a group anyway, no matter how low the probability of belonging to it.

Even so, the results obtained are interesting for some of the sensors, since it seems to correctly classify the anomalies within one of the distributions created by the model.

In the following image I show all the values of sensor 1, colored according to the group they belong to.

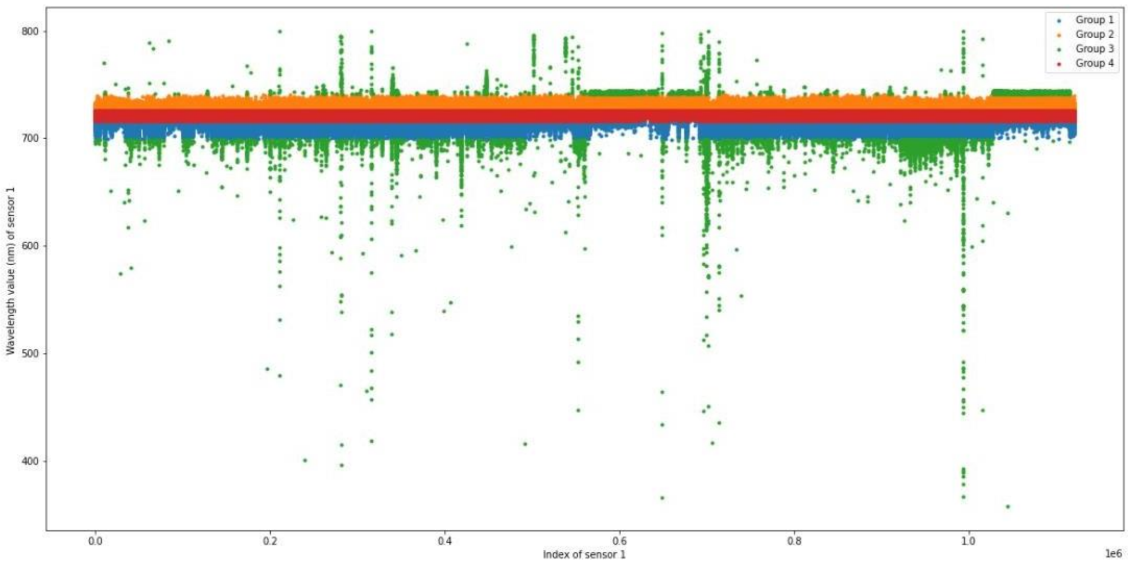


Fig.4.19 Wavelength values of sensor 1, colored according to the group they belong to.

The image above illustrates the four groups clearly. Group 4 is classifying the values that are closest to the mean. Next, we find two groups of values that could have been generated based on one of the distributions created by the model and appear to be within the limits of the previously applied techniques. Lastly, group 3 contains all the suspected outliers, including those with very high and very low values.

By examining the minimum and maximum values of each group, we can establish limits for each group:

Group	Lower boundary	Upper boundary
1	**698.96**	716.62
2	725.77	**741.03**
3	357.8	799.7
4	716.63	725.76

Fig.4.20 Table showing the minimum and maximum value found in each group, for the case of sensor 1.

For Sensor 1, the results are quite satisfactory. By analyzing the area that encompasses the blue, red, and orange groups (excluding group 3), the limits obtained (698.96- 741.03) are very similar to those obtained using the 3-sigma rule (703.31 - 743.56), despite following two reasonably different techniques.

4.1.3.3. Interpretation

In this technique, the same principle has been applied as in the previous two techniques, but with a different approach. Specifically, the values of each sensor have been classified into different Gaussian distributions. The idea behind it is to see if anomalies can be described by a different distribution than the one that generates the normal data. Based on the limitation, the results obtained seem to be quite satisfactory.

Since the basic principle of an anomaly is that they are much less frequent than the normal data, samples that were in the least significant group were marked as anomalous. This resulted in a detection rate of between 0.5% and 4% of anomalies depending on the sensor. When compared to the previous two techniques, these values fall between the results of the 3-sigma rule and IQR techniques. The differences between sensors may be due to the generalization of parameters that was used. Adapting the parameters to each sensor may lead to more consistent results.

It is interesting to note that, for example, in sensor 1, limits were obtained that are very close to those applied with the 3-sigma rule technique. This supports the hypothesis that anomalous data can be characterized according to a specific distribution and that if the probability of it being generated with that distribution is high, it may be an anomaly. This probability can be used as an anomaly score.

GMM can adapt to the specific distribution of the normal data, making it more effective than other techniques that rely on fixed thresholds or statistical measures. Also, the technique can determine probability of a sample being generated by the normal distribution, which can be used as an anomaly score.

In contrast, this technique is sensitive to the number of components and initialization used, so it may require trial and error to find the best parameters for a specific dataset.

4.1.4. Conclusions from the statistical techniques

Once we have seen different techniques using the fundamental principle that anomalies are observations that deviate significantly from the expected pattern or model of the data, here I discuss some conclusions of this type of techniques.

If the assumptions regarding the underlying data distribution hold true, statistical techniques provide a statistically justifiable solution for anomaly detection.

Furthermore, statistical techniques can operate in an unsupervised environment without the need for labeled data.

The key disadvantage of statistical techniques is that they rely on the assumption that the data is generated from a particular distribution. This assumption often does not hold true. Even when the statistical assumption can be reasonably justified, there are several hypotheses that can be applied to detect anomalies; choosing the best statistic is often not a straightforward task.

The implementation of these techniques is relatively easy, however, a major drawback when using them on multivariate data is that they are not able to capture the relationship or interdependence between different attributes.

A limitation of this techniques is that they are unable to detect anomalies that have attribute values that are individually common, but their combination is rare. This is because these techniques are not able to capture the interactions between different attributes, in our case, the sensors.

4.2. Nearest neighbor techniques

Nearest neighbor techniques define a data point as an outlier when its locality (or proximity) is sparsely populated.

The fundamental principle behind these techniques is that normal data instances occur in dense neighborhoods, while anomalies occur far from their closest neighbors.

Nearest neighbor-based anomaly detection techniques require a distance or similarity measure defined between two data instances. Distance (or similarity) between two data instances can be computed in different ways. For continuous attributes, Euclidean distance is a popular choice but other measures can be used. For multivariate data instances, distance or similarity is usually computed for each attribute and then combined.

Nearest neighbor-based anomaly detection techniques can be broadly grouped into two categories:

- Techniques that use the distance of a data instance to its kth nearest neighbor as the anomaly score.
- Techniques that compute the relative density of each data instance to compute its anomaly score.

4.2.1. LOF

The Local Outlier Factor (LOF) is a density-based outlier detection algorithm that was first proposed by Markus M. Breunig in 2000. The algorithm aims to identify instances in a dataset that have a significantly lower density than their surrounding instances. It does this by comparing the density of an instance with the density of its k-nearest neighbors, where k is a user-specified parameter.

The Local Outlier Factor technique has been applied to detect anomalies in some fields like fraud in credit card [20]. Also, a variant of LOF have been applied to detect anomalies in large data streams [21].

For a given data point X ,K-distance is the distance between the point X, and it's Kth nearest neighbor. K-neighbors denoted by $N_k(A)$ includes a set of points that lie in or on the circle of radius K-distance. K-neighbors can be more than or equal to the value of K.

The following image shows an example where we have four points A, B, C and D:

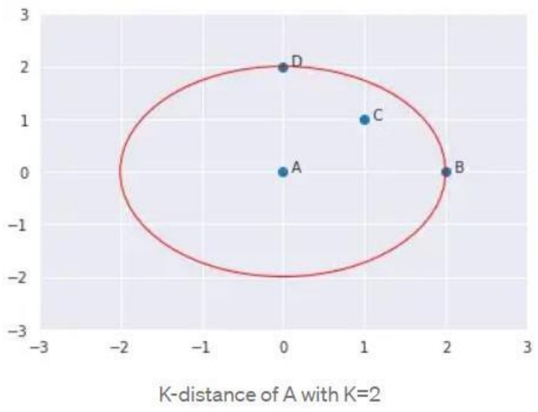


Fig.4.21 Illustrative image of the distance of the 3 closest points to point A

If K=2, K-neighbors of A will be C, B, and D. Here, the value of K=2 but the $||N_2(A)|| = 3$.

Then, the reachability distance is defined as the maximum of K-distance of Xj and the distance between Xi and Xj .

$$RD(X_i, X_j) = max(K - distance\ (X_j), distance\ (X_i, X_j) \quad \textbf{(4.5)}$$

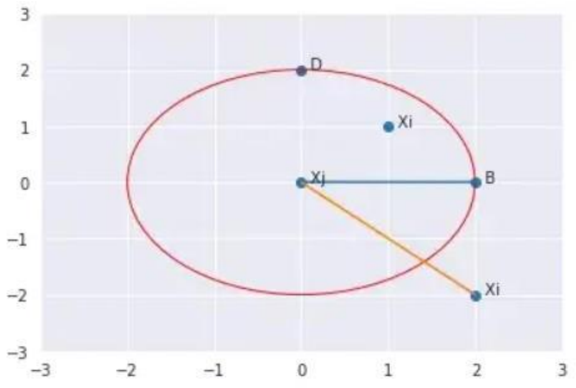


Fig.4.22 Illustration of reachability distance with K=2

In the image above we have an example of reachability, where if a point Xi lies within the K-neighbors of Xj, the reachability distance will be K-distance of Xj (blue line), else reachability distance will be the distance between Xi and Xj (orange line).

Local reachability density (LRD) is inverse of the average reachability distance of A from its neighbors. Intuitively according to LRD formula, more the average reachability distance.

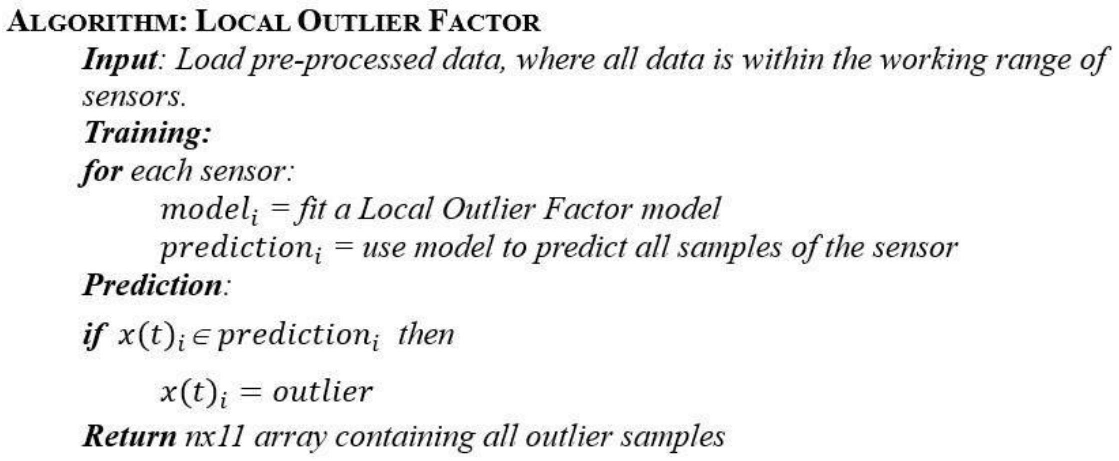
$$LRD_k(A) = \frac{1}{\sum_{X_j \in N_k(A)} \frac{RD(A,X_j)}{|N_k(A)|}} \tag{4.6}$$

LRD of each point is used to compare with the average LRD of its K neighbors. LOF is the ratio of the average LRD of the K neighbors of A to the LRD of A.

$$LOF_k(A) = \frac{\sum_{X_j \in N_k(A)} LRD_k(X_j)}{|N_k(A)|} \times \frac{1}{LRD_k(A)} \tag{4.7}$$

4.2.1.1. Methodology

The procedure used with this algorithm is defined in the following pseudocode:



```
ALGORITHM: LOCAL OUTLIER FACTOR
    Input: Load pre-processed data, where all data is within the working range of
    sensors.
    Training:
    for each sensor:
        $model_i$ = fit a Local Outlier Factor model
        $prediction_i$ = use model to predict all samples of the sensor
    Prediction:
    if $x(t)_i \in prediction_i$ then
        $x(t)_i = outlier$
    Return nx11 array containing all outlier samples
```

Fig.4.23 Pseudocode of the Local Outlier Factor model

4.2.1.2. Results

The number of outliers detected by the model for each sensor is as follows:

- Sensor 1: 1755
- Sensor 2: 2360
- Sensor 3: 1282
- Sensor 4: 2192
- Sensor 5: 1040
- Sensor 6: 1237
- Sensor 7: 1629
- Sensor 8: 1465
- Sensor 9: 1359
- Sensor 10: 1766
- Sensor 11: 2544

As we can observe, a small number of measurements have been identified as abnormal for each sensor. By examining the values of these measurements, we can see that there is a range of values where no anomaly has been identified. By taking the upper and lower bounds of these ranges for each sensor, we obtain the following limits:

Sensor	Lower boundary	Upper boundary
1	706.28	736.76
2	702.81	737.12
3	709.91	736.76
4	707.49	749.25
5	712.26	740.44
6	707.31	740.06
7	704.54	744.84
8	710.56	742.27
9	708.37	741.03
10	705.41	748.82
11	693.29	753.59

Fig.4.24 Table of the boundaries after applying local outlier factor analysis

These limits are quite similar to those obtained through previous techniques, however, the total number of anomalies identified is significantly lower. For instance, if we examine the number of measurements from Sensor 1 that fall outside of these limits, we find 25,314 samples, in contrast to 1,755.

This significant discrepancy could be attributed to the parameters used in this model. In particular, the most crucial parameter to consider is the K-neighbor, which we have set to the default value of 20. This may lead to values that fall outside these limits but have a high concentration of nearby values not being classified as anomalies.

4.2.1.3. Interpretation

With the Local Outlier Factor (LOF) technique, we have obtained some interesting results since the number of samples detected has been very low, between 0.1% and 0.2% per sensor. However, upon further analysis, we discovered that there were certain ranges of values that were not being considered as anomalies. By establishing limits in a similar manner to previous techniques, we found that these limits were very similar to those obtained from other techniques.

When applying these limits, the percentage of anomalous samples increased significantly, for example, from 0.15% to 2.2% for sensor 1. This is because the LOF model identifies anomalies based on local density. If there is a group of similar anomalous samples, the local density in that range of values will be high, and samples from that area will not be considered anomalous.

This can be customized with the "k-neighbors" parameter, which will determine the density of a zone by contemplating more or less points. When working with more than 1 million samples, the default value of 20 may give incorrect results.

The LOF algorithm is based on density, which makes it robust to different types of distributions and is sensitive to both global and local anomalies, which makes it useful for a wide range of application.

Despite the fact that this technique has been applied to each sensor individually in the study carried out, LOF is capable of detecting anomalies in high-dimensional data sets, where other techniques may fail.

The LOF algorithm has some drawbacks, such as its computational cost when used on a large data set in a multivariate approach. It is also sensitive to the selection of the "k-neighbors" parameter, which affects the density calculations of a given area. Additionally, the algorithm may not be effective on data sets with a small number of samples and it is limited to data sets in Euclidean space, so it may not be suitable for certain types of data such as categorical data.

4.2.2. Conclusions from the nearest neighbor techniques

A key advantage of nearest neighbor techniques is that they are unsupervised in nature and make no assumptions regarding the generative distribution of the data. Instead, they are purely data driven. In addition, adapting nearest-neighbor-based techniques to a different data type is straightforward and primarily requires defining a suitable distance measure for the given data.

On the contrary, using these techniques for unsupervised anomaly detection can lead to incorrect labeling of anomalies if the normal instances don't have sufficient nearby neighbors or if the anomalies have sufficient nearby neighbors.

The computational difficulty of the testing phase is a major issue as it involves computing the distance between each instance to determine the nearest neighbors.

The efficacy of a nearest neighbor technique is highly contingent on the distance measure, determined between two data instances, that effectively separates normal and abnormal instances. Defining accurate distance measures between instances can prove challenging depending on the data.

4.3. Linear models techniques

In order to contract the results obtained, different techniques have been tested with different approaches. In this chapter I will analyze a technique where, given their nature, the samples will be studied with a multivariate approach, instead of analyzing them sensor by sensor.

The attributes in real data are usually highly correlated. Such dependencies provide the ability to predict attributes from one another. The notions of prediction and anomaly detection are intimately related. Outliers are, after all, values that deviate from expected (or predicted) values on the basis of a particular model. Linear models focus on the use of inter-attribute dependencies to achieve this goal.

The linear model or spectral techniques are based on the assumption that data can be embedded into a lower dimensional subspace in which normal instances and anomalies appear significantly different. Data points that do not naturally fit this embedding model are, therefore, regarded as outliers. They are also based on the fact that real data attributes tend to be highly correlated. Such dependencies provide the ability to predict attributes from each other.

4.3.1. Autoencoders

Autoencoders are a type of neural network architecture that aims to reconstruct its input data by learning a compressed representation of it, called an encoding.

Neural networks are computational learning models that simulate the human nervous system. They are inspired by the idea that the brain is composed of many interconnected neurons that process information through a series of computations. Neural networks consist of multiple layers of interconnected nodes, called artificial neurons.

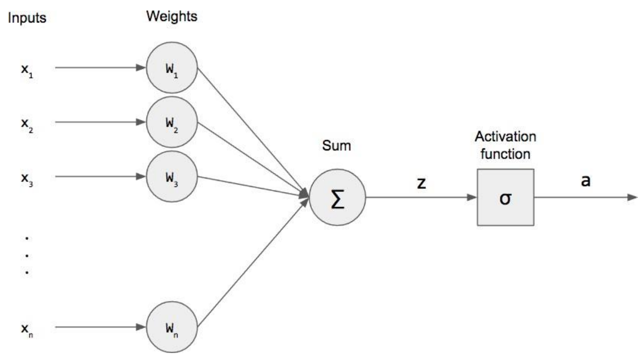


Fig.4.25 Architecture of artificial neuron, also called perceptron

As can be observed in the architecture mentioned above, each neuron receives multiple inputs from other neurons or the initial input, and processes the information by calculating the weighted sum of these inputs. The outcome of this calculation is then processed through an activation function, which determines the output of the neuron.

It is noteworthy that in this study we categorize this type of neural network as a linear model for convenience, but the outcome can be a non-linear model depending on the activation function utilized.

The specific neural network architecture used consist of two main parts: an encoder and a decoder. The encoder maps the input data to a lower-dimensional representation, while the decoder maps this representation back to the original input data. The idea behind autoencoders is to learn a compressed and efficient representation of the input data such that it can be used for tasks such as data visualization, dimensionality reduction or anomaly detection.

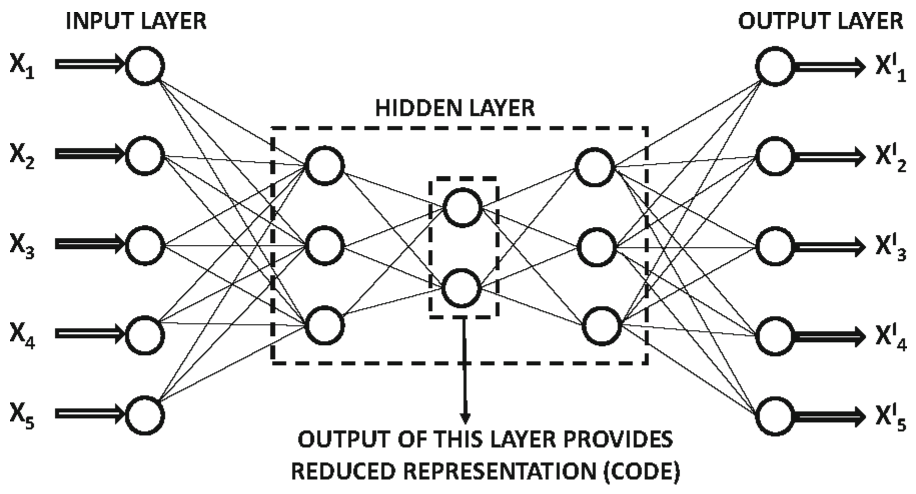


Fig.4.26 Autoencoder architecture

There are 4 hyperparameters that we need to set before training an autoencoder:

- Code size: number of nodes in the middle layer. Smaller size results in more compression.
- Number of layers: the autoencoder can be as deep as we like. In the figure above we have 1 layer in both the encoder and decoder, without considering the input and output.
- Number of nodes per layer: the autoencoder architecture we’re working on is called a stacked autoencoder since the layers are stacked one after another. The number of nodes per layer decreases with each subsequent layer of the encoder, and increases back in the decoder. Also, the decoder is symmetric to the encoder in terms of layer structure. As noted above this is not necessary and we have total control over these parameters.
- Loss function: The loss function is a mathematical function that is used to evaluate the performance of the network. The function compares the network's output to the desired output and calculates the error or difference between the two. Common loss functions used in neural networks include mean squared error, cross-entropy, and hinge loss

In addition to these parameters, other parameters related to how the mode learns are also needed, using a mathematical algorithm called gradient descent [22].

Gradient descent is basically an algorithm that starts with an initial set of parameter values and iteratively updates them in the direction of the negative gradient of the loss function, which represents the direction of steepest descent.

Some of these parameters are the following:

- Batch size: refers to the number of training examples used in one iteration of the gradient descent algorithm. A larger batch size can lead to faster convergence, but also requires more memory and computational resources. On the other hand, a smaller batch size can lead to a more fine-grained update of the model parameters, but may take more iterations to converge
- Epochs: An epoch is a complete iteration over the entire training dataset. The number of epochs determines the number of times the model will see the entire training dataset.
- Learning rate: The learning rate is a hyperparameter in machine learning that determines the step size at which the optimizer updates the model parameters during training. A high learning rate can cause the model to converge quickly, but may lead to overshooting the optimal solution. On the other hand, a low learning rate may converge more slowly, but can lead to finding a more accurate solution. The learning rate must be set carefully, as it can have a significant impact on the performance of the model.

4.3.1.1. Methodology

Determining the number of layers and neurons in an autoencoder is not an exact science. It is a process of trial and error and experimentation. Since we do not have labeled data, it is not possible to assess the performance of the model. Therefore, a series of simple networks have been designed to compare the results with other techniques.

One principle for selecting the number of neurons in the hidden layer can be the Explained Variance.

Explained Variance is a statistical measure of how much of the variation in a dataset can be attributed to each of the principal components generated by the Principal Component Analysis (PCA) method.

PCA is a technique used to reduce the dimensionality of data by finding the directions of maximum variance and projecting the data onto those directions. The amount of variance explained by each direction is called the explained variance and can be used to choose the number of dimensions to keep in a reduced dataset.

There is no specific or typical value of explained variance that is needed to select the number of components in a dataset. Typically, a higher explained variance is preferred as it indicates that more of the variation in the data is being captured by the chosen number of components. A common threshold values used to select the number of components is 95%, so it will be used as a reference.

The following image shows the pseudocode of the application of this methodology.

```
ALGORITHM: THREE-SIGMA-RULE
    Input: Load pre-processed data, where all data is within the working range of
    sensors
    Training:
    Fit the data with PCA model
    Calculate the explained variance as a function of the number of components
    Design different autoencoders neural network architectures:
        Select number of layers
        Select number of neurons for each layer
        Select neural network parameters (batch size, epochs, loss function)
    for each neural network architecture:
        Train the model
        Compute prediction for all samples
        Calculate the mean square error for each prediction = $Y_{error}$
        Compute the mean and standard deviation of this errors
            upper_boundary = $\mu_{prediction-error} - 3 \cdot \sigma_{prediction-error}$
            lower_boundary = $\mu_{prediction-error} + 3 \cdot \sigma_{prediction-error}$
    Prediction:
    if $Y_{error}(t) < lower\_boundary$ or $Y_{error}(t) > upper\_boundary$ then
        $s(t) = outlier$
    Return array containing all outlier samples
```

Fig.4.27 Pseudocode of the autoencoder method

Applying this neural network to our data set will produce a prediction of the value of each sensor. To establish a method for marking a sample as an anomaly, a mean square error has been applied between the prediction and the actual sample. Then, the 3-sigma-rule has been applied with these prediction-errors values.

A batch size of 32, 3 epochs, learning rate of 0.01 and the ReLU [23] loss function were used for all autoencoder architectures.

4.3.1.2. Results

The Explained Variances obtained after applying the PCA method are:

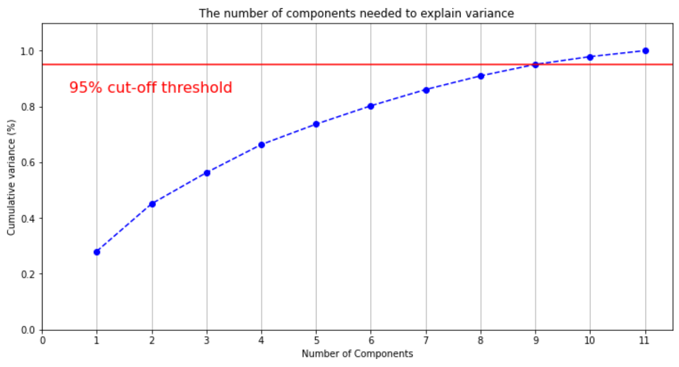


Fig.4.28 Cumulative variance percentage as a function of the number of components

We can clearly see that 9 components are just in the reference threshold of 95% of cumulative explained variance. Reducing the data to 1 component we would maintain 28% of the Explained Variance.

The first architecture that has been used uses two layers as an encoder and for the intermediate layer it uses 9 neurons, thus reducing the dimension to 9. As it is a symmetric architecture, the same structure as the encoder has been used inversely for the decoder. Subsequently, the results have been compared with other architectures.

Depending on the applied architecture we have obtained the following results:

Autoencoder architecture (Nº of neurons in layer 1,…, Nº of neurons in layer n)	Nº of anomalies
(11,10,9,10,11)	8723
(11,9,7,9,11)	5347
(11,6,3,6,11)	7879
(11,9,6,3,6,9,11)	6320

Fig.4.29 Number of anomalies depending on the applied autoencoder architecture

4.3.1.3. Interpretation

With this technique we have used a multivariate approach, defining each sample with 11 dimensions or fields, so the results cannot be compared with those obtained for each sensor with other techniques. Despite this, we tested various network architectures while keeping their hyperparameters constant and found between 5,000 and 9,000 anomalous samples, representing less than 0.1% of the total.

These results are similar to those obtained using the 3-sigma-rule, where at least 2 anomalous sensors were found, or the box-plot rule, where 4 or more anomalous sensors were found.

One interpretation of the results is that the models we designed are not able to accurately reconstruct sensor values when more than two sensors fail. This insight can further bolster the effectiveness of the other techniques, as it suggests that samples with failures in more than two sensors should be closely examined.

Autoencoders are a neural network architecture that can be trained unsupervised, meaning they do not require labeled data to learn patterns in the data, making them suitable for our problem. Autoencoders can be fine-tuned [24] by adjusting the number of layers and neurons in the network, or by using different types of activation functions or loss functions.

Some disadvantages of using autoencoders for unsupervised anomaly detection include:

- Autoencoders require a large amount of data to be trained effectively and may not work well with small datasets.
- Autoencoders can be sensitive to the choice of hyperparameters and may require a lot of fine-tuning to obtain good results.
- Autoencoders may not be able to detect all types of anomalies, especially those that are very different from the normal data used to train the model.
- Autoencoders may require a significant amount of computational resources.

4.3.2. Vector Autoregression

Vector Autoregression (VAR) is a multivariate forecasting algorithm that is used when two or more time series influence each other, and is an extension of the univariate Autoregressive model (AR) model.

Autoregressive (AR) models are a type of time series model that are used to predict future values based on past values. The basic idea behind AR models is to model the current value of a time series as a linear combination of its past values.

In such technique settings, the assumption of temporal continuity plays a critical role in identifying outliers. Temporal continuity refers to the fact that the patterns in the data are not expected to change abruptly, unless there are abnormal processes at work.

The mathematical representation of an AR(p) model is given by the following equation:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_p Y_{t-p} + \varepsilon_t \qquad \textbf{(4.8)}$$

where Y_t is the current value of the time series, α is a constant term, β_1, β_2, ..., β_p are the model parameters, and ε_t is a white noise error term. The value p is the order of the model, representing the number of past values used to predict the current value.

The VAR model uses a linear combination of past values for each variable, including both its own past values and the past values of other variables in the system. It models the relationship between multiple time series as a system of equations, with one equation for each variable.

For example, the system of equations for a VAR(1) model with two time series (variables ‘Y’ and ‘Y2’) is as follows:

$$Y_{1,t} = \alpha_1 + \beta_{11,1} Y_{1,t-1} + \beta_{12,1} Y_{2,t-1} + \varepsilon_{1,t} \qquad \textbf{(4.9)}$$

$$Y_{2,t} = \alpha_2 + \beta_{21,1} Y_{1,t-1} + \beta_{22,1} Y_{2,t-1} + \varepsilon_{2,t} \qquad \textbf{(4.10)}$$

The above equation is referred to as a VAR(1) model, because, each equation is of order 1, that is, it contains up to one lag of each of the predictors (Y1 and Y2).

Before applying the VAR model to detect unsupervised anomalies, it is important to ensure that the time series being analyzed are stationary. This means that the characteristics of the series, such as mean and variance, do not change over time. Without stationarity, the VAR model may not be able to accurately capture the relationships between the variables in the system.

For this study, since the key aspect of using VAR is the correlation between variables, the case of sensors 5 and 6 was examined, as they had the highest correlation between any two sensors in the study conducted in Section 2.3.4.

To evaluate the model's performance, we can calculate the mean square error between the predicted values and the actual values.

4.3.2.1. Methodology

To apply the VAR method to our problem, there are various approaches that can be taken. For instance, one approach is to make predictions by forecasting a specified number of samples using a selected order. For example, if an order of 4 is chosen, the model can predict the next 6 values. This approach is useful if the goal is to predict future means based on past values.

Alternatively, we can also use a slide-window approach where a window of order p is used to predict only the next value. Then, this window is moved one position forward, using the previous p values to predict the next value. The image below illustrates an example of these two different approaches.

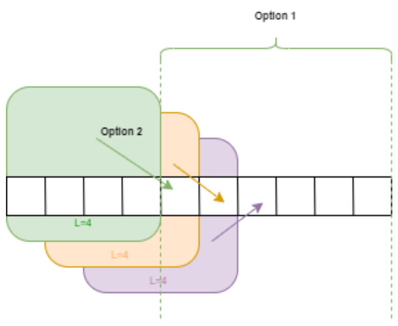
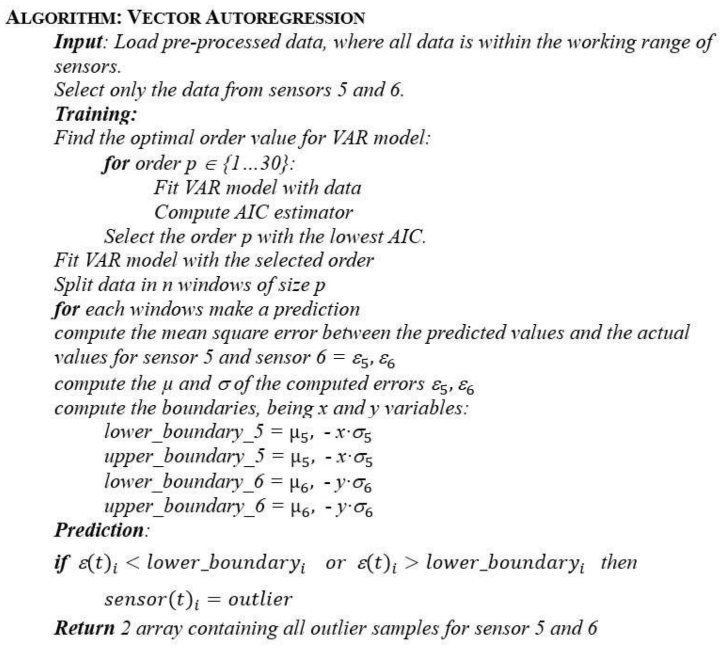


Fig.4.30 Two examples of using the VAR model for prediction are illustrated. The first option involves using 4 previous samples to predict the next 6 samples. The second option utilizes a sliding window of 4 values to predict only the upcoming value.

The pseudocode of the described methodology is shown in the following image:



```
ALGORITHM: VECTOR AUTOREGRESSION
    Input: Load pre-processed data, where all data is within the working range of
    sensors.
    Select only the data from sensors 5 and 6.
    Training:
    Find the optimal order value for VAR model:
        for order p ∈ {1…30}:
                Fit VAR model with data
                Compute AIC estimator
        Select the order p with the lowest AIC.
    Fit VAR model with the selected order
    Split data in n windows of size p
    for each windows make a prediction
    compute the mean square error between the predicted values and the actual
    values for sensor 5 and sensor 6 = $\varepsilon_5, \varepsilon_6$
    compute the $\mu$ and $\sigma$ of the computed errors $\varepsilon_5, \varepsilon_6$
    compute the boundaries, being x and y variables:
        lower_boundary_5 = $\mu_5$, $- x \cdot \sigma_5$
        upper_boundary_5 = $\mu_5$, $- x \cdot \sigma_5$
        lower_boundary_6 = $\mu_6$, $- y \cdot \sigma_6$
        upper_boundary_6 = $\mu_6$, $- y \cdot \sigma_6$
    Prediction:
    if $\varepsilon(t)_i < lower\_boundary_i$ or $\varepsilon(t)_i > lower\_boundary_i$ then
        $sensor(t)_i = outlier$
    Return 2 array containing all outlier samples for sensor 5 and 6
```

Fig.4.31 Vector Autoregression method pseudocode applied

4.3.2.2. Results

The initial steps taken involve ensuring that the time series are stationary, meaning that their characteristics such as mean and variance do not change over time. To accomplish this, the various sensors were examined using the Augmented Dickey-Fuller Test [25], and it was determined that from 1000 samples, a value below the significance level of 0.05 was obtained, indicating that the time series are indeed stationary.

The next step was to determine the appropriate order of the VAR model using the Akaike Information Criterion (AIC) [26], resulting in a value of 26.

In the next image I have plotted the first 26 samples of sensor 5 and 6:

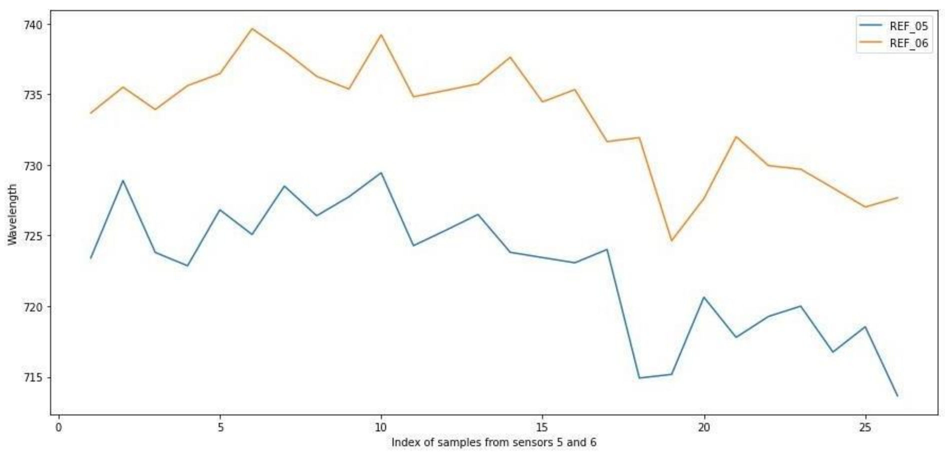


Fig.4.32 Graph of the first 26 samples from sensors 5 and 6 plotted

The graph above illustrates a clear correlation between the two sensors, as can be observed by the trend of the values decreasing and increasing in unison. For example, when one of the two sensors begins to reduce its values, the other sensor too.

With the 26th order VAR model in place, we have examined its performance on our dataset. To visualize this, we have plotted the first 50 predictions made by the model, starting from the 27th sample of the dataset, as the first 26 were needed for the initial prediction. This was done for sensors 5 and 6, and the predictions were compared to the actual values.

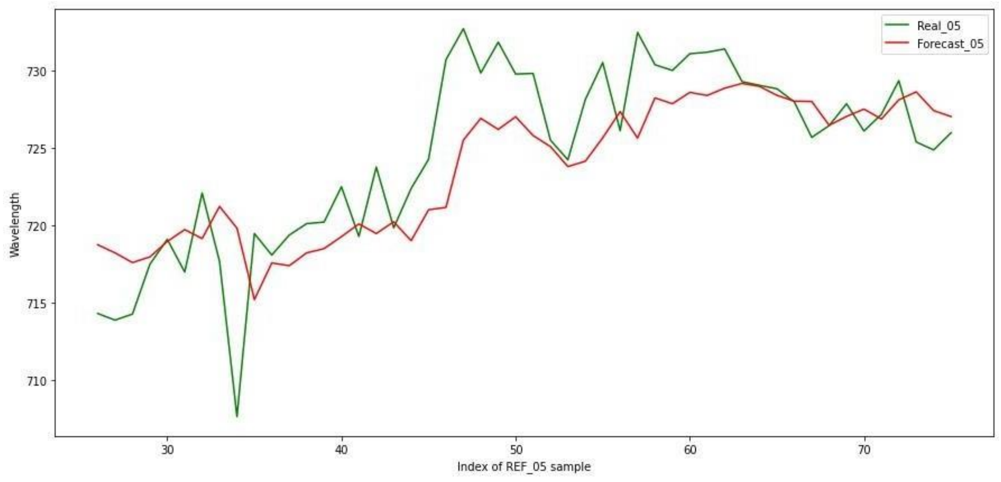


Fig.4.33 Comparison of 50 predictions made by the VAR model for sensor 5 with the actual values

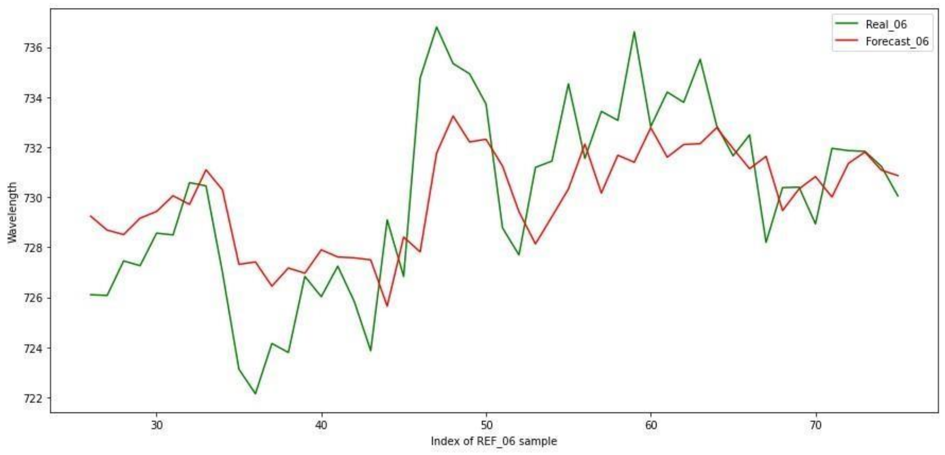
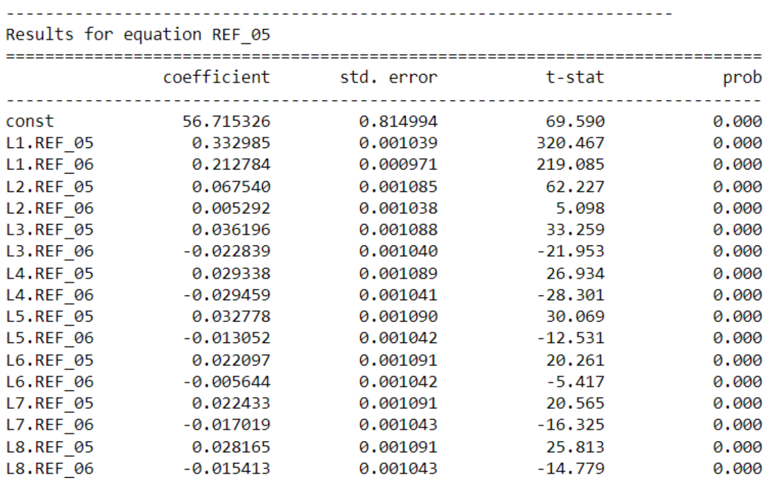


Fig.4.34 Comparison of 50 predictions made by the VAR model for sensor 6 with the actual values

The following image illustrates the coefficients that were calculated during the VAR model fitting process. It is noteworthy that for each previous sample, or lag, there are two coefficients, one for sensor 5 and one for sensor 6 (REF_05 and REF_06). This can be seen by examining the coefficients for each lag, such as lag 1 (L1).



```
--------------------------------------------------------------------
Results for equation REF_05
=============================================================================
                coefficient       std. error           t-stat            prob
-----------------------------------------------------------------------------
const             56.715326         0.814994           69.590           0.000
L1.REF_05          0.332985         0.001039          320.467           0.000
L1.REF_06          0.212784         0.000971          219.085           0.000
L2.REF_05          0.067540         0.001085           62.227           0.000
L2.REF_06          0.005292         0.001038            5.098           0.000
L3.REF_05          0.036196         0.001088           33.259           0.000
L3.REF_06         -0.022839         0.001040          -21.953           0.000
L4.REF_05          0.029338         0.001089           26.934           0.000
L4.REF_06         -0.029459         0.001041          -28.301           0.000
L5.REF_05          0.032778         0.001090           30.069           0.000
L5.REF_06         -0.013052         0.001042          -12.531           0.000
L6.REF_05          0.022097         0.001091           20.261           0.000
L6.REF_06         -0.005644         0.001042           -5.417           0.000
L7.REF_05          0.022433         0.001091           20.565           0.000
L7.REF_06         -0.017019         0.001043          -16.325           0.000
L8.REF_05          0.028165         0.001091           25.813           0.000
L8.REF_06         -0.015413         0.001043          -14.779           0.000
```

Fig.4.35 Values of the coefficients adjusted for the first 8 lags, used for predicting sensor 5

These coefficients show the behaviors learned by the model over a year. The more repeated behaviors will be better predicted, however if a value or set of values is outside of a trend or typical behavior, it will be further from the prediction.

The next step has been to analyze the squared difference between the predictions and the actual values. One way to visualize is to show a bar graph with the values of the squared error and at the same time the real values plotted.

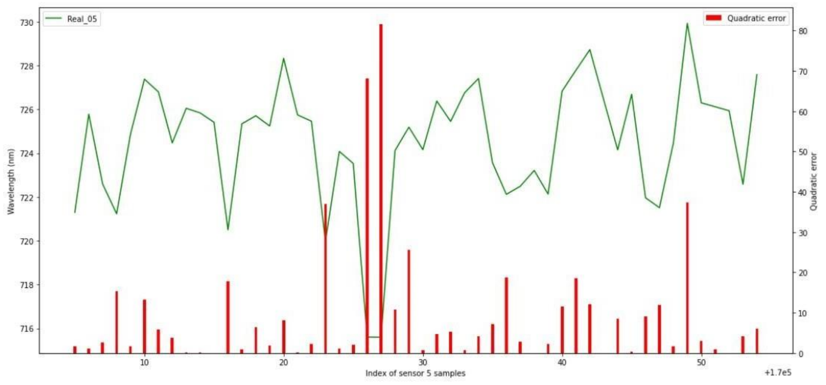


Fig.4.36 Bar graph of the squared error compared to the actual plotted value, for the case of sensor 5

In the previous graph it can be seen how the model has a very large error when a sudden change in trend occurs. However, a sudden change of values that occurs within a context can be well predicted and there is not a high error. For example, if we look at the two highest bars, we can see that the model has not predicted the change that has occurred well, changing from about 723 nanometers to 716. On the other hand, the following sample has a considerable jump in trend, going back at 723 nanometers, however the error is smaller.

To compare the VAR model's performance with other techniques, the number of sensor 5 samples that deviate more than three standard deviations from the mean error was studied. The results showed that 381 samples, or 0.034% of the total, were outside of these limits. This is a significantly lower percentage than what was obtained with other techniques. This is because the model now has variable thresholds, and therefore fewer samples may be outside of those limits.

To further compare, if a threshold of one fifth of a standard deviation was used, resulting in a squared error of more than 90, 0.7% of samples were flagged as anomalous, which is closer to the results obtained with other techniques.

4.3.2.3. Interpretation

The VAR model utilizes a unique approach compared to previous methods by incorporating the use of sliding windows as input, capturing a new dimension, that of time. Also captures the relationship between multiple time series, which can be useful in identifying patterns and trends that may not be visible when considering each time series independently.

This model adapts to local trends, thus changing the threshold over time. With this we are analyzing the anomalies in a contextual way in space and time.

On the one hand we are doing a specific analysis for a couple of sensors. This means that there will be different thresholds depending on the area of the folio, contextualizing according to the space of the folio.

On the other hand, we are taking into account the previous samples of the sensor to determine if the current sample is anomalous or not, and there may be moments in time where certain values could be considered abnormal or not. In this way we are also contextualizing in time.

By making use of these contexts, the model can decide if the same value is outlier or not, depending on the specific space and time that it occurred.

This model can make accurate predictions, as evidenced by its relatively low mean square error. When the 3-sigma rule was applied to these errors, we obtained a minimal number of anomalies. However, by decreasing the error threshold, we can achieve results comparable to other techniques.

One of the primary disadvantages of this model is its complexity, which makes it computationally expensive. Understanding the results can also be difficult as it adapts closely to the data, making it difficult to determine an appropriate threshold to classify anomalies.

To effectively utilize this model, we also need a significant amount of data to accurately estimate its parameters. This is a crucial aspect as the model's performance is also dependent on the selection of parameters.

Another possible problem with this model is the assumption of linearity between the time series, which may not always be the case in real-world data.

4.3.3. Conclusions from the linear model techniques

This chapter presents linear models for outlier detection. Many data sets show significant correlations among the different attributes. In such cases, linear modeling may provide an effective tool for removing the outliers from the underlying data.

Such methods can also be extended to nonlinear models, although the approach is computationally complex and can sometimes overfit the data. Many other mathematical models such as SVMs, matrix factorization and neural networks uses different variations of these concepts. Multilayer neural networks can model complex and nonlinear patterns, especially with the use of deep-learning methods

Different types of outliers can also be defined in time-series data, depending on whether it is desirable to identify deviating points in the series, or whether it is desirable to identify unusual shape subsequences. For the case of multidimensional data, we transform the multivariate time-series to univariate time-series by linearly combining the components of the multivariate time-series.

In summary, linear model-based techniques offer excellent scalability and versatility, making them suitable for a variety of large-scale anomaly detection problems.

On the other hand, linear models are limited in their ability to capture non-linear relationships in the data, which can result in a lower accuracy of anomaly detection compared to non-linear models. Also, linear models can easily overfit to the training data, which can result in poor performance on unseen data.

By last, linear models can be computationally expensive, especially for large datasets, making it challenging to use them in real-time anomaly detection applications.

CHAPTER 5. WEB APP DEVELOPMENT

Creating a web application prototype allows for the efficient testing and comparison of different models and techniques for anomaly detection. By implementing these models and techniques within a web application, it becomes possible to easily test and evaluate each one using the same or other datasets.

This allows for a fair comparison of the models and techniques, as well as a way to quickly identify which one is the most effective for a given dataset.

Additionally, a web application prototype also allows for easy access and user-friendliness. The ability to test and evaluate the models and techniques through a web interface makes it more accessible to a wider range of users, including those who may not be as familiar with programming or data analysis. This can facilitate the collaboration and communication between different company teams members and make it easier to share the results.

Furthermore, this tool can greatly improve the quality control process by providing a more efficient and effective means of identifying anomalies in the production process. This can lead to faster identification and resolution of issues, resulting in fewer defects and a higher overall product quality.

This chapter will explain the design process of a prototype that implements the study carried out so far, the selected technologies, the development process and the results obtained.

5.1. Motivations and requirements

I was tasked by the company to create a tool that can display the data and clearly highlight any anomalies present. With these two main objectives in mind, I have formulated the following requirements.

- Data upload capability: The application should have the ability to upload data in the form of a CSV file, so that operators can easily upload their dataset to the application.
- Data processing capability: The application should have the necessary processing power to perform the anomaly detection techniques and models on the uploaded dataset.
- User interface: The application should have a user-friendly interface that allows operators to easily navigate the application and understand the results.
- Anomaly detection models: The application should include a variety of anomaly detection models, such as statistical methods and neural networks, to provide operators with a range of options for detecting anomalies in their dataset.

- Results visualization: The application should be able to present the results of the anomaly detection in a clear and concise manner, such as highlighting the anomalous samples in a table or graph.

In general, the design of this prototype has been centered on simplicity and flexibility, with the goal of creating a user-friendly and easy-to-use tool that does not impede the workflow and provides the operator with the ability to adjust the settings to align with the current process and obtain desired results

5.2. Technologies

For simplicity and rapid development, as well as to easily integrate all the libraries and models used in data analysis, the Django framework was selected. Django is a Python web framework that encourages rapid development and clean, pragmatic design. It comes with built-in support for working with databases, where the technology used is known as Object-Relational Mapping (ORM) which allows developers to interact with databases using Python code rather than SQL.

5.2.1. Database

In our particular case the data will be saved in SQLite. SQLite is a lightweight, file-based relational database management system (RDBMS) that is commonly used with Django. When using SQLite with Django, the database is stored as a single file on the filesystem of the server. This file contains all the data and the database structure, and the SQLite engine reads and writes to this file to interact with the data.

SQLite is a serverless, zero-configuration database engine, it doesn't require a separate server process or system to run, making it easy to set up and use. It is also portable, meaning that it can run on any platform that supports Python, including Windows, macOS, and Linux.

SQLite is a good choice for small to medium-sized applications, especially those that do not require a lot of concurrent access to the database. It can be useful for development and testing environments, or for small web applications that don't expect to have a lot of traffic.

The combination of these characteristics makes SQLite a viable choice for our project.

5.2.2. Backend and frontend

As mentioned, Django is a web framework and has been used to handle the logic of the application, as well as to return the rendered HTML.

In Django, views handle the logic of the web application and templates handle the presentation of the data. This logic is handle by Python functions called

Views. They handle HTTP requests, process the data, and return HTTP responses. They are responsible for retrieving data from the database, performing calculations, and determining which template to use to display the data. They are defined in the views.py file of the app and can be easily reused across different parts of the application.
Templates are HTML files that define the layout and structure of the pages. They are used to display the data that views retrieve from the database. Templates are separate from the views and can be reused across different views. When a user makes a request to the web application, the request is handled by a view. The view retrieves the necessary data from the database, performs any necessary calculations, and then determines which template to use to display the data. The data is passed to the template, which uses the placeholders to fill in the dynamic content. The final result is an HTML page that is returned to the user's browser.

Once we have the technology chosen for the development of the application, a mockup has been designed with the characteristics that we want to show.

5.3. Design

Prototyping is an essential step in the design process of a web application. It allows designers to test their designs and make changes before the development process begins. The prototyping process starts with the creation of wireframes, which are simple, low-fidelity visual representations of the web application's layout and structure.

5.3.1. Wireframe and user experience

Wireframes are used to map out the basic layout of the pages and to identify any potential issues or problems with the design.

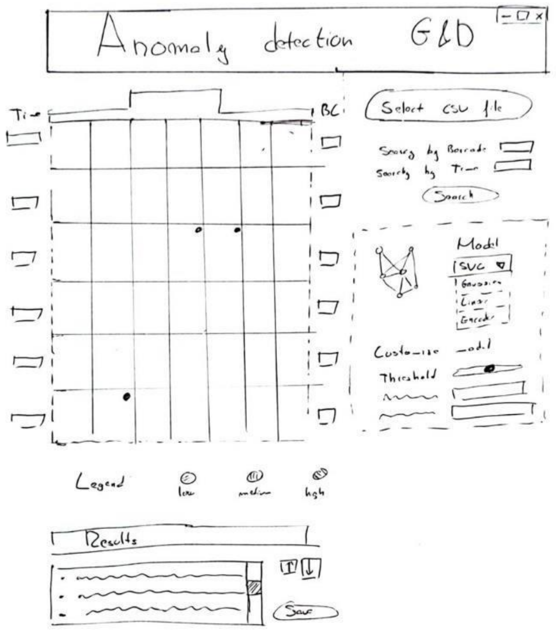


Fig.5.1 Conceptual idea of the application in a wireframe

In this first design I identified the needs and characteristics of the web app. The main purpose is to visually display the collected data, allowing the user to upload files in their original format. A large main button will allow the operator to select one of the CSV files for further analysis.

The server will then process the selected file and display a set of samples on the screen.
Each sample will consist of a color cell that represents the original color measured by the sensor. To identify cells that have been classified as anomalies, a mark such as a dot will be placed in one of the corners of the cell. In this step of the design, it was planned to use three colors for the mark, indicating three different levels of anomaly. A small legend located below the set of cells will identify which level each color belongs to.

The data from the samples with the highest overall score, that is, with the highest number of sensors classified as anomalous, will be displayed in a rectangle at the bottom of the web page. Two buttons next to the rectangle will allow the user to order these results from highest to lowest score. A save button will then allow the user to export these results to a file for better reading and greater adaptability for sharing or processing.

Finally, a box on the right with the name 'Model' is intended to give the user some flexibility when choosing the model with which the data will be processed, as well as some of its parameters. This will allow the user to adjust the results obtained if the default model did not meet expectations or did not work correctly.

5.3.2. Mockup

Once the wireframes are complete, the next step is creating mockups, which are high-fidelity visual representations of the final product.

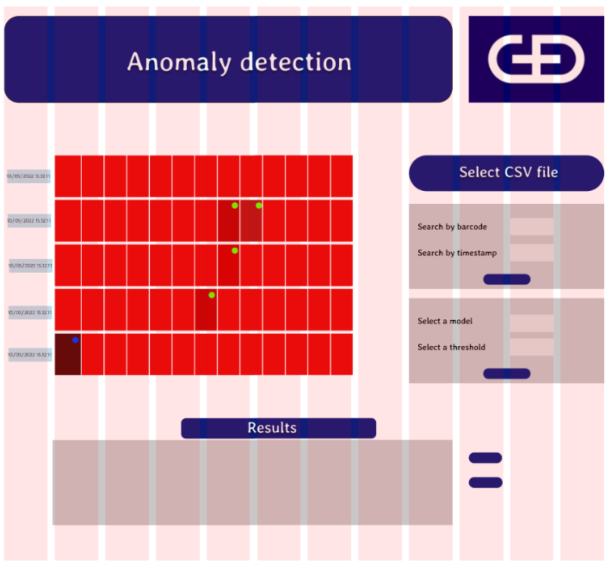


Fig.5.2 Original mockup of the application

For this, the Figma tool has been used since it is free to use and does not require any installation. I have used one of its features which is to split the mockup into 12 columns to get some perspective on the sizes to use when implementing the interface. 12 was used because it is the number of columns that the Bootstrap grid system uses to layout and align content.

In this design, the model part has been simplified, leaving only the threshold parameter. This is because it is difficult to find common parameters between the different models, as well as unnecessary complexity for the end user. With this design, a first representation of what is a series of samples of sheets converted to color has been obtained. You can see that some cells have a different reddish color from the others and the colored dots indicate that an anomaly has been detected in that area.

In the final stage of the design, a simple and visually pleasing color palette was chosen for the web application. The primary color used is blue, which is taken from the company's logo. The color palette that was developed as a result includes the following colors:

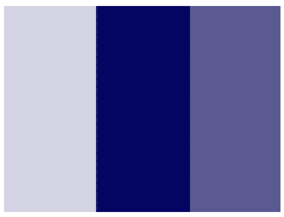


Fig.5.3 Color palette chosen for the application

After completing the design and layout of the web application, and determining the user flow, we are prepared to design the architecture of the data.

5.4. Data diagram

The architectural design of the data is a crucial step in the development of any web application. It involves determining the best way to organize and structure the data to support the application's functional requirements. One way to represent the architectural design of the data is through the use of a UML (Unified Modeling Language) diagram.

In the following image we can see the UML diagram that has been designed for this web application. In it we can see two related entities in a 1:11 relationship. The first one is the "Sample", where it represents each of the rows of our dataset. In it we can see 3 fields:

- Timestamp: time when the measurement was taken, with the following format: "YYYY-MM-DD HH:MM:SS"". E.g., 2021-01-31 19:22:34
- Barcode: value that identifies a section of the foil
- Global score: number of anomalous sensors that have the sample

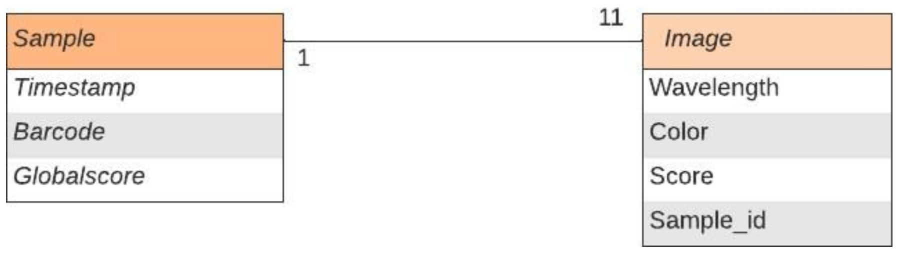


Fig.5.4 UML diagram which shows the Sample and Image entities

The other entity is the 'Image', a visual representation of the measurement of each sensor with the following fields:

- Wavelength: the wavelength measured by the sensor
- Color: The color in hexadecimal format of the wavelength
- Score: indicates if this measure is anomalous
- Sample_id: identifier of the sample entity, thus obtaining a One-to-many [27] relationship

5.5. Development

Once we design the architecture and user flow to meet the requirements is time to develop the web application.

In the next section I'll discuss the key steps in building a Django web application.

5.5.1. Development steps of a Django web application

The first step taken in set up the development environment. In this case is a virtual environment python composed by Django 4.1.3 and django-boostrap-v5.

In this environment we will create a new Django project by running the command "django-admin startproject [projectname]" in the terminal. This will create a new directory with the same specified, which will contain the basic file structure for a Django project.

Within the project directory, we will create a new Django app by running the command "python manage.py startapp [appname]". This will create a new directory with the same name as your app, which will contain the files for your app's. It is the models, views, and templates.

In the settings.py file located in the project directory, we will need to add the app to the list of installed apps. This will ensure that Django recognizes and includes the app when it runs.

Once we have the Django project we will create the models specified in figure X in the models.py file. This model defines the fields and data structure for the app's database.

After creating the models, a migration file is needed. This is achieved by running the command "python manage.py makemigrations [appname]".

Finally, we will need to apply the migration by running the command "python manage.py migrate". This will create the necessary database tables for your app.

At this point Django has internally created the sqlite3 database tables.Next, we will need to create the views in the views.py file. These views handle user requests and define the logic for displaying data on the front-end.

After creating your views, we will create a urls.py file where we define the URL patterns for your app, linking specific URLs to the specific views created.

Finally, we will need to create templates, which defines the HTML structure and layout for your app's pages.

Once we know how to develop a Django web application, we'll see how it can be used to suit our needs and requirements.

5.5.2. Architecture

During the development of the app, I encountered some challenges that had to be addressed using Django. Specifically, Django maps a URL to a view and then returns a template, which is an HTML file with some data inserted. This is not well suited for a single-page app architecture. To overcome this limitation, I incorporated a new element and adapted the architecture.

The diagram below illustrates an example of the proposed architecture:

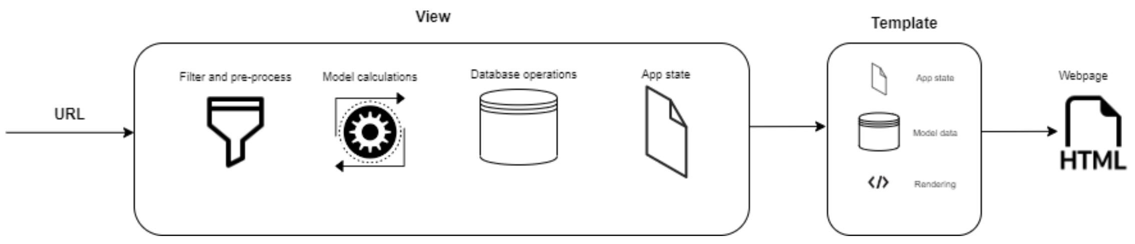


Fig.5.5 Architecture of the web app where the main Django components (URLS, Views and Templates) are showed and how they connect

In the previous image we can see the 3 main elements in the architecture: the URL, the View and the template. As has been commented, each URL is processed by a View. The following logic has been implemented in it:

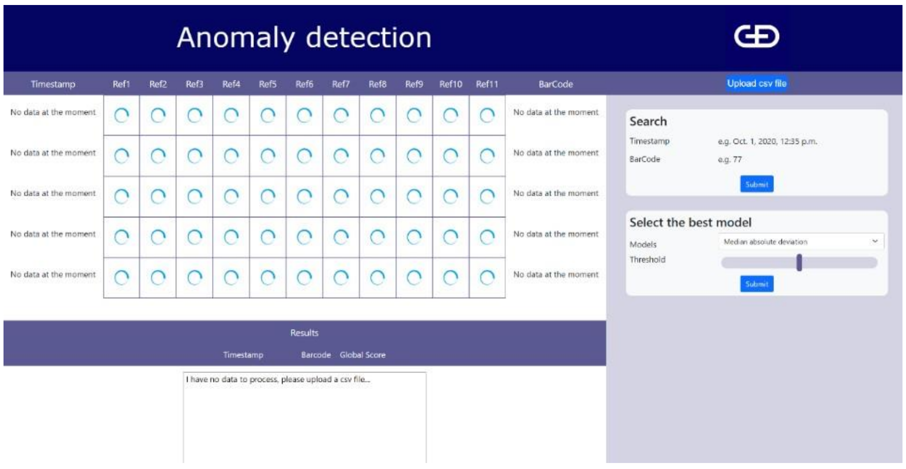
- Filter and pre-process: depending on the view, this is in charge of check that data and request is in the correct format, and perform some data type operations is it needed
- Model calculation: Perform all operations related to the selected model
- Database operations: Once the data has been processed and analyzed, the data can already be stored in the database. These will already contain the classification of anomalies
- App state: This is the new element added to make the website a single page application (SPA) [28]. The idea is to use Django's cache framework [29] feature to keep the current state of the app at any time in an efficient way. Thereby, we will return the same html in the all request with the current state as the new data needed to insert

5.6. Result

The result of the HTML design for the initial view of the application is presented below. The design largely retains its original appearance. A bar, serving as a label for the values of each sample and a main button to upload a CSV file, is visible. Upon pressing the button, a pop-up window appears, allowing the user to navigate to the location of the files. After selecting a file, it will be sent automatically and processed.

The cells, representing areas of the sheet, display an icon indicating the absence of data. The results section has been adapted to display only the most relevant values.

On the right side we can see two clearly differentiated sections, where on the one hand the user will be able to filter by Barcode or timestamp the area of the foil that he wants to see. On the other hand, there is a section where the user can choose the available models and the threshold to use, so that the results are processed differently.



Timestamp	Ref1	Ref2	Ref3	Ref4	Ref5	Ref6	Ref7	Ref8	Ref9	Ref10	Ref11	BarCode
No data at the moment	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	No data at the moment
No data at the moment	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	No data at the moment
No data at the moment	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	No data at the moment
No data at the moment	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	No data at the moment
No data at the moment	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌	No data at the moment

Timestamp e.g. Oct. 1, 2020, 12:35 p.m.
BarCode e.g. 77

Models Median absolute deviation
Threshold

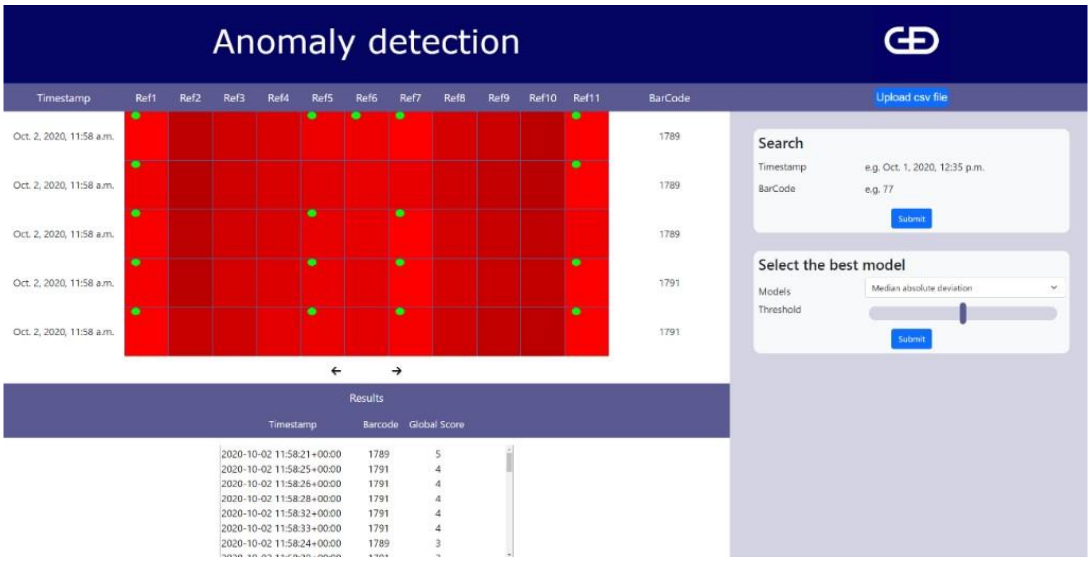
Timestamp	Barcode	Global Score

I have no data to process, please upload a csv file...

Fig.5.6 First view of the app

5.7. Testing

After the user has selected the file, it will be processed. After some time, the first 5 samples of the processed file are returned, with each sensor value represented by a color and labeled by a circle if it has been classified as abnormal. Also, for each row of cells, we can see at what time the sample was taken and the barcode that was assigned.


Anomaly detection

Upload csv file

Timestamp	Ref1	Ref2	Ref3	Ref4	Ref5	Ref6	Ref7	Ref8	Ref9	Ref10	Ref11	BarCode
Oct. 2, 2020, 11:58 a.m.	●				●	●	●				●	1789
Oct. 2, 2020, 11:58 a.m.	●										●	1789
Oct. 2, 2020, 11:58 a.m.	●				●		●					1789
Oct. 2, 2020, 11:58 a.m.	●				●		●				●	1791
Oct. 2, 2020, 11:58 a.m.	●				●		●				●	1791

Results

Timestamp	Barcode	Global Score
2020-10-02 11:58:21+00:00	1789	5
2020-10-02 11:58:25+00:00	1791	4
2020-10-02 11:58:26+00:00	1791	4
2020-10-02 11:58:28+00:00	1791	4
2020-10-02 11:58:32+00:00	1791	4
2020-10-02 11:58:33+00:00	1791	4
2020-10-02 11:58:24+00:00	1789	3

Search

Timestamp: e.g. Oct. 1, 2020, 12:35 p.m.

BarCode: e.g. 77

Submit

Select the best model

Models: Median absolute deviation

Threshold

Submit

Fig.5.7 Example of the application once it has processed the data.

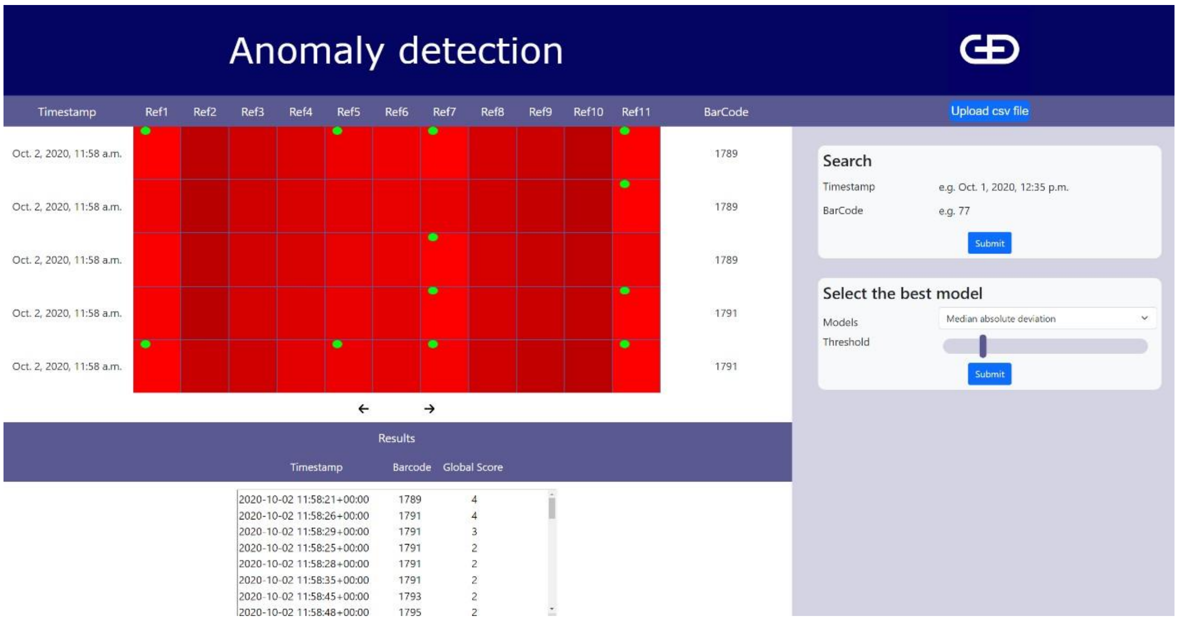
The 5 samples represented in color are equivalent to capturing 5 seconds of the foil, that is equivalent to about 6 and a half meters of foil.

During the data analysis (2.3) phase, we observed a significant and rapid variation between sensors and samples, which is reflected in the different shades of red shown in the image. However, it's important to note that not every variation in color is considered an anomaly

In the image above, there are also two arrows at the bottom of the cell set, allowing the user to display the next or previous 5 samples, enabling them to observe the visual behavior of the foil over time

On the other hand, the samples ordered according to their overall score are also returned, in this way the user will be able to see when the samples with the greatest errors have been produced. You could then filter by timestamps to see what that sample would look like-

The following image displays the same selected file as in the previous screenshot, but with a different threshold applied. It is evident that some sensors are now not categorized as anomalous. The results section shows that the sample with the highest errors now has 4 abnormal sensors instead of 5.



Anomaly detection

Timestamp	Ref1	Ref2	Ref3	Ref4	Ref5	Ref6	Ref7	Ref8	Ref9	Ref10	Ref11	BarCode
Oct. 2, 2020, 11:58 a.m.	●				●		●				●	1789
Oct. 2, 2020, 11:58 a.m.											●	1789
Oct. 2, 2020, 11:58 a.m.							●					1789
Oct. 2, 2020, 11:58 a.m.							●				●	1791
Oct. 2, 2020, 11:58 a.m.	●				●		●				●	1791

Results

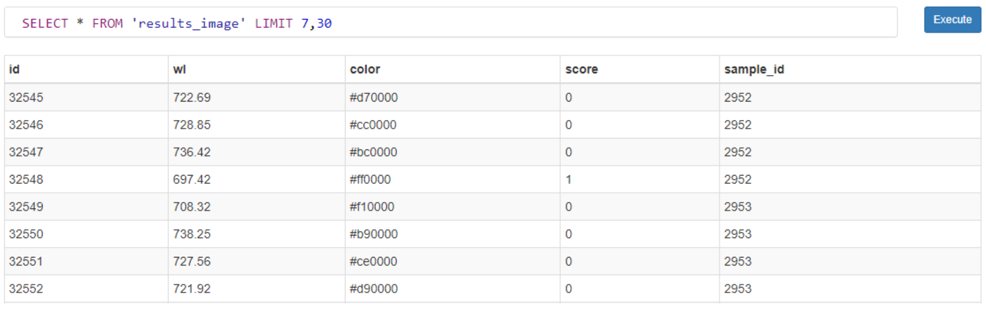
Timestamp	Barcode	Global Score
2020-10-02 11:58:21+00:00	1789	4
2020-10-02 11:58:26+00:00	1791	4
2020-10-02 11:58:29+00:00	1791	3
2020-10-02 11:58:25+00:00	1791	2
2020-10-02 11:58:28+00:00	1791	2
2020-10-02 11:58:35+00:00	1791	2
2020-10-02 11:58:45+00:00	1793	2
2020-10-02 11:58:48+00:00	1795	2

Search
Timestamp e.g. Oct. 1, 2020, 12:35 p.m.
BarCode e.g. 77

Select the best model
Models: Median absolute deviation
Threshold

Fig.5.8 Example of the application once it has processed the same data but using different thresholds.

To see how the sensors are stored we can make an SQL query and get the data. In the following image, 8 data entries have been selected that are equivalent to a measurement from a sensor. As can be seen, the first 4 entries correspond to sensors in the sample with identifier 2952, and the other entries are the values of the sensors in the following sample. It can be seen how each wavelength has its color equivalent in hexadecimal format.



```
SELECT * FROM 'results_image' LIMIT 7,30
```

id	wl	color	score	sample_id
32545	722.69	#d70000	0	2952
32546	728.85	#cc0000	0	2952
32547	736.42	#bc0000	0	2952
32548	697.42	#ff0000	1	2952
32549	708.32	#f10000	0	2953
32550	738.25	#b90000	0	2953
32551	727.56	#ce0000	0	2953
32552	721.92	#d90000	0	2953

Fig.5.9 Samples of sensor data stored in the database.

CHAPTER 6. CONCLUSIONS

In this project, we aimed to address a real-world issue in the production of colored metallic foil. Our goal was to develop a prototype tool that could detect instances when the color of the foil deviates from the desired value, thus identifying parts that need to be removed from the production process. To achieve this, we approached the problem as an unsupervised anomaly detection problem and established specific objectives and requirements for the project.

We began by studying the data source and its characteristics, including any limitations. We then performed a cleaning and preparation process to ensure that the data was suitable for analysis. Furthermore, we studied various field such as descriptive statistics, distribution and correlation, which allowed us to characterize the system.

In order to have a better understanding of the problem, we reviewed the literature on anomalies and their different types. We also examined the various techniques that have been used to detect anomalies in similar systems.

We then applied different techniques and evaluated the advantages and disadvantages of each technique and compared the results obtained. These techniques are based on different principles for identifying anomalous observations in the data.

One of the principles we used in this project was that anomalies are observations that deviate significantly from the expected pattern or model of the data. To identify these anomalies, we assumed that the data was generated by a Gaussian distribution and applied techniques such as the 3-sigma rule, the Box Plot rule, and the Gaussian mixture model. These techniques allowed us to identify different ratios of anomalies depending on the sensor, potentially indicating that certain areas of the system are more prone to errors or some sensor wear. Additionally, we characterized each sensor measurement into 4 distribution groups, with the anomalies falling into one of these groups.

Another principle we used was based on the idea that data instances tend to occur in dense neighborhoods, while anomalies occur far from their nearest neighbors. To apply this principle, we used the Local Outlier Factor (LOF) algorithm. This technique helped us to see how the density of the data affected the detection of anomalies, and how different parameters affected the results. By analyzing the results, we were able to establish comparable thresholds to those used in other techniques. However, it's important to note that certain values that fall outside of these thresholds may not necessarily be considered anomalous, as they could have a high concentration depending on the chosen parameters

The third principle we used was based on the idea that data can be embedded in a lower dimensional subspace and have, in which normal and anomaly instances appear significantly different. Also, on the fact that real data attributes tend to be highly correlated. To apply these principles, we used techniques such as

Autoencoders and Vector Autoregression (VAR) model. By using these techniques, we were able to identify instances that deviated significantly from the norm by comparing the reconstruction error of a single sample in a lower dimension or the prediction of a linear combination of previous sample values and other time series data.

Additionally, we were able to see how the temporal and spatial correlation affected the detection of anomalies and how different parameters affected the results.

Overall, these principles helped us to understand the different ways in which anomalies can be detected in the data and the advantages and disadvantages of each approach.

We designed and developed a prototype web app tool for detecting errors in the production process of a colored metallic foil. The tool utilizes a variety of techniques, each based on different principles of anomaly detection, to identify instances that deviate significantly from the expected pattern or model of the data. The design and architecture of the tool were carefully considered to ensure ease of use for the end user. The prototype web app allows for the uploading of a CSV file, and allows the user to select from a variety of techniques for detecting anomalies. The app then returns a visual representation of the sensor measurements, highlighting any instances that have been identified as anomalous.

The benefits of using this tool for the company include improved efficiency and quality in the production process, as well as cost savings by reducing the need for manual error detection. Additionally, the tool can provide valuable insights into the underlying causes of errors, allowing for targeted improvements to the system. Overall, the prototype web app tool is a valuable addition to the company's quality control process, helping to ensure that only high-quality products are delivered to customers.

In conclusion, this project successfully achieved all its set objectives. We delved into the field of anomaly detection and applied various techniques, gaining valuable experience in data preparation and processing. The study of anomalies and the development of a web-based application, equipped with the expected functionalities, added further depth to our understanding of the topic.

6.1. Future lines

However, there are still many opportunities for future work. One area that could be explored is the use of more advanced techniques, such robust subspace recovery layer (RSR layer), a neural network that extract the underlying subspace from a latent representation of the given data and removes outliers that lie away from this subspace. It is used within an autoencoder and have demonstrated state-of-the-art precision and recall.

Another area that could be improved is the web framework. Currently, the web app is built using Django, which is a lightweight framework for building web applications. However, other frameworks such as React or Angular, which are more appropriate for single-page apps and reactive web, could be considered for future development. This would allow for more flexibility and scalability in the web app.

In addition, the web app could be dockerized to make it more flexible in different hardware environments. This would enable the web app to be easily deployed in various environments without the need for extensive configuration and setup.

Additional enhancements can be made to streamline the way we handle and store the samples. One potential solution is to implement a system where a batch of samples is processed and returned to the user in real-time, rather than requiring them to wait for the entire dataset to be processed and saved to the database.

6.2. Personal assessment

In this study, I was able to gain a comprehensive understanding of the key steps involved in a data analysis project and tailor it to the specific context. I delved into the field of anomaly detection, gaining an understanding of the different types and characteristics of anomalies. Through the examination of various techniques, methods, and algorithms, I gained experience in identifying and analyzing anomalies.

This field is highly relevant and critical in many industries. For example, is essential in many real-world applications such as network intrusion detection, fraud detection, quality control and fault diagnosis. Also, anomaly detection techniques can lead to increased efficiency in various industries by detecting problems early on and preventing major issues. That is why the knowledge acquired in this study can add value to me in a wide range of fields and problems.

Although it was not possible to measure the results obtained in this project, a balanced approach was taken between complexity and practicality. Instead of focusing solely on the state-of-the-art methods, simpler models were studied and their results were compared.

Additionally, this project also reinforced other skills such as application development. By building a prototype, I learned a new web framework, which will give me more flexibility in future projects.

In conclusion, this project allowed me to apply the knowledge acquired during my degree and learn new skills, while solving a real problem faced by a company.

BIBLIOGRAPHY

[1] K.Huang Xiemin (2020,Jun. 30) *Google Capstone Project: How Can Bellabeat, A Wellness Technology Company Play It Smart?* [Online]. Available: https://medium.com/analytics-vidhya/this-case-study-is-for-google-data-analytics-gda-capstone-project-course-54047cccf7cb

[2] Pandas (2023, Jan. 19) *Pandas documentation* [Online]. Available: https://pandas.pydata.org/docs/index.html

[3] Python (2023, Jan. 31) *Pickle — Python object serialization* [Online]. Available: https://docs.python.org/3/library/pickle.html

[4] F. Y. Edgeworth. "On Discordant Observations", *Philosophical Magazine*, Vol.23, pp. 364–375, 1887.

[5] F.J Anscombe, "Rejection of Outliers", Technometrics, Vol.2, pp. 123-146, 1960

[6] W.A Shewhart, *Economic control of quality of manufactured product*. North Eastern States Libraries, 1923

[7] H.E Solberg, A. Lahti. "Detection of outliers in reference distributions: performance of Horn's algorithm", *Clinical Chemistry*, Vol. 51, pp. 2326–2332, Dec 2005

[8] S.E Guttormson, R.J Marks. "Elliptical novelty grouping for on-line short-turn detection of excited running rotors", *IEEE Transaction on Energy Conversion*, Vol.14, Mar 1999

[9] Seaborn (Last visit 2022, Oct. 24) *seaborn: statistical data visualization* [Online] Available: https://seaborn.pydata.org/index.html

[10] Wikipedia (2023, Jan. 26*) Maximum likelihood estimation* [Online]. Available: https://en.wikipedia.org/wiki/Maximum_likelihood_estimation#:~:text=In%20statistics%2C%20maximum%20likelihood%20estimation,observed%20data%20is%20most%20probable.

[11] Wikipedia (2023, Jan. 25) *Likelihood function* [Online]. Available: https://en.wikipedia.org/wiki/Likelihood_function

[12] Wikipedia (2022, Nov. 24) *Covariance Matrix* [Online]. Available: https://en.wikipedia.org/wiki/Covariance_matrix

[13] S.J Hickinbotham and J.Austin, "Neural networks for novelty detection in airframe strain data", Dept. Comput. Sci., University of York, UK, 2000

[14] Tarassenko, Hann, Young. “Integrated monitoring and analysis for early warning of patient deterioration”, British Journal of Anaesthesia, V.97, pp. 64-68, Jul 2006

[15] K.Yamanishi, J.Takeuchi. “Discovering outlier filtering rules from unlabeled data: combining a supervised learner with an unsupervised learner” in *Conf. Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, San Francisco, CA, 2001, pp. 389–394

[16] R.G. Brereton, (2021 Apr. 14) *Multivariate classification models* [Online]. Available: https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/full/10.1002/cem.3332

[17] scikit-learn (Last visit 2022, Nov. 10) *scikit-learn-Machine Learning in Python* [Online]. Available: https://scikit-learn.org/stable/

[18] Kaggle (2018) *Iris Flower Dataset* [Online]. Available: https://www.kaggle.com/datasets/arshid/iris-flower-dataset

[19] ScienceDirect (Last visit 2022, Dec. 6) *Bayesian Information Criterion* [Online] Available: https://www.sciencedirect.com/topics/social-sciences/bayesian-information-criterion

[20] S.Sugidamayatno, D.Lelono. “Outlier Detection Credit Card Transactions Using Local Outlier Factor Algorithm (LOF)”, *IJCCS (Indonesian Journal of Computing and Cybernetics Systems*, Vol.13, pp. 409-419, Oct. 2019

[21] S.Hasani, S.Krrabaj. “Survey and Proposal of an Adaptive Anomaly Detection Algorithm for Periodic Data Streams”, *Journal of Computer and Communications*, Vol.7 Aug. 2019

[22] Wikipedia (2023, Jan. 30) *Gradient descent* [Online]. Available: https://en.wikipedia.org/wiki/Gradient_descent

[23] J. Brownlee (2019, Jan. 9) *Gentle Introduction to the Rectified Linear Unit (ReLU)* [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[24] S.Barreto (2022, Nov. 4*) What is Fine-tuning in Neural Networks?* [Online]. Availale: https://www.baeldung.com/cs/fine-tuning-nn

[25] S.Prabhakaran (2019, Nov. 2) *Augmented Dickey Fuller Test (ADF Test)* [Online]. Available: https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/

[26] R.Bevans (2020, Mar. 26) *Akaike Information Criterion* | *When & How to Use It* [Online]. Available: https://www.scribbr.com/statistics/akaike-information-criterion/

[27] Baeldung (2022, May. 29*) Hibernate One to Many Annotation Tutorial* [Online]. Available: https://www.baeldung.com/hibernate-one-to-many

[28] Wikipedia (2023, Jan. 16) *Single-page application* [Online]. Available: https://es.wikipedia.org/wiki/Single-page_application

[29] Django documentation (Last visit 2022, Nov. 14) *Django's cache framework* [Online]. Available: https://docs.djangoproject.com/en/4.1/topics/cache/

[30] Charu C. Aggarwal, *Outlier Analysis* ,"An Introduction to Outlier Analysis", Chap. 1, pp. 1-18, "Probabilistic and Statistical Models for Outlier Detection", Chap. 2, pp. 36-61, "Linear Models for Outlier Detection", Chap. 3, pp- 65-107, "Proximity-Based Outlier Detection", Chap. 4, pp. 111-137, "Time Series and Multidimensional Streaming Outlier Detection", Chap. 9, pp. 276-280 (2017)