Acknowledgements

I would like to give special thanks to my tutors, Dr. Jordi Cuadros and Dr. Albert Fornells, who have guided me through the process of writing this memoir and have always pushed me to deliver at my highest standard. Moreover, I also want to acknowledge my family for helping me push through when the workload seemed too high.

Abstract

Given the rising trends in industrial environments to digitalise data collection, allowing for a fast retrieval of data at high volumes, classic quality control methods to detect out-of-control (OOC) processes such as statistical process control (SPC) have seen increased limitations. Due to that this project, tests the viability of using alternative methods to monitor processes within the popular field of machine learning (ML).

To achieve this, code is developed to create a synthetic dataset which simulates observations in an industrial process, which can be in-control or out-of-control. This data is manipulated so observations from different out-of-control situations are contextualised and differentiated from observations where the process is in-control. Then two ML algorithms, that can tackle the problem of identifying out-of-control processes are chosen, and ML models using these algorithms are created, validated, and tuned. Finally, SPC methods are automated. Both, SPC methods and ML models, specifically isolation forest and support vector machine-based models, are used to predict whether a process is in-control or not within the same benchmark.

The results obtained show that the ML models can be adapted to identify OOC processes and are ideal substitutes to SPC methods, as both methods have similar measures of correct predictions with F1-score of 0.71 and 0.72 using SPC method variations and 0.79 and 0.80 using isolation forest based mode and SVM based model respectively. A few differences between the methods can be observed, however. More precisely ML models have a better balance between identifying OOC instances correctly and not labelling in-control instances as OOC. On the other hand, SPC methods are slightly better at detecting OOC processes but at the cost of wrongly labelling more in-control situations as OOC. When observing the predictions per type of OOC pattern, it is possible to see that SPC methods have an improved ability to detect trends and the support vector machine-based model generally has lower number of correct identifications of OOC patterns than the other methods.

Resumen

Dada la tendencia en el entorno industrial de digitalizar la recogida de datos, lo que permite una rápida recuperación de estos en grandes volúmenes, los métodos clásicos de control de calidad para detectar procesos fuera de control, como el control estadístico de procesos, han visto incrementadas sus limitaciones. Debido a ello, este proyecto pone a prueba la viabilidad de utilizar métodos alternativos para monitorizar procesos dentro del campo del aprendizaje automático.

Para ello, se desarrolla un código para crear un conjunto de datos sintéticos que simula las observaciones de un proceso industrial, que puede estar en control o fuera de control. Estos datos se manipulan para que las observaciones de las diferentes situaciones fuera de control se contextualicen y se diferencien de las observaciones en las que el proceso está bajo control. A continuación, se eligen dos algoritmos de ML que puedan abordar el problema de la identificación de los procesos fuera de control, y se crean, validan y ajustan los modelos de ML que utilizan estos algoritmos. Por último, se automatizan los métodos de control estadístico de procesos. Tanto los métodos de control estadístico de procesos como los modelos de aprendizaje automático, concretamente los modelos basados en bosques de aislamiento y máquinas de vectores de soporte se utilizan para predecir si un proceso está en control o no dentro del mismo punto de referencia.

Los resultados obtenidos muestran que los modelos aprendizaje automático pueden adaptarse para identificar procesos fuera de control y son sustitutos ideales de los métodos de control estadístico de procesos, ya que ambos métodos tienen medidas similares de predicciones correctas con valores de F1-score de 0.71 y 0.72 usando variaciones de los métodos de control estadístico de procesos y 0.79 y 0.80 usando el modelo basado en bosque de aislamiento y máquinas de vectores de soporte respectivamente. Sin embargo, se observan algunas diferencias entre los métodos. En concreto, los modelos aprendizaje automático tienen un mejor equilibrio entre la identificación correcta de los casos fuera de control y la no etiquetación de los casos en control como fuera de control. Por otro lado, los métodos de control estadístico de procesos son ligeramente mejores en la detección de procesos fuera de control, pero a costa de etiquetar erróneamente más situaciones en control como fuera de control. Al observar las predicciones por tipo de patrón fuera de control, se puede ver que los métodos de control estadístico de proceso son más capaces de detectar tendencias y que el modelo de máquina de vectores de soporte generalmente identifica un número menor de observaciones de patrones fuera de control que los otros métodos.

Resum

Donades les tendències creixents dels entorns industrials per digitalitzar la recollida de dades, que permeten la recuperació ràpida de dades en grans volums, els mètodes clàssics de control de qualitat per detectar processos fora de control com el control estadístic de processos han patit de augments en les seves limitacions. Per això, aquest projecte prova la viabilitat d'utilitzar mètodes alternatius per supervisar processos dins del popular camp de l'aprenentatge automàtic.

Per aconseguir-ho, es desenvolupen codis per crear un conjunt de dades sintètiques que simulin observacions en un procés industrial, que pot estar controlat o fora de control. Aquestes dades es manipulen de manera que les observacions de diferents situacions fora de control es contextualitzen i es diferencien de les observacions on el procés està controlat. A continuació, es trien dos algorismes de l'aprenentatge automàtic, que poden abordar el problema d'identificar processos fora de control, es creen, es validen i s'ajusten models de l'aprenentatge automàtic que utilitzen aquests algorismes. Finalment, els mètodes control estadístic de processos s' automatitzen. Tant els mètodes control estadístic de processos com els models l'aprenentatge automàtic, específicament els models basats en el algoritme del bosc d'aïllament i en els classificadors de vectors de suport, s'utilitzen per predir si un procés està controlat o no dins del mateix punt de referència.

Els resultats obtinguts mostren que els models l'aprenentatge automàtic es poden adaptar per identificar processos fora de control i són substituts ideals dels mètodes de control estadístic de processos, ja que ambdós mètodes tenen mesures similars de prediccions correctes amb valors de F1-score de 0.71 i 0.72 utilitzant variacions dels mètodes de control estadístic de processos i 0.79 i 0.80 utilitzant els models basats en bosc d'aïllament en els classificadors de vectors de suport respectivament. Tot i així, es poden observar algunes diferències entre els mètodes. Més precisament, els models l'aprenentatge automàtic tenen un millor equilibri entre la identificació correcta de les instàncies fora de control i no etiquetar les instàncies en control com a fora de control. D'altra banda, els mètodes control estadístic de processos són lleugerament millors per detectar processos fora de control, però al preu d'etiquetar incorrectament més situacions en control com a fora de control. Quan s'observen les prediccions per tipus de patró fora de control, és possible veure que els mètodes control estadístic de processos són més capaços de detectar tendències i el model basat en classificadors de vectors de suport generalment identifica un numero menor de observacions dels patrons fora de control que els altres mètodes.

Table of Contents

List of Figures

List of Tables

Glossary of Acronyms

AD	Anomaly Detection
CL	Centre Line
FN	False Negative
FP	False Positive
IC	In Control
LCL	Lower Control Line
LS	Least Squares
ML	Machine Learning
OOC	Out of Control
QC	Quality Control
QMS	Quality Management Systems
RBF	Radial Based Function
SPC	Statistical Process Control
SVM	Support Vector Machines
TN	True Negative
TP	True Positive
UCL	Upper Control Line
WE	Western Electric

1 Introduction

In the 21st century, businesses have evolved by increasingly digitalising and integrating new technologies to their processes. This movement is known as industry 4.0 and it has provided new opportunities to manage these processes. One such opportunity is the use of machine learning models as a substitute for current quality control tools, to help monitor a process and avoid production of faulty products (American Society for Quality, n.d.).

Quality control is a component of quality management which helps businesses fulfil quality requirements or specifications allowing them to pursue sustained success in a competitive market (International Organization for Standardization, 2015). Currently, methods such as statistical process control (SPC) are used. SPC allows to track and control a process ensuring it is operating efficiently and products are specification-conforming. It provides tools that evaluate the cause of the variability of the characteristics that define a product, enabling to detect when it is associated to the process being out-of-control (OOC), known as an assignable variability (NIST/SEMATECH, 2012). To discriminate when a process is OOC it uses a set of rules that are based on probability or heuristic observations of patterns which have been developed by companies such as Western Electric, or quality experts such as Lloyd S. Nelson (Western Electric Company, 1956). It is important to state that classically SPC is conducted at a low pace, by manually obtaining data, using acceptance sampling, and manually updating charts and tools, therefore the data analysis is performed offline (NIST/SEMATECH, 2012). This is because it is a method that was developed before the availability of automatic data obtention, which would have made inspecting all data related to a product too expensive and too long (NIST/SEMATECH, 2012).

Digitalization and data storing for industrial processes with the help of sensors and databases, which characterize the industry 4.0, have allowed companies to retrieve high volumes of data at high velocities, known as big data (Oracle, 2022).This data can be leveraged to track the status of a process almost in real time. However, due to the pace at which SPC is conducted, the lag between the analysed data, which is obtained manually, and the live data, which is automatically retrieved, can cause a delay in identifying when a process is OOC. Moreover, given the probabilistic and heuristic nature of identifying OOC statuses using SPC, the more data is analysed the higher is the chance to label a process as OOC when in reality it is in control (IC) (NIST/SEMATECH, 2012).

One solution to adapt SPC to the high speed of data collection would be to automate it by creating software able to apply SPC methods to a dataset. However, thanks to the availability

of big data, tools such as machine learning (ML) can be taken advantage of. This is because ML models rely on large amounts of data to learn its structure and find patterns. By learning from the structure of the data, machine learning models are able to model the characteristics of normal data and can discriminate anomalies, or directly learn the characteristics of anomalies to identify them. This is possible given the features of anomalies deviate from the ones of the majority of the data. This practice is called anomaly detection and it could be used to identify when a process is OOC. Some of the most common algorithms used for anomaly detection are decision trees and ensemble methods, deep learning, and kernel-based methods (Polzer, 2021).

Given the limitations of SPC and the possibility of leveraging machine learning models using new anomaly detection algorithms, the main objectives of this project are:

1) Create a representative synthetic dataset of the common assignable variabilities which are present in industrial environments
2) Choose a set of anomaly detection algorithms to then create and train machine learning models that can be used to identify assignable variabilities
3) Apply the automated SPC methods and the ML models to identify OOC instances within a test dataset
4) Evaluate both ML models and the automated approach of SPC and compare their performance to detect OOC instances

The work is organized as follows. In the second chapter legal aspects of the project are detailed and considered, to ensure the correct legal development of the project. The third chapter will detail the reach of the project, including its deliverables and their specifications. Moreover, it will explore the expected use of economic and time resources. The fourth chapter will include a synthesis of the relevant theoretical background of quality control, focusing on the use of statistical process control, as well as machine learning. This is done in order to help fully understand the concepts exposed in the project's development. The fifth chapter will include the technical memoir of the project which covers methodology of how (1) develop the synthetic dataset, (2) perform data pre-processing, (3) create and validate automated SPC methods, and (4) create and use different anomaly detection machine learning models. Moreover, this chapter will express the results obtained. The sixth chapter will detail the final financial evaluation of the project. Finally, the seventh chapter will present the conclusions of the work and how they tie back with the objectives presented in this introduction.

2 Legal Framework

Within this chapter a study of legislation applicable and relevant to the project will be detailed.

First off, given no personal data is used for this project the organic law 3/2018 of December 5, for the protection of personal data and guarantee of digital right cannot be applied (Jefatura del Estado de España, 2018).

Secondly, according to the regulation on ownership, protection, and exploitation of intellectual and industrial property rights redacted by the research committee of IQS, the ownership of the results delivered by a project correspond to the individuals that have contributed intellectually towards the obtention of such results (Comissió de Recerca d'IQS, 2021). Therefore, all code, data, models, and the memoir related to the development of this project are owned by the author of the memoir, and in the case of the data the ownership is also attributed to the mentors of the memoir. It is important to detail however, that given the project was developed while working with IQS, and more specifically for the ASISTEMBE research group, they must be mentioned alongside the results.

This regulation complies and is based upon the law of intellectual right, Royal Decree-Law 12/2017 currently in place in Spain, which also states that the owner has the right of being recognized as the author and has the power to decide how the results are distributed and the conditions upon which they are. Thus, it has been decided that the results will be shared with a specific general public license (GNU-GPL) for each result, found in the code repository. The license attributes the following legal characteristics each result:

1) Copyleft license: Allows for the use of the software and its posterior versions, by any user, and avoids posterior appropriation of the software and related documents
2) Free software license: Attributes the software as being free, thus respecting the user's liberties for use, distribution, modification, and improvements to be applied to the software and related documents (Free Software Foundation Inc., 2022)

3 Project Viability Analysis

In this chapter the reach and specifications of the results of the project will be detailed, as well as the expected economic and time resources used to deliver the results.

The project will deliver as results:

- A GitHub code repository
- A synthetic dataset (DS1)
- The dataset from which prediction results are obtained (DS2)
- An automated approach of SPC methods
- Two trained ML models
- The predictions made by each model and the SPC method for the test dataset
- A memoir of the project

All datasets will be in compressed format, and they will be included in the GitHub repository of the project. DS1 will be made of series of observations composed of IC observations and OOC observations that derive from 7 different patterns. IC data will be considered to have a normal distribution sampling with constant expected value. The OOC patterns will be presented in a singular way; thus, no more than one OOC pattern is present at the same time, nor more than one pattern is present in the same series of observations. Information about the class of the observation, the point of start of the OOC, as well as the type of pattern will also be included.

DS2 is the resulting datasets from the data pre-processing conducted. These are the datasets used as a benchmark to compare the use of SPC and ML models to detect OOC situations in industrial environments. It is important to state that no analysis will be included on what window sizes, nor which features, both being part of the techniques applied during data pre-processing, will work best to make predictions with the ML models.

Furthermore, a way to automatically implement SPC methods, such as control chart creation, limits calculation and rules implementation will be programmed. Two different set of rules will be used for the SPC analysis of the datasets.

Finally, the cross-validation and hyperparameter tuning routines for each model as well as two trained ML models which use different algorithms will be returned as object files.

The software to be used in the project is:

- R version 4.1.3 (2022-03-10) (Copyright © 2022 The R Foundation for Statistical Computing)
- RStudio as an Integrated Development Environment ("Prairie Trillium" Release (9f796939, 2022-02-16) for Windows)
- Python 3.9.12 (© Copyright 2001-2022, Python Software Foundation)
- Anaconda distribution (© 2022 Anaconda Inc. All rights reserved.)

In *figure 1*, a Gantt diagram, with the steps followed to carry out the project in chronological order, as well as the expected dedication of time for each task, is displayed. The work done by the project owner results in a total of 325 hours, while the work done by each mentor results in a total of 40 hours.

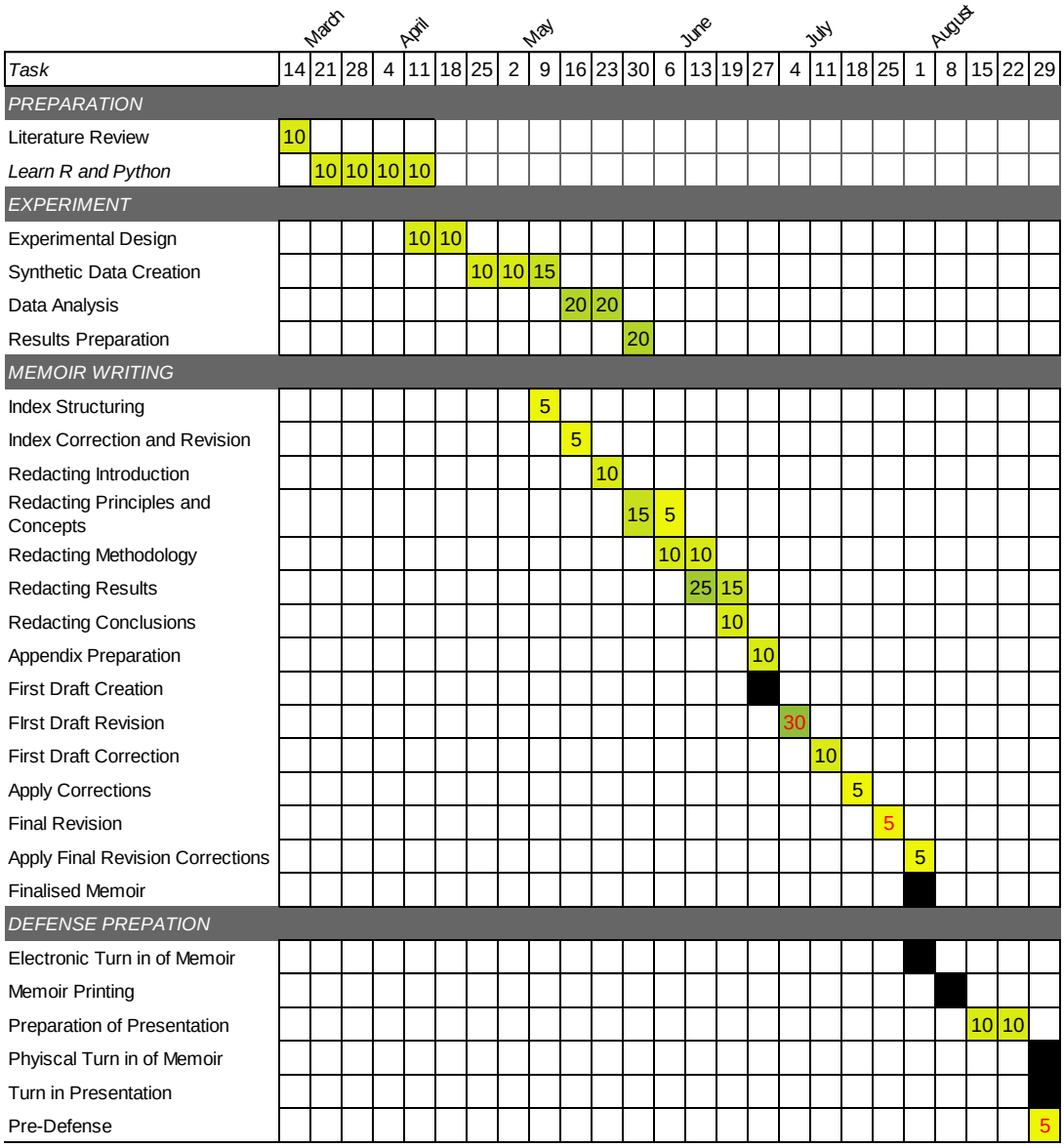


Figure 1. Gantt diagram for the project. Source: personal collection

The estimated cost of the project is 13.000,00 euros. This is accounting mainly for human resources, one junior engineer as project owner and two senior engineers as mentors, their indirect costs such as, facilities maintenance and administration fees, hardware depreciation for one laptop, and software costs.

4 Principles and Concepts

Within this chapter the main concepts that ground the project are exposed. The chapter starts by describing quality control (QC) and is followed by the explanation of machine learning (ML) and anomaly detection concepts.

4.1 Quality Control

To explore quality control as related to this project, apart from basic background, the sub-chapter will expose what SPC is and how it is implemented. Followed by the description of control charts as an SPC tool and finalized with the detailed information of how to build a specific set of control charts, known as I-MR charts.

4.1.1 Basic Background

Quality has always been an integral part of doing business. This is because one of the keys for a successful company is to have a desired product that has value and fulfils a client's necessities and expectations. A product of high value, is one that is desired, is produced at the lowest cost possible, has an acceptable degree of quality, and can satisfy demand. In order to produce products at high volumes, which maintain an acceptable degree of quality, a process of quality control must be implemented, hereby making quality control a key part of having a profitable business (Cuadros, 2022) (NIST/SEMATECH, 2012).

Another way of framing the scope of quality control is to avoid manufacturing products with defects. This can be done through the use of techniques such as statistical process control (NIST/SEMATECH, 2012).

4.1.2 Statistical Process Control

The practice of integrating statistics to process control, started in the 1920s thanks to Dr. Shewhart and the development of sampling theory (NIST/SEMATECH, 2012). Dr. Shewhart took advantage of statistical concepts and practices to develop a set of tools that were able to detect unconformities of products based on a historical knowledge of what a conforming products characteristic should be like (NIST/SEMATECH, 2012).

Statistical process control starts from the basis that variability is inherent in any manufacturing process, however, there is variability that can be associated to chance and variability that is specially caused, for instance by a malfunction of a machine in the manufacturing process. In fact, a process that only has variability associated with chance (common causes) is said to be stable or in control, whereas, whenever the cause for variability is considered assignable

the process is said to be out-of-control (Montgomery, 2013). Most of the time processes behave under control, however, this stability cannot last forever, thus eventually an assignable cause of variation will occur. The goal of SPC is to detect these assignable variations as quick as possible so the cause of the deviation can be investigated, and the process can be recalibrated before many units that do not fit standards or specifications are manufactured.

The classical implementation of SPC, follows the steps displayed in *figure 2*.

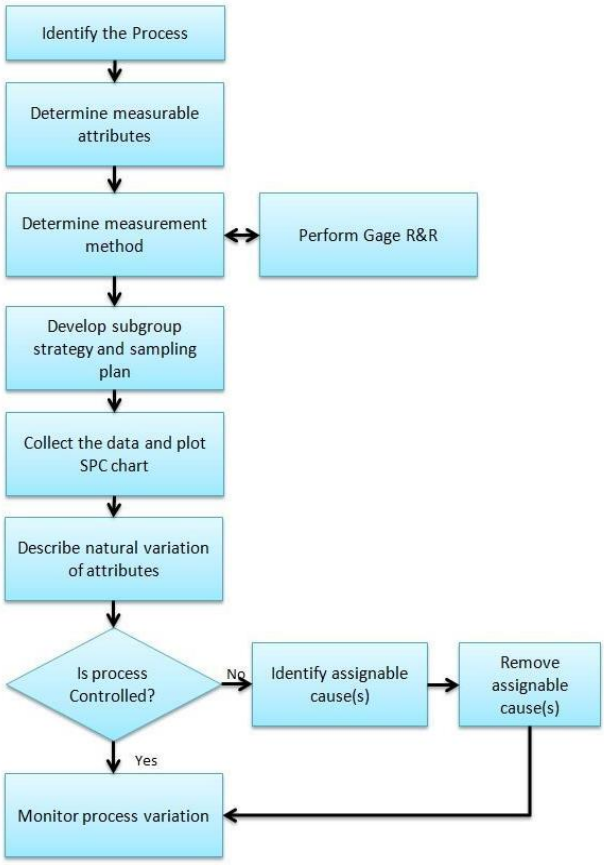


Figure 2. Diagram of SPC implementation. Source: Hessing, six sigma study guide, n.d.

As seen, the first step to apply SPC is to identify the process. This means to find the key process that affects the quality of a product, in regard to the customers' needs for it. For example, the hardness of a screen impacts the performance of a phone, given adequate hardness avoids indentation and scratches on the screen. Thus, the process of manufacturing the screen should be inspected (Hessing, Rational Sub Grouping, n.d.).

After the selection of the key quality process, the key attributes that need to be measured in the process have to be identified. For instance, Brinell hardness can be chosen to measure the hardness of the screen (Hessing, Rational Sub Grouping, n.d.).

A measurement method for the selected attribute and the measuring tools used must be standardised. In addition, one can perform a gage repeatability and reproducibility to set the amount of variation caused by the measurement system (Hessing, Rational Sub Grouping, n.d.).

Once all of this is set, a way to measure the quality of a product is created. To be able to measure and compare this attribute for a process, a subgroup strategy must be developed, as well as a establishing a sampling size and frequency. Subgrouping is the practice of organising data in groups that were produced under the same conditions. For instance, selecting 5 screens from the current population. Subgroup selection has various strategies, however, the most common in industrial settings are to consider 100% of the population or use probabilistic subgrouping. Once a subgrouping strategy is determined the frequency and number of times the characteristics of a subgroup are to be measured has to be determined, establishing the sampling plan (Hessing, Rational Sub Grouping, n.d.).

The collected data is now analysed using adequate SPC methods, based on the subgroup size and type of data. In the initial analysis it is key to calculate the limits where the process can be considered IC, in other words the limits accounting for the natural variability in the process. This can be done if the process is already IC and systematic errors, which derive from special causes of variability, are not present. If it is not IC or there is the presence of systematic errors, the causes for such must be identified and the process recalibrated. Once the process is recalibrated the control limits must be recalculated. The process of calibrating a process and calculating control limits is known as phase I(NIST/SEMATECH, 2012).

Once phase I is completed, the plotting and monitoring of the process can start. This is called phase II. In this phase the goal is to detect when the process starts to deviate from the expected behaviour and assignable variabilities occur. Data is collected following the designated plan through time, plot and contrasted with control limits. If the data follows outside of these limits the process is OOC, and the error that causes such variability must be identified and removed (NIST/SEMATECH, 2012).

For instance, as can be seen in *figure 3* below, until time is equal to *t1* the mean and standard deviation of the analysed characteristic's subgroup are within the calculated boundaries therefore the variation inherent to the characteristic is under statistical control and so is the

process. However, from *t1* to *t2* the mean shifts and takes a new value outside of the control area, this is an assignable variability, and the process is out-of-control. Similarly, from *t2 to t3* another assignable variation occurs when the standard deviation increases thus making this subgroup behave differently than historically IC subgroups. Finally, after *t3* there is an example of assignable variability due to both the mean and the standard deviation shifting the characteristic outside of the control area.

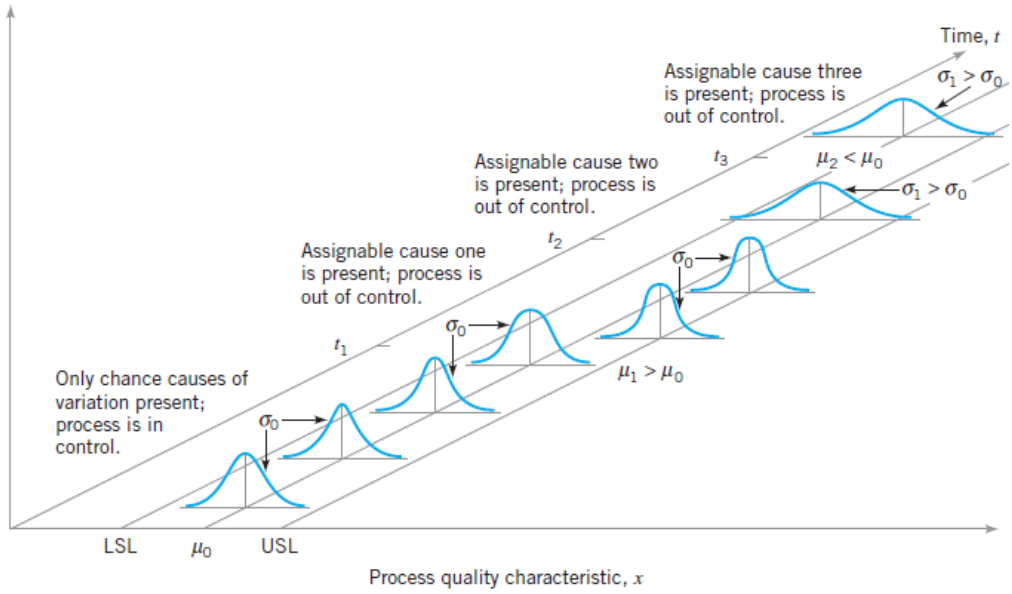


Figure 3. Comparing natural process variation versus assignable variation. Source: Montgomery, 2013

This process is the basis used by one of the most important tools is SPC, control charts.

4.1.3 Control Charts

Control charts present the evolution through time or sample number of a quality attribute that has been calculated from a rational subgroup of a certain process' product. *Figure 4* is an example of a basic control chart, and as it can be observed it also contains three lines, the centre line (CL) the upper control line (UCL) and the lower control line (LCL). The centre line represents the mean value of the quality attribute when the process is in control. The boundary lines represent the zone where nearly all the sample points will fall within, if the process is in control, this range is usually $\pm 3\sigma$ from the centre line, where σ is the standard deviation of the characteristic when in the process is in control. The amplitude of the control range was chosen by Dr. Shewhart due to the following concepts:

1) Chebyshev's inequality shows that for any probability distribution, the probability that a sample is superior to k standard deviations is maximum $\frac{1}{k^2}$. Therefore, 88.9% of points fall within three standard deviations of the mean.
2) In normal distribution 99.7% of the points fall within three standard deviations from the mean. (Shewart, 1923)

This indicates that there will be false positives, on average 1 for every 371 samples if the process follows a normal distribution.

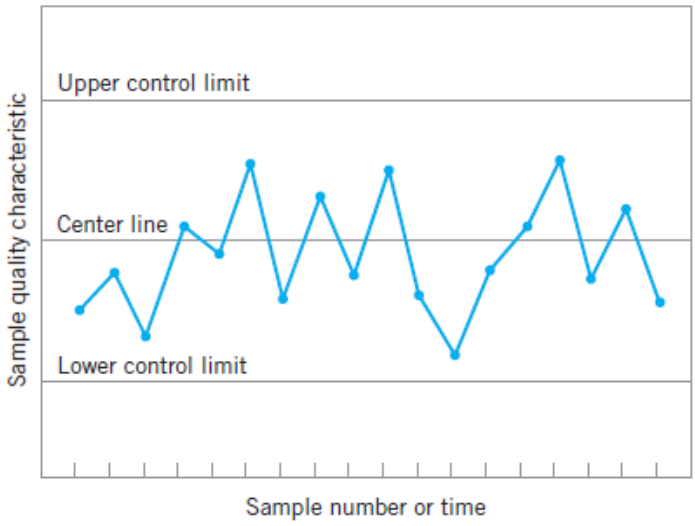
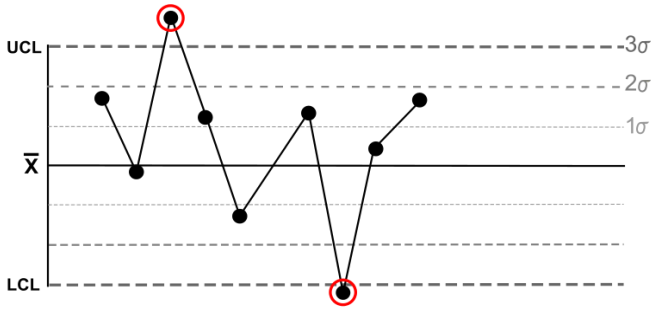


Figure 4. Example of a control chart. Source: Montgomery, 2013

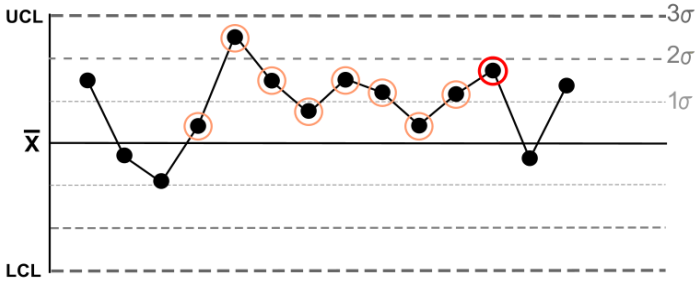
Control charts can have added elements such as $\pm 2\sigma$ and $\pm 1\sigma$ lines to detect certain patterns which are associated with an out-of-control process. The patterns are detected using a set of rules which have been developed based on statistics and experience over time analysing the most common out-of-control situations in industry. In *figure 5* below the most common rules are shown. It is important to detail that the point that will be categorised as an assignable variability is the last point that fulfils a rule, as seen in the image below where the labelled point is the one surrounded in a brighter red circle. The more rules are applied to a specific control chart, the more sensible it will be and the higher the chance of false negatives, but the lower the chance of false positives, and vice versa applying less rules (NIST/SEMATECH, 2012). Every industry uses their combination of the rules detailed below in *figure 5*. For instance, two common combinations of rules are Western Electric and Nelson rules. Western Electric rules include number 1, 2,5, 6, with the modification that rule 2 considers 8 points or

more. On the other hand, Nelson rules include 1 through 8, thus making Nelson rules more sensible to different types of patterns.

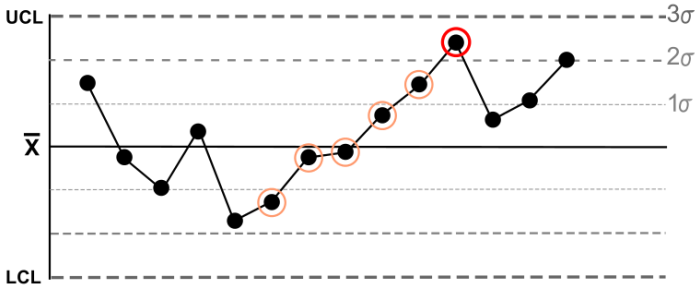
Rule 1: One point is more than 3 standard deviations from the mean



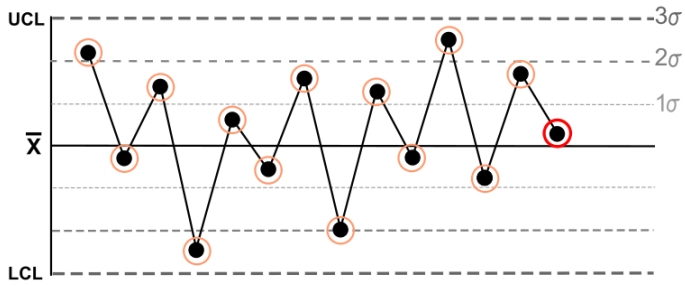
Rule 2: Nine (or more) points in a row are on the same side of the mean



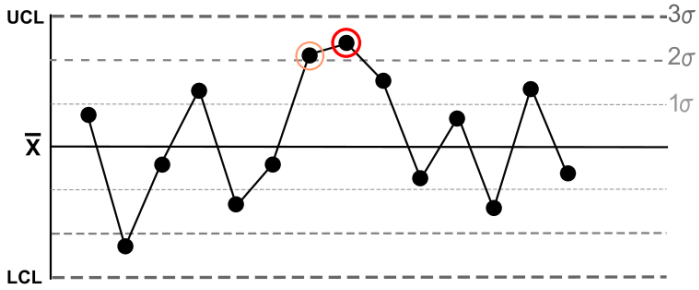
Rule 3: Six (or more) points in a row are continually increasing (or decreasing)



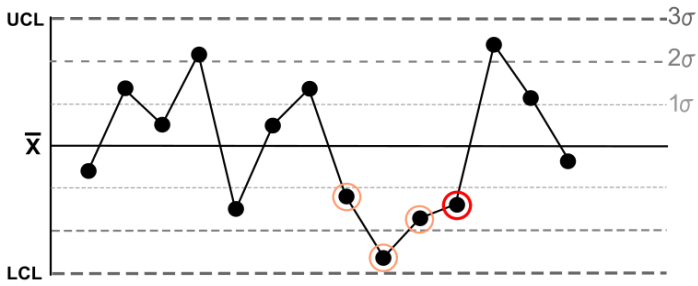
Rule 4: Fourteen (or more) points in a row alternate in direction, increasing then decreasing



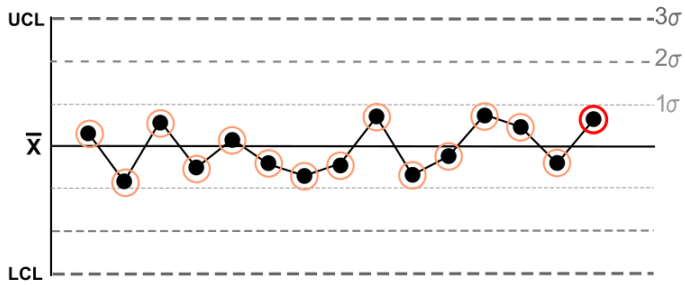
Rule 5: Two (or three) out of three points in a row are more than 2 standard deviations from the mean in the same direction



Rule 6: Four (or five) out of five points in a row are more than 1 standard deviation from the mean in the same direction



Rule 7: Fifteen points in a row are all within 1 standard deviation of the mean on either side of the mean



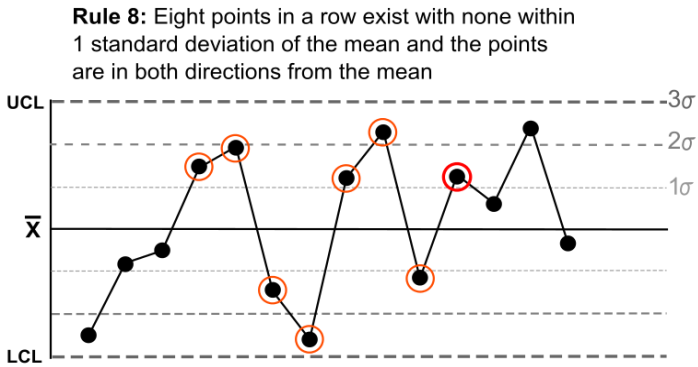


Figure 5. Control chart rules for out-of-control process detection. Source: GMcGlinn, 2008

Once a control chart is built it is the represented and the rules are applied to determine whether a process is in control or not.

4.1.4 I-MR Control Charts

In some cases, it is impractical to obtain subgroups, to create a control chart. For instance, in the chemical industries when batches are homogeneous and the change in measurement is mostly due to the inherent variability while measuring, or in a manufacturing process where each unit can be inspected due to automation technology. Therefore, a chart for sample size of one is used (NIST/SEMATECH, 2012). This chart is known as the individual control chart (I-Chart), in which the raw measured values are represented along with the calculated control limits. It is used jointly with a moving range chart (MR-Chart), which represents the range between two consecutively measured values, along with the calculated control limits. These two charts used together allow to detect assignable variabilities in the process. They are built using following the phase I steps (NIST/SEMATECH, 2012).

Phase I: Calculate control limits from IC data where x_i is an individual observation

1) Calculate the moving range as,

$$MR_i = |x_i - x_{i-1}| \qquad \text{(eq.1)}$$

2) Calculate the moving range and individual observations average
3) Calculate the control limits for each chart as displayed in the *table 1 and 2* below,

	CL	UCL	LCL
MR-CHART	$\overline{MR}$	$D_4\overline{MR}$	$D_3\overline{MR}$
I-CHART	$\bar{x}$	$\bar{x} + E_2\overline{MR}$	$\bar{x} - E_2\overline{MR}$

Table 1. Control limits equations. Adapted from: Institute of Quality and Reliability, n.d

E_2	D_3	D_4
2.66	0	3.267

Table 2. Control limits constants. Adapted from: Institute of Quality and Reliability, n.d

The constants to calculate the control limits are quite large due to the limited observations used for the range estimation. This estimation is done based on the following logic.

First there is a relationship between the mean range, $\overline{MR}$, and the standard deviation of a normal distribution which is dependent on sample size (NIST/SEMATECH, 2012).

This concludes that to correctly estimate the value of the standard deviation of the individual observations the following equation must be used,

$$\sigma = \frac{\overline{MR}}{d_2} \qquad \text{(eq.2)}$$

Where d_2 is a constant that depends on the subgroup size. Therefore, to calculate the individual values chart limit the following equation is used,

$$Contol\ limit = \bar{x} \pm 3\frac{\overline{MR}}{d_2} = \bar{x} \pm E_2\overline{MR} \qquad \text{(eq.3)}$$

Second, the unbiased estimation of the unknown standard deviation for the moving range is represented by,

$$\sigma_{MR} = d_3\sigma = d_3\frac{\overline{MR}}{d_2} \qquad \text{(eq.4)}$$

When this is plugged in the equation to calculate the control limits of the moving range chart,

$$\overline{MR} \pm 3d_3\frac{\overline{MR}}{d_2} \qquad \text{(eq.5)}$$

And if,

$$\boldsymbol{D_3} = 1 - 3\frac{\overline{d_3}}{d_2} \quad \text{(eq.6)}$$

And,

$$\boldsymbol{D_4} = 1 + 3\frac{\overline{d_3}}{d_2} \quad \text{(eq.7)}$$

Then the control limits can be written as seen in *table 1* (NCSS, n.d.).

After completing phase I, the charts are represented for phase II with their corresponding control limits and a combination of the rules displayed in *figure 5* are applied to detect assignable variabilities. Usually in the MR-Chart only rule 1 is applied, whereas for the I-Chart the Western Electric or Nelson rule sets are applied (NIST/SEMATECH, 2012).

4.2 Machine Learning and Anomaly Detection

To give a correct understanding of machine learning this sub-chapter will start by exposing the basic definition of ML and its types. Second, the general types of problems that can be solved with ML and the specific problem of anomaly detection are presented. Third, the description of the process of building an ML model will be explained. Finally, the sub-chapter will end by giving an intuition of how two ML algorithms, relevant to this project, work.

4.2.1 Basic Background

Machine learning is a field within artificial intelligence focused on identifying patterns and relationships from data with the aim to make predictions or take decisions using large amounts of historical data without being explicitly programmed. ML models can be classified in three main branches based on the way they learn, being:

1) Supervised
2) Unsupervised
3) Semi-supervised
4) Reinforcement learning.

In supervised learning, the model learns how to make predictions from data which has been previously classified. For example, a model learning to predict if an e-mail is considered spam

from the key words in other e-mails, which have been categorized as spam or not (Fumo, 2017).

Unsupervised learning tries to group data based on similarities and patterns without explicitly being told the category of the data. For instance, a model creating a cluster of similar types of flowers based on the length of their petals (Fumo, 2017).

Semi-supervised learning occurs when the model has to learn from a dataset where the minority of data is labelled, and the majority is not (Raj, n.d.).

Reinforcement learning is concerned on how intelligent agents must act to maximize a reward system returned by a certain environment. This would be the case of a model teaching an intelligent agent to play a game by simulating such game and letting the agent figure out the best actions to achieve the desired result and behaviour (Fumo, 2017).

In order to correctly discuss machine learning, the following terminology must be clarified:

Features are the input variables used to make a prediction For instance, a feature can be the colour of a petal, used to predict the species of the flower (Google, 2022). An **observation** is a data point, a row, or a sample in a dataset. In a dataset about attributes of flowers an observation can be a petal length of 3 cm and the colour red (Google, 2022). A **class** is one of a number of target values to label an observation. In the recurring example it could be, for the species of a flower, a rose (Google, 2022).

Moreover, to avoid misconceptions due to common confusion between the terms the use of training, validation, and test set in this memoir follow the Google machine learning glossary definitions (Google, 2022, Ripley, 2019). Where they are defined as:

1) A training set is “the subset of the dataset used to train a model.”
2) A validation set is “A subset of the dataset—disjoint from the training set—used in validation.”
3) A test set is “the subset of the dataset that you use to test your model after the model has gone through initial vetting by the validation set.”

4.2.2 Types of Problems

Machine learning techniques can also be classified based on the nature of the problem they tackle, where the general types are:

1. Classification
2. Regression

3. Clustering

Classification problems try to predict the class, of an observation given a set of features. For instance, given a picture, returning whether it is a cat or a dog.

On the other hand, a regression problem, given a set of features, returns the predicted value in a continuous scale. For example, predicting the temperature of a city based on a set of physical measures done.

Finally, clustering tries to group data based on the similarity of their features. The difference between clustering and classification is the type of learning the model uses. Usually, for classification it is a supervised learning model, whereas for a clustering problem the model uses unsupervised learning approach (Raj, n.d.)

Anomaly detection is a specific type of problem that can be tackled by machine learning. Anomaly detection has the goal to differentiate normal data from erroneous occurrences also known as anomalies or outliers (Wikipedia Contributors, 2022). Outliers are rare observations that do not follow the general pattern or do not have similar feature values to the ones observed in the majority of the data. The fact that these are rare observation implies that the dataset is usually imbalanced, however, this is not always the case. The main difference between this and a classification problem is the fact that the outlier observations deviate from the majority of data, which creates an asymmetry in the data. This means that in any data space where various observations of the majority class lie, the minority class cannot be found. On the other hand, in a classification problem, two classes may lie in the same area of the data space. This however does not limit the fact that an atypical majority class observation can be labelled as an anomaly even if no minority class observations are near (Hennign, 2022).

4.2.3 Process of Developing a Machine Learning Model

In this section the general diagram of machine learning model development, schematically represented in *figure 6,* will be discussed, breaking down each individual step.

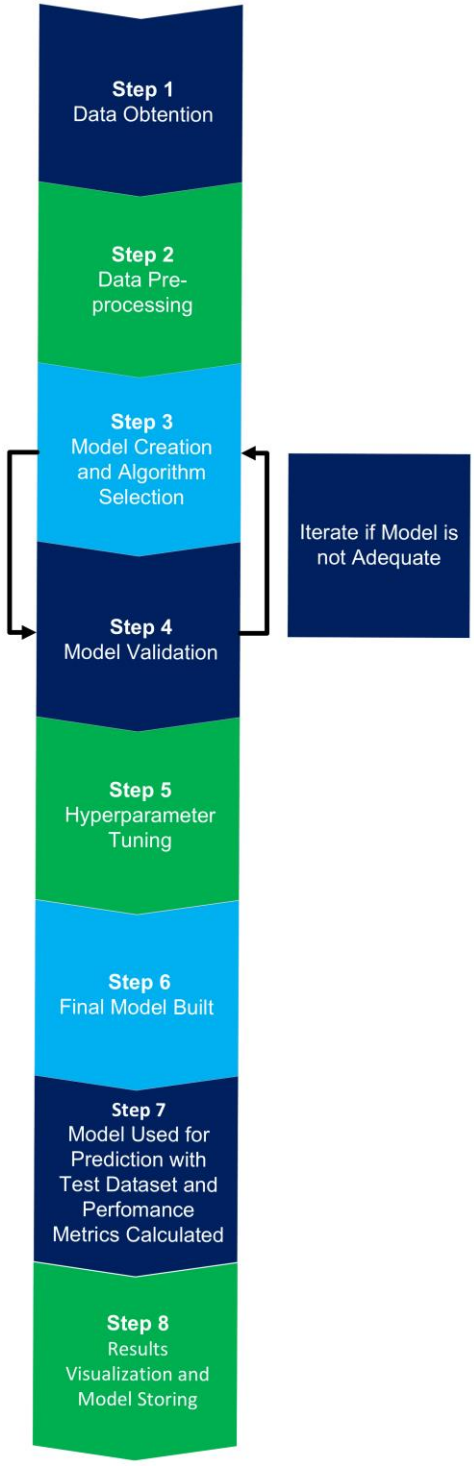


Figure 6. Machine learning project flowchart. Source: personal collection

4.2.3.1 Data Obtention

Raw data is the basis for the creation of an ML model, given the scope of using ML is to make predictions based on what is learned from historical data. Raw data can be obtained from real life measurements, archives, or repositories. However, when there is a partial or total lack of the needed data to represent an environment, it can be obtained through mathematical simulations, to mimic the behaviour of the data in the environment. This is called synthetic data. By mimicking the behaviour of realistic data in the environment, synthetic data can be leveraged to train ML models to then make predictions based on real data (Wikipedia Contributors, 2022) (Devaux, 2022).

4.2.3.2 Data Pre-processing

Raw data many times cannot be directly used to train an ML model, this may be due to issues such as missing values, the presence of errors in the feature values, or other negative factors which affect training of an ML model (scikit-learn developers, n.d.). Moreover, data pre-processing also includes the manipulation of data to enhance the performance of the model which will learn from it. Thus, a set of practices are usually conducted to minimize the impact of the negative factors and facilitate the model's training.

The most relevant practices to this project are feature engineering and scaling.

Feature engineering is the practice of extracting information from raw data to facilitate a model's learning. This can be done through numerical transformations of raw data, grouping aggregated values, and creating new knowledge-based parameters relevant to the problem, known as feature generation (Wikipedia Contributors, 2022).

Scaling is used to equalize the ranges of the different features in a dataset, given spread ranges can cause noise in the results for some types of models. Different types of scaling techniques can be used, and for this project it is done using the following equation,

$$x_{scaled} = \frac{x - \mu}{\sigma} \qquad \text{(eq.8)}$$

This type of scaling is called standardisation and it removes the mean by centring values around 0 and scales them to have a standard deviation of the order of a unit (scikit-learn developers, n.d.).

4.2.3.3 Model Creation and Algorithm Selection

Model creation involves exploring the processed data, which is partially done during data pre-processing as well, and choosing the possible algorithms that can learn from it and tackle the posed problem. More specifically, this step can be done by visualizing the data and observing characteristics such as dimensions, data types and class distributions. Then based on what is discovered an appropriate algorithm is chosen to build the model which will learn from the data and make predictions.

4.2.3.4 Model Validation

Once a model is created with the chosen algorithm it must be validated. This is done with the help of a train/test strategy. The train/test strategy trains the model so it can learn the patterns and the general structure of the data and uses it for prediction with data not used during training. The predictions are compared to the real values' labels to obtain a measure of how well the data from which the model learns allows it to assess new problems. By doing well what is meant is that it is not learning just the specific patterns of the training data, but it is able to generalise what is has learnt to previously unseen data as well, in other words learning intelligently. According to Google's machine learning glossary, validating is "a process used, as part of training, to evaluate the quality of a machine learning model using the validation set. Because the validation set is disjoint from the training set, validation helps ensure that the model's performance generalises beyond the training set." (Google, 2022).

There are various train/test strategies which can be used to evaluate a model. In this project, the relevant strategy is a K-fold cross-validation.

A K-fold cross-validation works with the following logic

1) Shuffles the training set data
2) Divides the training data in k-splits of equal size
3) Trains the model with k-1 splits
4) Uses the remaining split to return predictions
5) Computes evaluation measure based on the predictions and the real answer
6) Stores evaluation measure for the evaluated group
7) Repeats step 3 and 5 to for all combinations of test and train groups

Figure 7 shows the possible combinations of a 5-fold cross-validation (Pedregosa, Varoquax, & Gramfor, 2011).

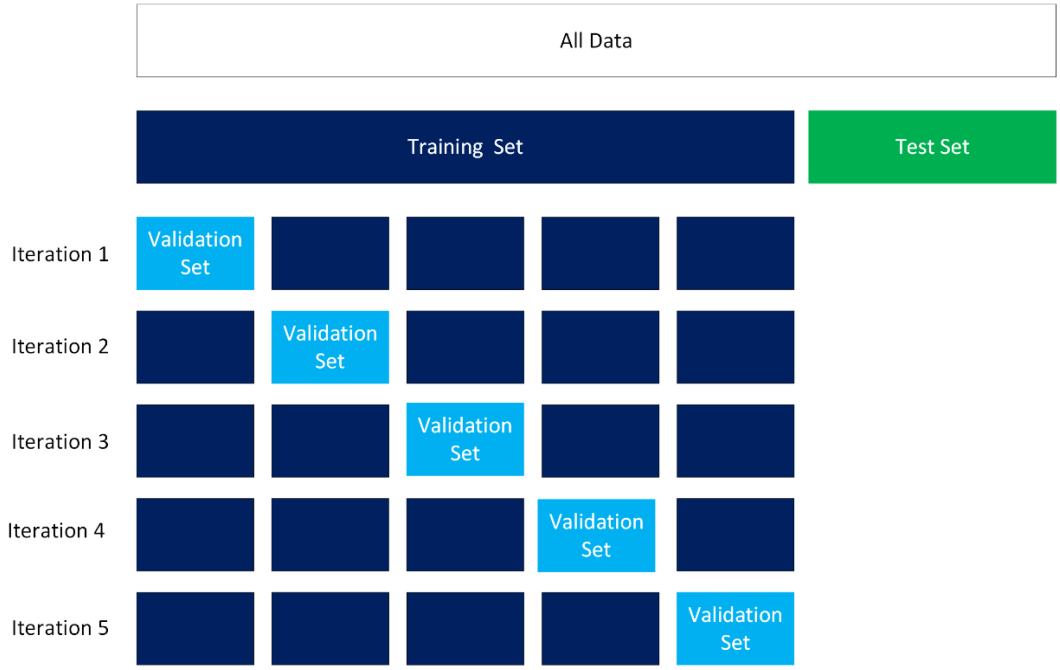


Figure 7. 5-fold cross-validation. Source: personal collection

8) Returns the mean of all individual split evaluation measures

This train/test strategy allows for two things:

1) Obtain a metric that evaluates how the model's training permits it to learn to be able to predict from unseen data
2) Obtain such metric by training and testing with all groups of data, to observe whether a specific grouping can possibly be more significative to the training of the model

Once a measure for the train/test strategy is returned, if it is not adequate, then another model and algorithm that better suit the data have to be found. Otherwise, the model's learning allows it to adequately make predictions from unseen data and it and the next step is tuning its hyperparameters.

4.2.3.5 Hyperparameter Tuning

The complete process of choosing a set of hyperparameters to build the final model can be done as observed in *figure 8*. In order to compare how the different hyperparameters affect the model's training and prediction capabilities a train/test strategy needs to be used, for instance, a k-fold cross-validation.

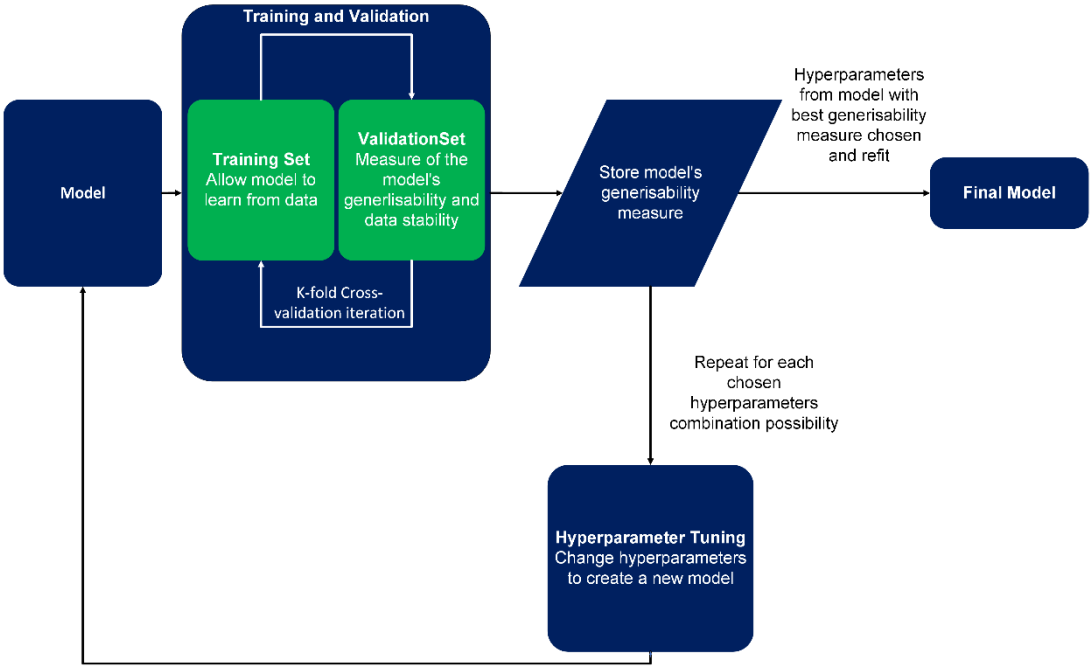


Figure 8. Process of discovery, training and validating, and hyperparameter tuning. Source: personal collection

Following the diagram, model with a certain combination of hyperparameters is trained and evaluated to obtain a measure of its capabilities. Once that is done this measure is stored and the hyperparameters of the model are changed, thus discovering a new model, which has not been trained and validated. This model with the changed hyperparameters will also follow through with the train/test strategy chosen and a measure for its generalisability is also stored. This process is repeated for all chosen hyperparameter combinations. Once all the differently tuned models have been trained and validated, the hyperparameters from the one with the best generalisability score are chosen and the final model built (scikit-learn developers, n.d.).

4.2.3.6 Final Model Building

To build the final model it is refit on the whole training set, with the chosen hyperparameters (Pedregosa, Varoquax, & Gramfor, 2011). The refit is done to come up with the final model, since cross-validation is just a way of evaluating the model and allow to compare the effect of different hyperparameters, not building it (Bogdanovist, n.d.). Finally, the final model is then used to make predictions with the test dataset and determine its performance as will be discussed in the next section.

4.2.3.7 Model Validation Metrics

Once the final model has been chosen, it can be used to make predictions with data that has not been used during the train and validate stage, an independent test set. This part of the data is also known as holdout set, and it is used to see the performance of the final model to make predictions.

In order to be able to evaluate a model's performance, and hence compare it with other models, the following scoring metrics must be understood and correctly labelled for this problem,

1) Confusion matrix and its parameters
2) Precision, recall and F1-score

A confusion matrix is a visualization that compares the predicted class of each observation from a model to the real observations' class, this can be seen in the *figure 9* below. In the *figure 9*, it can be observed that true negatives (TN) and true positives (TP) correspond to correct predictions done by the model. In contrary, a false positive (FP) and a false negative (FN) occurs when the data is labelled incorrectly compared to its true value.

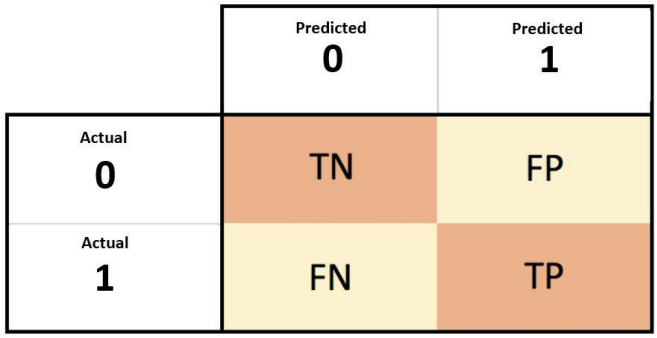


Figure 9. Confusion matrix illustration. Source: Bhagwat, Abdolahnejad, & Moocarme, 2019

From this it is possible to obtain three other metrics, with the following formulas,

$$Precision = \frac{TP}{TP + FP} \qquad \text{(eq.9)}$$

The precision is the frequency with which a model was correct when predicting a positive class (Google, 2022), in this case OOC.

$$Recall = \frac{TP}{TP + FN} \qquad \text{(eq.10)}$$

Whereas the recall is a measure of the correctly identified positive class observations out of all positive class observations (Pedregosa, Varoquax, & Gramfor, 2011).

$$F1 - score = 2 \cdot \frac{(Precision \cdot Recall)}{(Precision + Recall)} \quad \text{(eq.11)}$$

The F1-score is the harmonic mean of both.

The higher these scores the better the model's predictions.

4.2.3.8 Model Storing

The final step is to store the model. This can be done by serializing the model's data object, using libraries such as pickle or joblib. Serialization means to converting a python object into a byte stream so it can be stored in a file, which can be used later to recover the original object through deserialization (Agrawal, 2014).

4.2.4 Isolation Forest Algorithm

Isolation forest is an unsupervised classification algorithm, that is commonly used in anomaly detection and was developed by Fei Tony Liu in his PhD thesis (Liu, Ting, & Zhou, 2008). It is based on the idea that isolating anomalies, given they are few and different, will be easier than isolating normal instances. Visually this can be seen in the following *figure 10*. As seen in the left quadrant, when isolating a non-anomaly point, the number of partitions will be greater as it lives in a higher density zone. On the other hand, an anomaly is easier to isolate because it has a different value and is not found in a high-density zone.

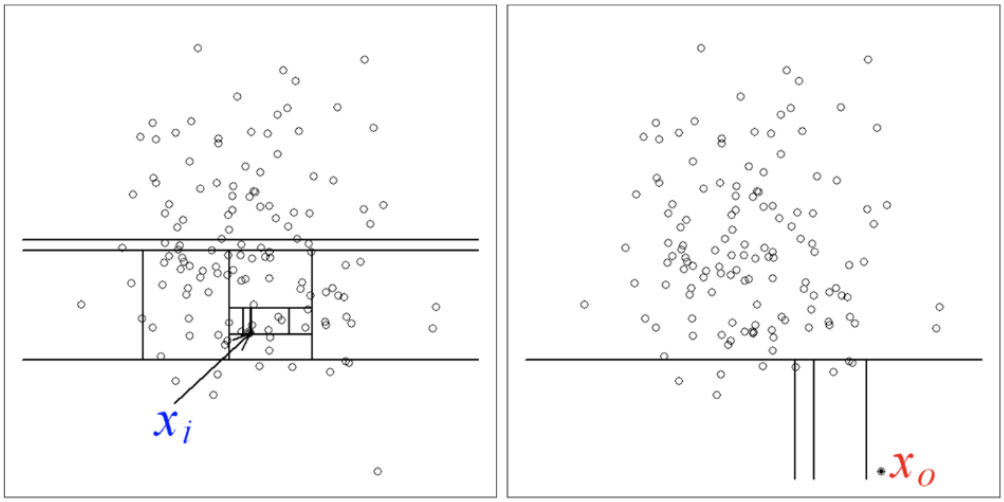


Figure 10. Visual representation of isolation forest partitions. Source: Liu, Ting, & Zhou, 2008

Before describing the algorithm, the terms isolation tree and path length have to be defined.

An isolation tree is composed by tests and nodes, and it is a type of data structure which is in used in this algorithm to represent a set of partitions. Nodes can either be external with no child (no continuation) or an internal with one test and two daughter nodes T_l and T_r (Liu, Ting, & Zhou, 2008). Nodes are associated with a set of data points in the training set (Wikipedia Contributors, 2022). On the other hand, a test consists of comparison between a feature q and a split value p, which is a value between the maximum and minimum of the selected feature. An example of a tree can be seen *figure 11* below, where internal nodes are represented by the dark blue boxes, and external nodes by the green boxes.

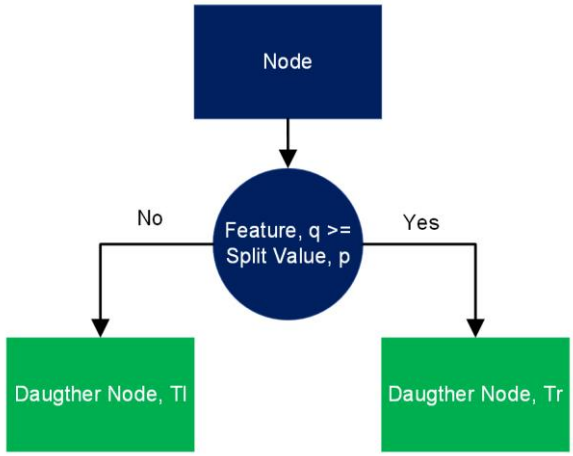


Figure 11. Visual representation of an isolation tree. Source: personal collection

The path length, defines the number of edges an observation goes through, starting from the root node of a tree to reach an external node.

Now the general algorithm's logic can be seen in *figure 12* below.

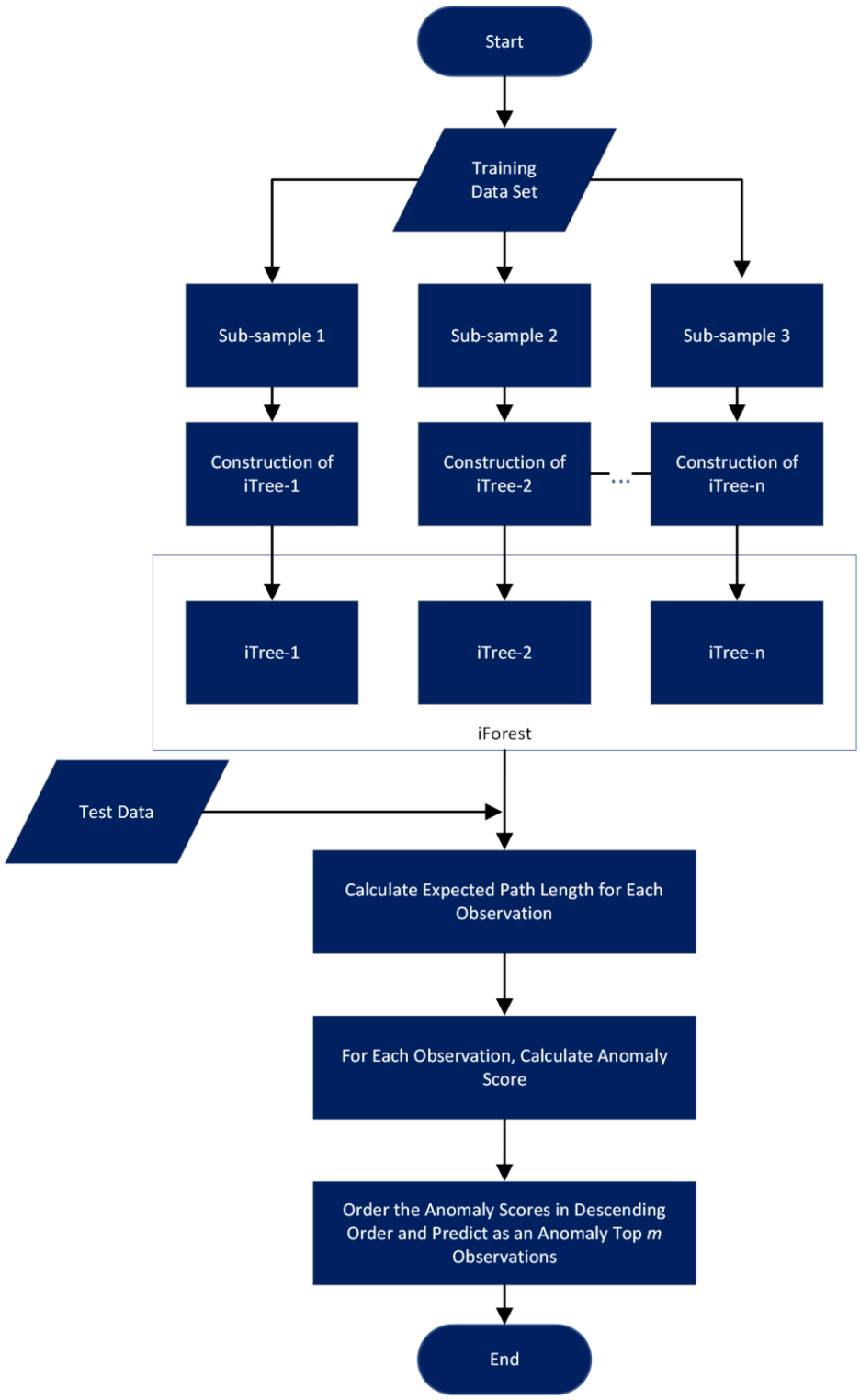


Figure 12. Isolation forest algorithm's logic. Adapted from: Xie, 2019

The algorithm starts by extracting a random sub-sample from the training data. Then an isolation tree is constructed. This is done by recursively partitioning the sub-sample until a specific tree height is reached or observations are isolated.

For instance, the process of building the tree seen in *figure 13 would* be as follows.

1) Let the sub sample have 3 features, a, b, and c, and be composed by 5 observations
2) The root node, node 1, related to the 5 observations is created
3) A test condition is created by randomly selects feature a, and a random split value $p1$, which is a value between the minimum and maximum values that feature a can take found in the training data
4) The 5 observations related to the node 1 are separated based on their value of a

As seen 4 of the observations have a value of a above or equal to $p1$ and one has it below.

5) From this test two daughter nodes are derived

One of them is an external node, node 3 in green, given it is related to only one observation, which cannot be further split. The other, node 2 in blue, being a new internal node related to 4 observations.

6) A test is created, for feature c with split value $p2$
7) Data related to the node 2 is consequently split

This is done until isolating all observations in the sub-sample.

For each tree, the feature and split value chosen in the test of each internal node is stored, along with other data, such as node ids and ids of their connections, which define the structure of the tree. The structure has to be recorded in order to know where the external nodes are located, as it is crucial for the prediction phase.

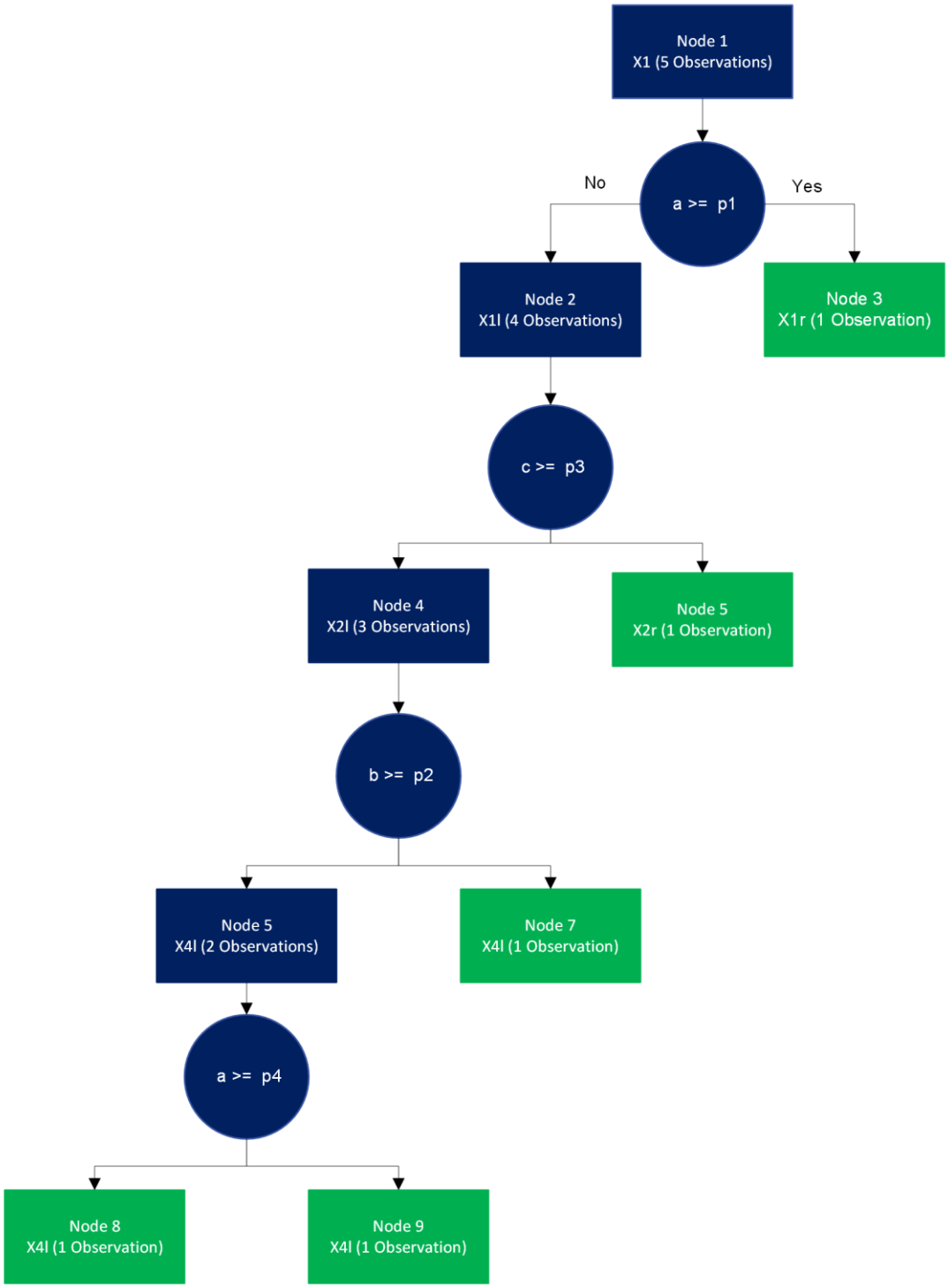


Figure 13. Visual representation of an isolation tree. Adapted from: Anello, 2021

This is process of tree creation and storing is repeated for a set number of sub-samples. All the created isolation trees form what is known as an isolation forest. Now the algorithm is ready to make predictions based in the test data.

To do that, for each observation in the test set the path length is calculated for each tree in the isolation forest created with the training data and stored. The path length is calculated letting the observation "pass" through the nodes of a tree based on the test conditions that were stored during the training stage. This is done until the observation reaches an external node. For instance, for a test observation with parameter a greater than $p1$ and parameter c smaller than $p3$ for the tree in *figure 13* the test instance would start from node 1. Then its path is decided by the test condition and it this case "pass" through to node 2. Once in node 2 the observation's path will be decided based on the test condition it will be "passed" through, in this case to node 5. The observation has reached an external node and thus its path length is equal to the number of testing instances it has gone through which is 2.

As this is done for each isolation tree created the average of the path length can be calculated, this value is also known as the expected path of the tested observation.

The next step is the calculation of the anomaly score for each observation. The anomaly score is essentially a normalised ratio between the:

1) Average path length to get to an external node of the isolation trees in the isolation forest created by the training data
2) The expected path length of an observation (Arpit, 2020).

The intuition being that if the expected path length of an observation is higher than the average path length of the isolation trees, thus signalling the observation is hard to isolate and probably not an anomaly, the ratio takes a value closer to 0. On the other hand, if it the expected path length of an observation is lower than the average path length of the isolation trees in isolation forest, signalling the observation was easier to isolate, the ratio takes a value closer to 1. (Liu, Ting, & Zhou, 2008).

Finally, once the scores are calculated for each observation, they are sorted in descending order, and the top m observations are labelled as anomalies, where m is a hyperparameter.

4.2.5 Support Vector Machines (SVM) Algorithm

SVM is a supervised classification algorithm which is based on the idea of creating a hyperplane, in the case of binary classification, or a set of hyperplanes, in the case of a multi-class classification, that separate a couple of linearly separable classes of data with the

highest possible margin (Fletcher, 2008). In a two-dimensional case a set of hyperplanes and how they separate data based on their class can be seen in the *figure 14* below.

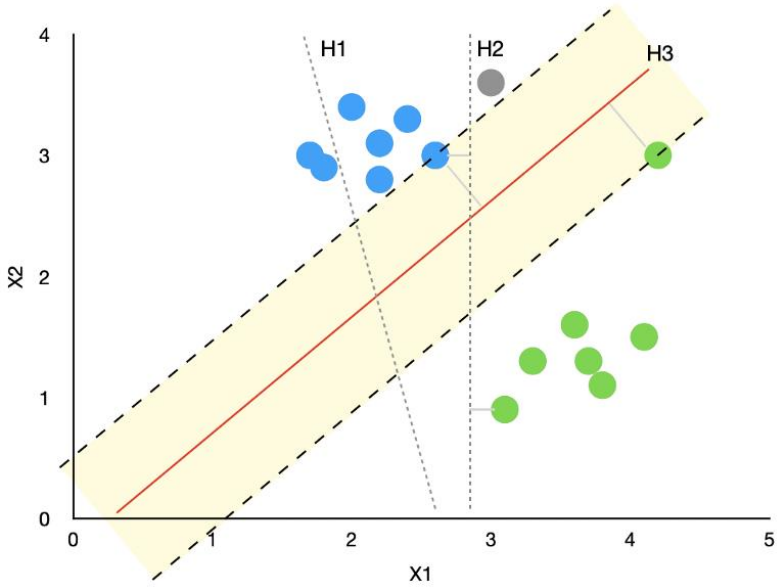


Figure 14. Hyperplane separating two classes. Source: Dobilas, 2021

Observing this case there are three hyperplane possibilities H1, H2 and H3. H1 is not able to separate the classes thus it would not be a valid choice of hyperplane. H2 does separate the classes but the margin is low. Thus, if a new point such as the one in grey is to be classified it would lie on the green side of the hyperplane, when it seems like it would be more fitting to be of blue class. Finally, H3 is the hyperplane that separates the two classes with the highest possible margin, shaded in orange. Support vectors are the data points that lie closest to the margins of the hyperplane, on in this case lie on it.

When two classes of data are completely linearly separable then a hard margin can be used where no data points can lie in such margin, as seen in *figure 14* above. When it is not possible a soft margin has to be used. The soft margin gives a certain flexibility to the amount of data that can be found within the margins. This flexibility is given by a hyperparameter C which gives a weight to what is known as the “slack” parameter ξ (Dobilas, 2021). The effect of soft margin can be seen below in *figure 15*. As seen, soft margining allows for one of the data points of the green class to fall within the margins allowing for greater margins to be found.

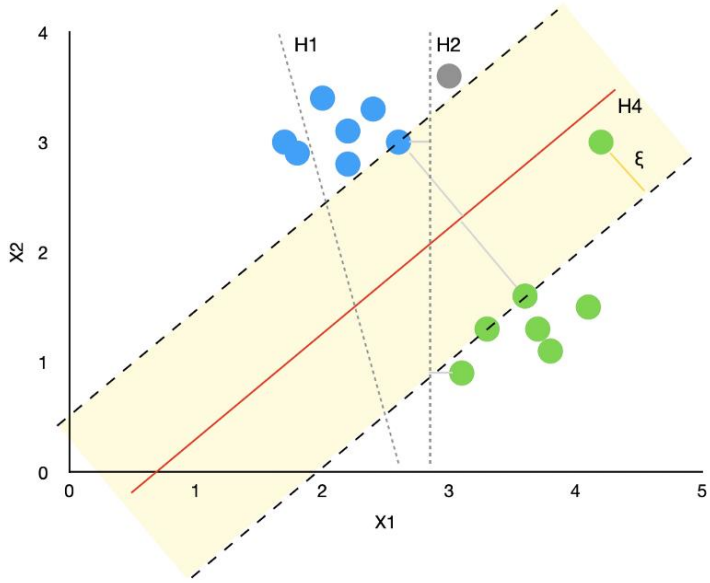


Figure 15. Visualization of the effect of using a soft margin. Source: Dobilas, 2021

SVM can also be used in non-linear problem with the help of what is known as the kernel trick. Essentially the kernel trick, uses a specific kernel function that brings the original non-lineal problem, and it transforms it into a lineal one within a higher dimensional space. Conceptually what is done can be seen in the *figure 16* below, where data is represented in a higher dimension in which the data can be separated by a hyperplane (Willimitis, 2018). In this dimension de SVM algorithm can be trained, and the hyperplane function and the support vectors stored.

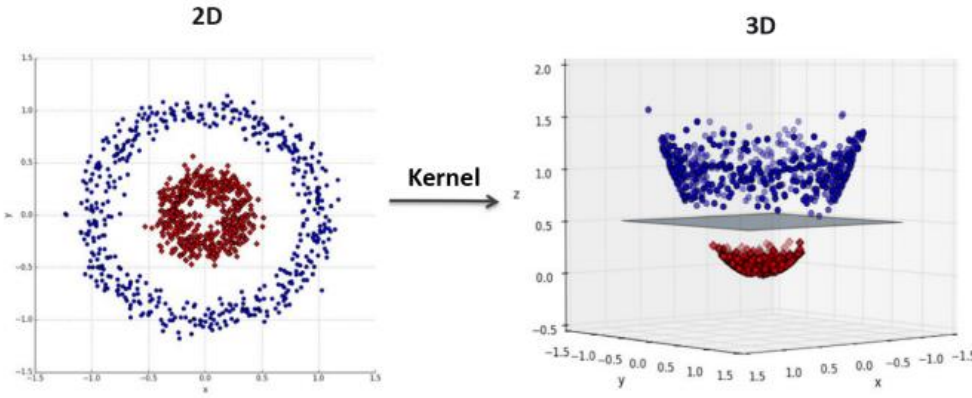


Figure 16. Kernel trick representing data in a higher dimension where it can be linearly separated. Source: Hachimi, 2020

To find the optimal hyperplane the data is defined as vectors where x_i is the i[th] observation's features and y_i is the class it belongs to. Formally,

$$D = (x_0, y_0), \dots, (x_{l-1}, y_{l-1}), x \in \Re^n, y \in -1,1 \quad \text{(eq.12)}$$

The hyperplane is defined as,

$$H_0 = \langle \mathrm{w}, \mathrm{x} \rangle + \mathrm{b} = 0 \quad \text{(eq.13)}$$

Which is the scalar product of a vector of weights w and the variables x_i in the dimension of the hyperplane. Knowing that the constraint that all the points in the analysed dimension must be in the correct side of the hyperplane it can be defined that,

$$\mathrm{yi} \cdot (\langle \mathrm{w}, \mathrm{x}_i \rangle + \mathrm{b}) \geq 1, \mathrm{i} = 0, \dots, \mathrm{l} - 1 \quad \text{(eq.14)}$$

Two additional hyperplanes are defined such that they are parallel to H_0 they represent the tips of the support vectors,

$$H_1 = \langle \mathrm{w}, \mathrm{x} \rangle + \mathrm{b} = 1 \quad \text{(eq.15)}$$

$$H_2 = \langle \mathrm{w}, \mathrm{x} \rangle + \mathrm{b} = -1 \quad \text{(eq.16)}$$

The distance between these two hyperplanes can be defines as,

$$d = \frac{2}{\|w\|} \quad \text{(eq.17)}$$

All of this can be visualised in the following *figure* 17,

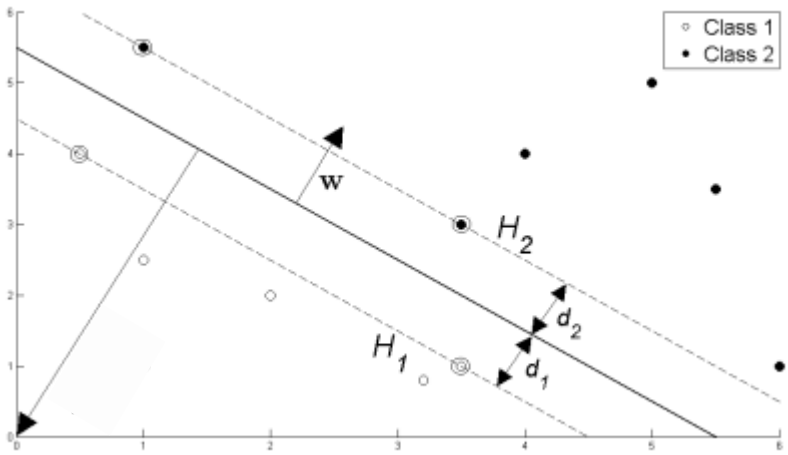


Figure 17. Representation of hyperplane. Source: Fletcher, 2008

Given the definition of the distance between hyperplanes the problem is now an optimisation one where d must be maximised, or $\frac{\|w\|^2}{2}$ minimized, by changing H_1 and H_2, while fulfilling the constraints described in (eq.4). This can be done using the Lagrangian multiplier method to obtain the optimum values for x and b (Berwick, n.d.) (Kantardzic, 2011).

To relax the constraints and create a soft margin the optimisation parameter is modified to,

$$\frac{\|w\|^2}{2} + \mathrm{C} \cdot \sum_{i=0}^{l-1} \xi_i \qquad \text{(eq.18)}$$

Where C is an arbitrary number and ξ is the distance from a hyperplane with the points that are in the incorrect part of it as seen in *figure 15*.

Once the hyperplane is found with the help of the training data, it's support vectors are stored. Now if an observation is tested depending on which zone of the hyperplane it resides it a prediction can be made.

5 Technical Memoir

In this chapter the methodology to develop this project and accomplish the objectives will be discussed, as well as the results obtained.

5.1 Methodology

A diagram to outline the general steps taken to carry out the project can be seen in *figure 18* below.

As seen the first step is the creation of synthetic data, in green, to create DS1.

After creating a base dataset, SPC automation is developed, and ML models are created. These two steps are done independently.

For SPC automation the creation and validation of automated control chart rules is conducted as seen in *figure 18* in orange. This is done with the help of manually created testing observation sets.

On the other hand, ML models are created. To do that first a set of data pre-processing steps are conducted, as seen in *figure 18* in light blue. Then the process of developing an ML model is followed, and two models are obtained as a result. This is seen in red in *figure 18.*

Once control chart rules and ML models are created and validated, the SPC methods and ML models are used to make predictions. From these predictions a set of performance measures are obtained.

Further insight of the main steps will be given along the chapter.

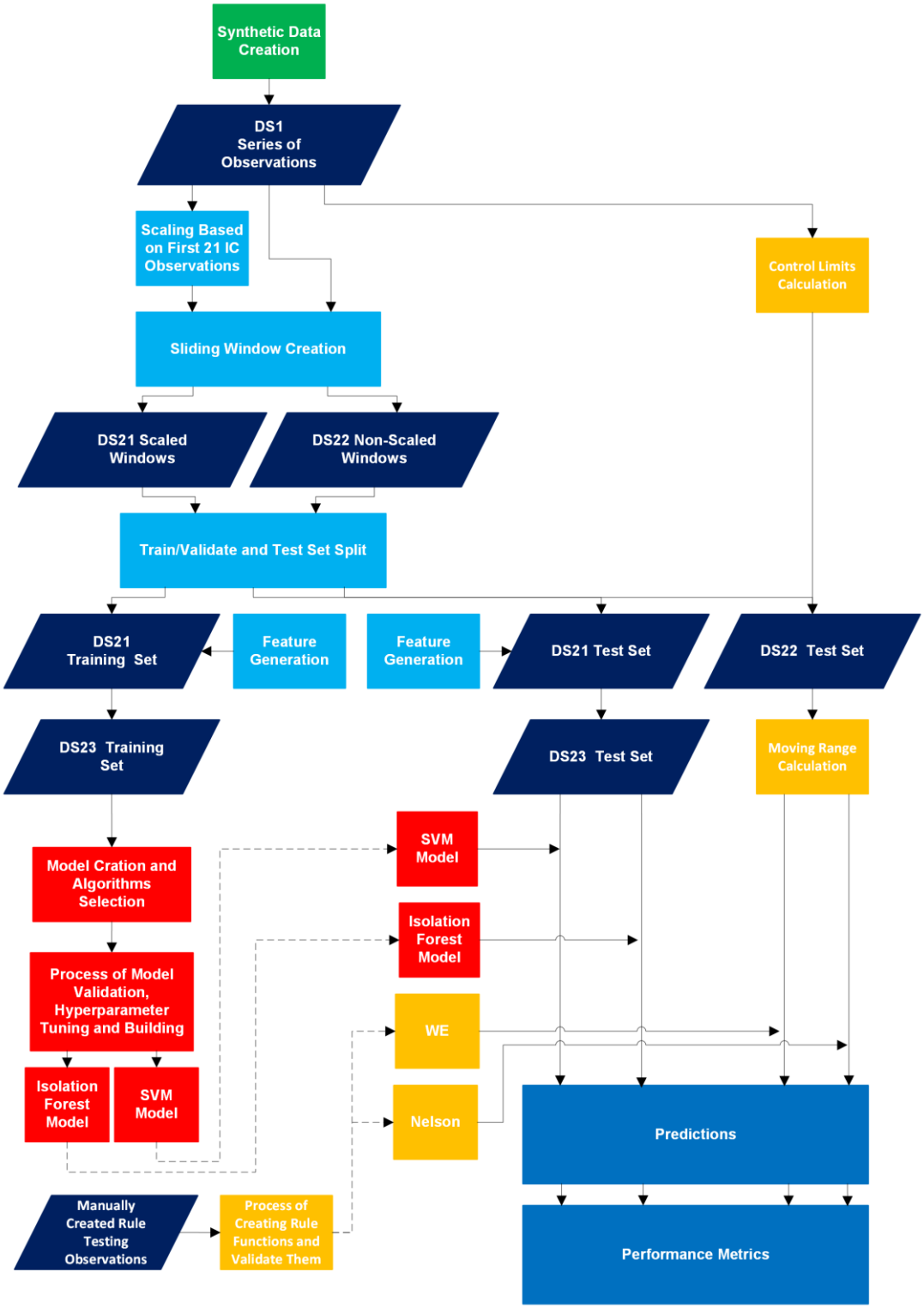


Figure 18. Diagram of the methodology to develop the project. Source: personal collection

5.1.1 Synthetic data creation

A dataset is needed to be able to train (in the case of machine learning), evaluate and compare different methods to tackle a problem. In this case the problem being to identify assignable variabilities in a process. Even if there is an abundance of industrial data, there is no benchmarking dataset representing real processes where assignable variabilities, causing OOC situations, are labelled (Xu, et al., 2019). Labels are needed to be able to evaluate models, compare them and depending on the case, to train a supervised model. Moreover, even if other synthetic datasets can be found (Alcock, 1999), the patterns are presented with fixed number of observations, and what is desired is that these patterns are present with variable length. This should be done to allow for representability as patterns do not always have the same length in industry. For these reasons, one of the outcomes of this project is the generation of a dataset to mimic the typical IC and OOC behaviours of quality attributes in the manufacturing industry which can be observed in *figure 19*. The definition of the dataset and the procedure to create will be further outlined.

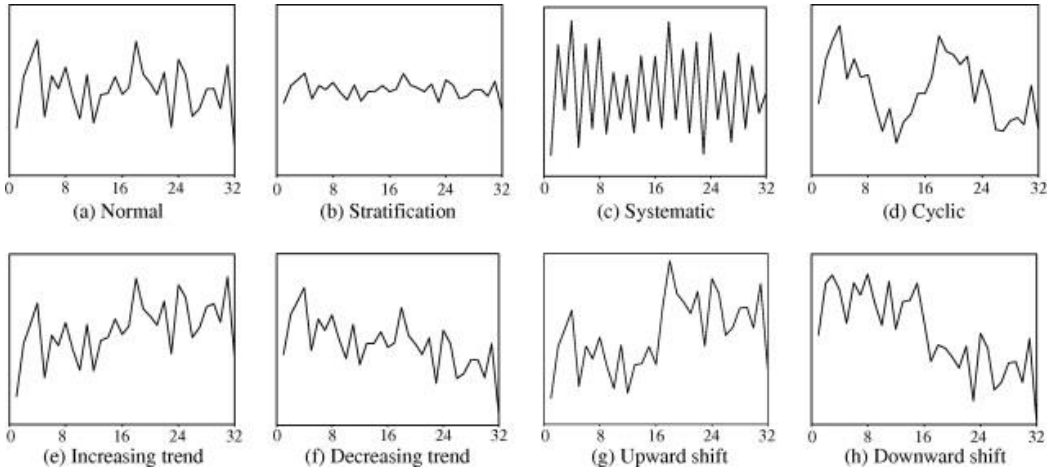


Figure 19. Basic control chart patterns. Source: Gauri & Chakraborty, 2009

5.1.1.1 Definition

The created dataset DS1, will be fully synthetic, meaning all the observations will be created through mathematical functions that simulate the behavior of original data. It will represent all the patterns seen in *figure 19* and more importantly it will label whether the observations come from an OOC pattern or the normal pattern.

To do that, the created dataset will be formed by various series of observations of size equal to 60 observations, where the first 21 are simulated with the normal pattern of mean 10 and standard deviation 1. After the 21st point, one of the OOC patterns starts until reaching the

designed size of the observation series. The start of the pattern escalates one position per every series created, therefore for each type of pattern there are 40 series of 60 observations. If this is done for each pattern, the total number of series becomes 320 (including 40 series with completely IC observations). This substructure is repeated five times, thus delivering a total number of 1600 series. This way more data can be used to train the ML models. The observations dataset will be accompanied by two auxiliary ones one of which contains the class of the of the individual observations and the other containing the type of pattern and the position where the pattern starts for each series of observations.

These datasets will be created in three separate data frames:

1) TimeSeries1.csv
2) TimeSeries1_Classification.csv
3) TimeSeries1_Aux.csv

With the help of the pandas library available in python. The observation data frame will be structured to have each series as a row, where each column corresponds to a single observation. The column names correspond to the position of the observation starting from 0. The class data frame follows the same structure, but each column returns the class of the corresponding observation in such position. Observations simulated with OOC patterns are labelled by the digit 1 and the ones created with the normal ones by the digit 0. The final data frame will contain two columns, each row corresponding to the auxiliary data for each series of observations. The column names will be “Position”, which will return the position of the start of the pattern (starting to count from base 0), and “Pattern Type” returning the type of pattern present in the corresponding series of observations.

These data frames will be saved as three separate CSV files for further use in the project. The decimal separator is a point, the column separator a comma and the row separator is a tab. In order to easily distribute the CSV files are joined in a folder and it is compressed in a zip file.

5.1.1.2 Creation

The basic patterns in *figure 19* are the most common OOC patterns in manufacturing, and they were identified by Western Electric Company in their statistical quality control handbook (Western Electric Company, 1956). A brief description of each pattern and its possible causes or relation with the process is stated below.

1) The normal pattern indicated an in-control production process.

2) A stratified pattern shows that the data has a lower variance, and it is more packed around the mean. This can occur due to computational errors (Xu, et al., 2019).
3) A systematic pattern displays a high point followed by a low point and then again by another high point, thus enabling to foresee point to point fluctuations.
4) A cyclic pattern displays a periodic occurrence of peaks and throughs. These can be associated with periodic events within a process such as voltage fluctuations.
5) Trend patterns display the gradual increase or decrease in the quality attribute represented in the chart. This can be associated with tool wear, operator fatigue, or lack of supervision.
6) Shift patterns occur when there is a sudden rise or fall of the mean in the data. This is usually caused by the modification of the process, such as having new operators, new materials, or device faults.

To create these basic patterns, the equations in *table 3* are used, where most of the parameters were referred to Xu, et al., 2019.

DESCRIPTION	EQUATION	PARAMETERS
NORMAL	$y_t = \mu + r(t) \cdot \delta$	$\mu = 10$
STRATIFIED	$y_t = \mu + r(t) \cdot \delta'$	$\sigma = 1$
SYSTEMATIC	$y_t = \mu + r(t) \cdot \delta + d(-1)^t$	$\delta = 1\sigma$
CYCLIC	$y_t = \mu + r(t) \cdot \delta + a \cdot sin(2\pi/T)$	$r(t) \sim \mathrm{N}(0,1)$
UPWARD TREND	$y_t = \mu + r(t) \cdot \delta - t \cdot g$	$\delta' \sim \mathrm{U}(0.2\sigma, 0.4\sigma)$
DOWNWARD TREND	$y_t = \mu + r(t) \cdot \delta - t \cdot g$	$d \sim \mathrm{U}(1\sigma, 3\sigma)$
UPWARD SHIFT	$y_t = \mu + r(t) \cdot \delta + k \cdot s$ $k = 1\ if\ t \geq P, else\ k = 0$	$a \sim \mathrm{U}(1.5\sigma, 2.5\sigma)$ $T \sim \mathrm{U}(8, 32)$ $g \sim \mathrm{U}(0.05\sigma, 0.2\sigma)$ $s \sim \mathrm{U}(1\sigma, 3\sigma)$
DOWNWARD SHIFT	$y_t = \mu + r(t) \cdot \delta - k \cdot s$ $k = 1\ if\ t \geq P, else\ k = 0$	$t = 1,2,3 \ldots L$

Table 3. Mathematical formulas to simulate basic control chart patterns. Adapted from: Xu, et al., 2019

Each basic pattern was created as a python function with the help of the NumPy and random libraries.

The NumPy library is used to:

1) Create arrays
2) Manipulate arrays

The random library is used to:

1) Pull random samples from a normal distribution centred around 0 and with standard deviation of 1. This allows to create the $r(t)$ parameter in *table 3.*
2) Draw random samples from a uniform distribution between the selected ranges. This allows to create the parameters δ', d, a, T, g and s in *table 3*.

The input parameters for each of the basic pattern functions are the mean of the process, the standard deviation, and the number of observations to be created. However, in the case of the shifts patterns another parameter is defined based on when the shifts start. An example of the code to create a basic pattern is the following,

```
def normal(mu, sigma, numobs):
    return(mu + sigma * np.random.normal(size = (numobs)))
```

Code Snippet 1. In control basic pattern code in python. Source: personal collection

As observed the code is for the IC process and it returns a vector of dimension *(numobs, 1)* containing observation values. Based on the information from *table 3* the pattern will have a mean of 10 and a standard deviation of 1 and the inevitable accidental variability inherent to the process with no assignable cause is created thanks to $r(t)$.

To return a stratified pattern, what needs to be done is multiply each point of an IC observation series with a value between 0.2σ and 0.4σ. This is done by the parameter δ' in *table 3*.

The systematic pattern is created by adding to each observation a value d, if the observation's position is even and resting such value if the observation's position is odd. This value is known as the degree of system departure (Xu, et al., 2019).

For the cyclic patterns, a sinusoidal wave with a random amplitude a, and period T is added to each corresponding observation.

Creating a trend is done by adding to each value a linearly increasing vector with slope g in the case of an upward trend and resting such vector in the case of a downward trend.

Finally, to create a shift a value s is added or subtracted to all observations.

The procedure followed to create the final dataset arrays is as seen in *figure 20:*

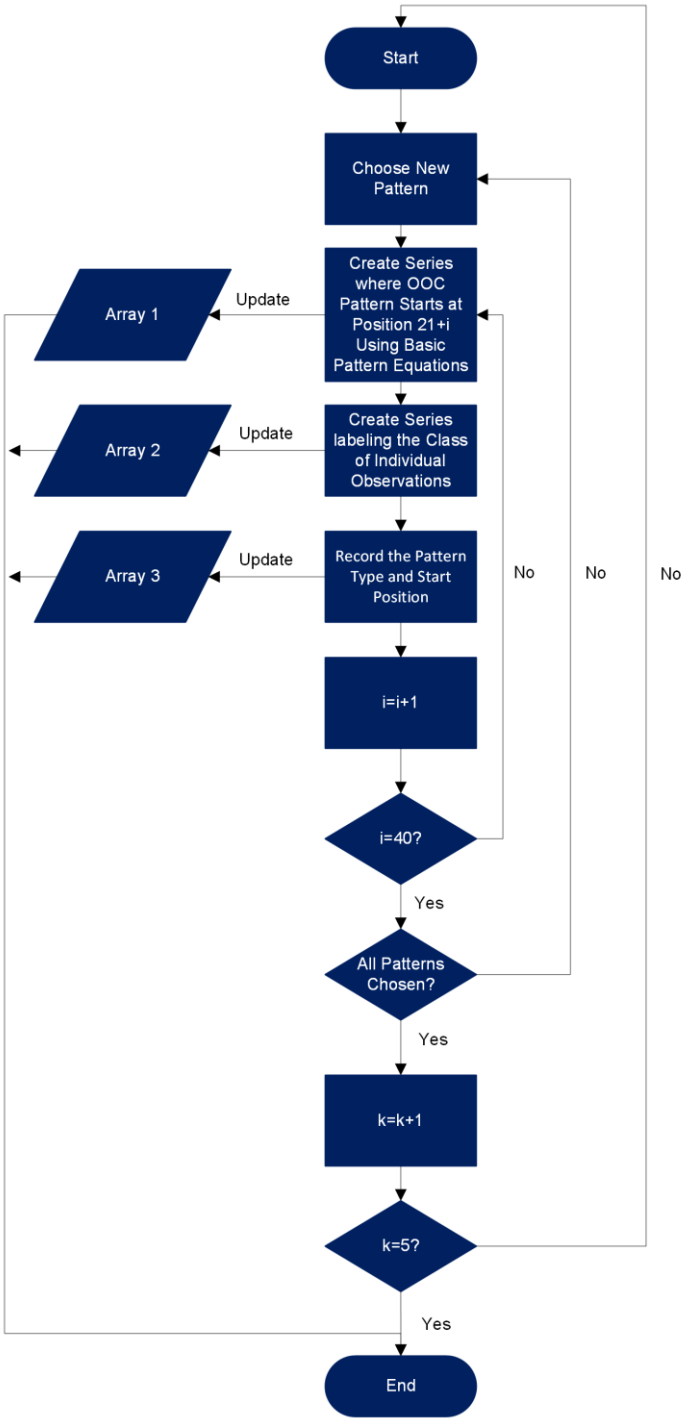


Figure 20. Logic followed to create observation dataset. Source: personal collection

The arrays are then saved as separate data frames, converted to CSV files, joined in a folder which is then compressed in a zip file, as described in the dataset definition.

5.1.2 Data Pre-processing

One of the objectives of the project is to create two ML models to identify OOC instances. To do that data pre-processing as a step in the process of ML model creation must be performed.

As explained data pre-processing is done as a step of developing an ML model to fix errors and negatively affecting factors in the data as well as manipulating it to improve the machine learning model's learning and performance. Given the dataset has been synthetically created, the data is considered to have no errors of negatively affecting factors. Thus, this phase of the project will be centred in feature engineering.

In this project three main manipulations are done to the dataset DS1:

1) Scaling (using eq.8) each series of observations based on the corresponding mean and standard deviation of the first 21 observations, which are simulated with the normal pattern

This is done to scale the possible anomalies and consider the expected mean and standard deviation from the process' historical data, which in this case is defined by the first 21 observations. Therefore, it allows for the use of the machine learning model with series of observations that have different means and standard deviations.

2) Creating a set of sliding windows of 20 observations for each series of observations to give DS2

For reference of what is meant by a sliding window, a visual representation can be seen in the *figure 21* below. When applied to the series of observations given there are 60 observations in a single series, and the size of the window is of 20, the number of sliding windows for each series will be 41.

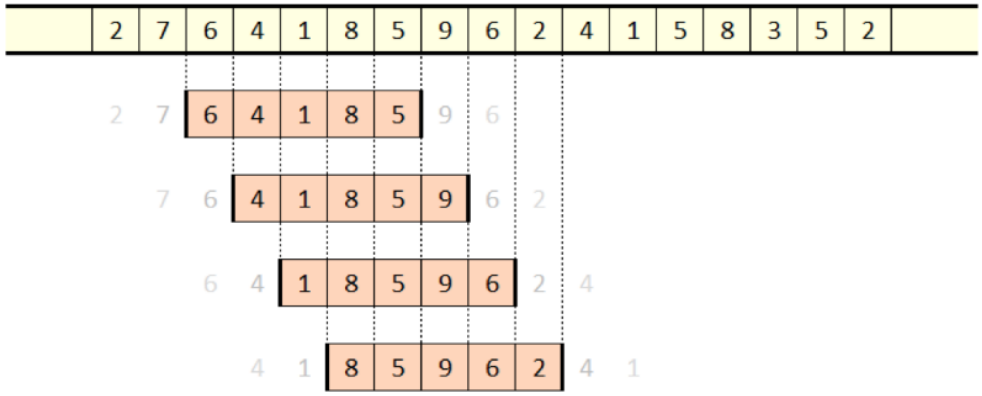


Figure 21. Sliding window algorithm. Source: Lu, 2019

The intuition and reason behind this process is that in the manufacturing industries anomalies do not come just as single point anomalies, but also as collective patterns. Therefore, the only way for a machine learning model to detect them is to look at a window of observations. In other words, this step is done to contextualise the current observation based on the previous observations.

Parallelly to the creation of the windows, these are labelled in an auxiliary dataset with the digit 1 if they contain observations that were simulated with a OOC pattern, or 0 in the case the observation was simulate with the normal pattern.

3) From each window created extract a set of statistical descriptors, or features. This practice is known as feature generation.

The reason behind this practice of feature generation is that if a set of descriptive features are created for each window, where the features have a different distribution for windows with OOC observations than for windows with IC ones, the easier it will be for a machine learning model to differentiate them.

The features generated are summarised in the *table 4* below.

FEATURES FOR EACH WINDOW
Last Value
Mean of all Observations
Standard Deviation of all Observations
Mean of Last 5 Observations
Standard Deviation of Last 5 Observations
Finite Difference of the Last 2 Observations
Kurtosis
Number of Changes in the Slope
Weighted Average based on Number of Observation
Slope by Least Square Fit Line (LS)
Number of Mean Crossings
Mean Distance Between Consecutive Observations
Range of Slopes of LS Lines Fitted to 6 Different Subsets of the Window

Table 4. Features created for the sliding windows. Source: personal collection

Some of the features have been chosen based on *Feature-based recognition of control chart patterns* paper by Gauri et al., 2006. The features are thought so that when observations from OOC patterns is present, the feature displays a different distribution than when there are only observations that follow the normal pattern.

To clarify how some of these features are generated a brief description will be given.

Kurtosis is calculated using the following equation,

$$k(x) = \sum_{i=0}^{n} \frac{(x_i \cdot \bar{x})^4}{n \cdot \sigma^4} \qquad \text{(eq.19)}$$

The weighted average is calculated using the following equation,

$$wavg(x,p) = \sum_{i=0}^{n} \frac{x_i \cdot p_i}{n} \quad \text{(eq.20)}$$

The mean distance between consecutive observations is calculated using the following equation,

$$rdist(x) = \left(\frac{\sum_{i=1}^{n} \sqrt{(x_{i+1} - x_i)^2 + (p_{i+1} - p_i)^2}}{n-1} \right) / \sigma \quad \text{(eq.21)}$$

The range of the slopes of the least squared lines fitted to different subsets of the window is calculated by using the following logic:

1) Separate the set of observations in 4 consecutive chunks of equal size
2) Join a set of chunks
3) Calculate the slope of the joined chunks
4) Repeat step 1-3 for all combinations of chunks
5) Calculate the range between the maximum and minimum slopes calculated

Once the features have been created to be able to see if they achieve the objective of differentiating windows which contain observations that come from OOC patterns, their distributions are plotted and observed.

To be able to train ML models data is needed. This data cannot be used for testing the performance of the models. Thus, DS2 is split into a training/validation set and an independent test set. This split is done considering how the data is structured, 5 blocks of 320 series of observations with all patterns. Thus, the training/validation set includes the first 4 blocks and the test set the last block.

This method of data pre-processing, as stated, is mainly done as part of the process of developing ML models. However, to allow for comparison between SPC methods and ML methods at the time of making predictions some steps must be performed for SPC as well. Briefly explained, based on how data pre-processing is done, the idea is that ML models will use the features generated from the windows to be trained, and predict which windows contain observations that come from OOC patterns. This means that to be able to compare results SPC methods must be performed on each window generated by step two of data pre-processing of the same test set used for ML models to predict with the same concept.

Another detail of importance to comment is that training the ML models with the feature generated from windows of 20 observations has another advantage; the ML model can be used for streaming data. This is because as long as the stream of data contains the last 20 observations, obtained by a sensor for instance, features can be extracted and then the window flagged alerting that the process may be OOC.

While conducting data pre-processing DS2 is saved. This is because the predictions for each method are essentially done on the windows created from DS1.

DS2 is saved after performing the first two steps of data pre-processing. DS2 is composed by three data frames:

1) WindowsScaled.csv
2) Label.csv
3) OriginalSeriesPattern.csv

Saved as CSV files with a point as a decimal separator and a tab as a column separator and joined in a folder which is compressed. The first data frame contains the windows of observations extracted from each series of observations. Each row represents a window, and the columns define the position (starting from base 0) of the individual observations within a window. The second data frame contains one column with the class of each window, 1 if any of the individual observations in the window come from a OOC pattern and 0 if all the individual observations in the window come from the normal IC pattern. The third data frame represents the pattern type of the original series of observation the window comes from.

It is also important to state that there are two versions of DS2, one scaled with step 1 and one non-scaled. The test split of the non-scaled dataset will be used to make predictions using SPC methods, and still allow for correct comparison of the predictions done with both ML models and SPC as stated previously.

5.1.3 SPC Automation

To automate the application of SPC methods, specifically I-MR control charts to detect assignable variabilities, and thus achieving one of the objectives of the project, a set of functions have been created using python with the help of NumPy library.

First a function to calculate the moving range and the control limits based on the equations presented in *table 1* was created. This was validated by calculating the control limits of a randomly generated series of observations manually and comparing the values obtained by the function.

After that, a set of functions that implement the rules defined in *figure 5* are created. This is done by manually creating a set of observations and checking the functioning of the rule, if the function works as expected then the next rule is tested, otherwise the function is modified until it works. This process can be seen in *figure 22* below.

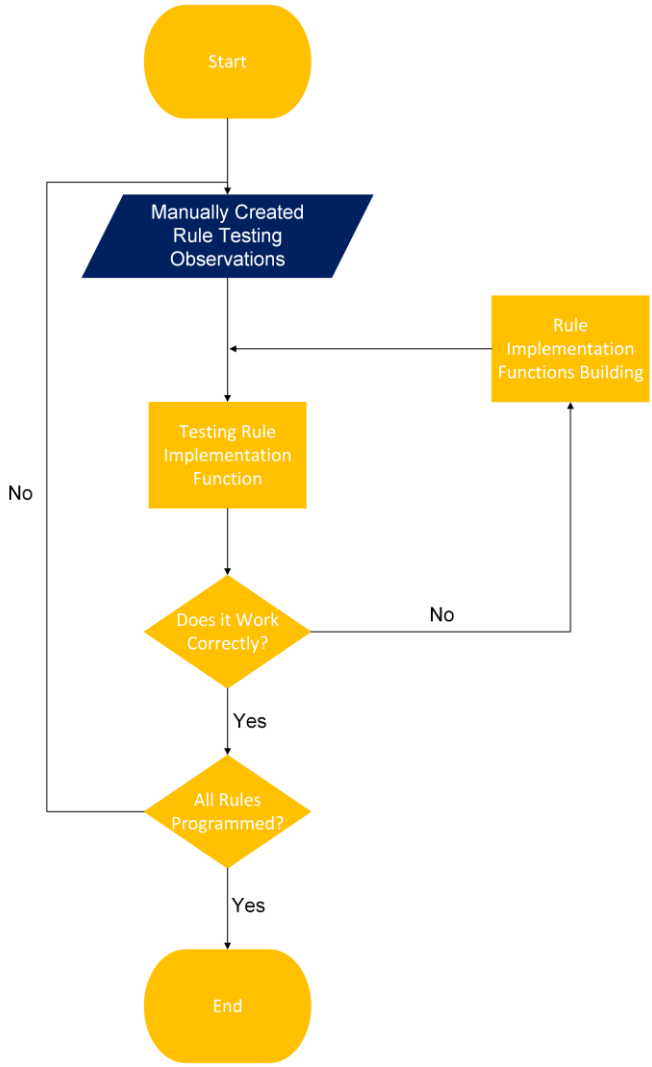


Figure 22. Process of creating and validating control chart rule functions. Source: personal collection

The manually created testing observations were created for each rule to test for two situations:

1) Fulfilling the rule definition

For instance, for rule 1 determined observations were manually created to be out of the 3σ control limit.

2) Incorrect identifications of rule fulfilling

For instance, if all observations are normal and the rule is not fulfilled that it does not flag observations as assignable variabilities.

Once this is done the SPC methods can be used to make predictions for any series of observations.

To make predictions the steps below are followed for each series of observations of the test split of DS22.

1) Given the first 21 observations of each series of observation in DS1 always come from the IC pattern, these are taken as historical values to calculate the control limits
2) For each window in each series of observations use the corresponding control limits, calculate the moving range values, and apply control chart rules to the individual values and the moving range values calculated
3) If a rule detects an OOC instance, then flag such window
4) Repeat steps 1-3 for each series of observations in the dataset
5) Create a data frame with all the prediction vectors and save it as a CSV file
6) Perform steps 1-5 for WE and Nelson set of rules

The CSV files are named:

1) TimeSeries1_WE_Classification.csv
2) TimeSeries1_Nelson_Classification.csv

These data frames both contain a single column which returns the predicted class of each corresponding window.

It is important to state that the predictions are done this way to be able to compare the SPC methods to the ML models. This is because, this way of implementing SPC, tests the observations in each window to see if they break a rule and then returns the prediction that such window contains observations that can be associated with an OOC instance.

5.1.4 ML Model Building and Algorithm Selection

Observations which come from OOC patterns as explained are inherently different to the rest of the data. Thus, features that come from windows will also differ from the normal data. What this means is that these windows can be considered anomalies. For this reason, the anomaly detection algorithm of isolation forest is chosen to create a possible model, to be able to learn anomaly windows behaviour and predict if a test observation is an anomaly. This would allow

to identify windows where the observations derived from OOC patterns, achieving the objective of this project.

On the other hand, given the data available is labelled, SVM can be used to explicitly learn which windows contained OOC observations. Thus, in this case, given the way the feature engineering is done, what SVM can do is predict whether windows of 20 observations contained values that were simulated with OOC patterns. Again this, intrinsically tackles the objective of being able to identify OOC situations.

5.1.5 Process of Isolation Forest Model Development

The isolation forest model is created with the help of sklearn library in python. The dataset that will be used to train the model is the training set DS23 seen in *figure 18* which represents the features extracted from each window. It is important to detail that given the model does not need the features to have the same scale, DS23 is not standardised (Tenraj, 2020).

Once the model is created it is validated using a 5-fold cross-validation routine. This can be done even if it is an unsupervised algorithm because the data is labelled, thus an evaluation method can be used (Nordby, 2021). This is done with the help of a function in the sklearn library. This function returns the following measures:

1) F1-scores for each evaluated split. This score is obtained contrasting the predictions made by the model and the actual labels of the validation sets
2) Mean and standard deviation of the F1-scores calculated

Next the model's hyperparameters can be tuned. Again, this is doable given it is possible to evaluate the model's predictions. To do this sklearn provides a function called RandomizedGridSearchCV, where the inputs are the model, an arbitrary parameter grid, the type of cross-validation strategy, the type of scoring and finally the number of iterations. This function choses a random combination of all the hyperparameters given by the parameter grid, and cross-validates the model. The cross-validation will return the measures previously detailed. This process is repeated for an arbitrary number of iterations of which the cross-validation measures are stored. Finally, a ranking of the measure is returned, and the model trained with the hyperparameters' that returns the best measure is chosen to be re-trained.

The reason of using randomized grid search is, given the large amount of hyperparameters to tune for this algorithm, to diminish the number of iteration possibilities, without greatly affecting the hyperparameter tuning. The intuition is that by testing random combinations of

all hyperparameters, it is bound to test a combination close to the best one that could be found exhaustively testing all combinations. This can be visualized better in the following *figure 23*,

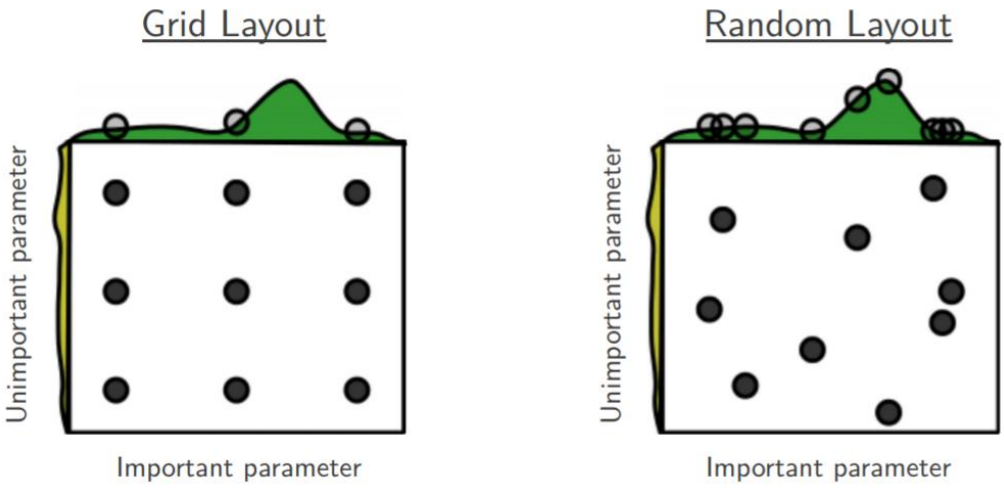


Figure 23. Random hyperparameter tuning, versus grid (exhaustive) hyperparameter tuning. Source: Worcester, 2019

The hyperparameters to be tuned while performing a 5-fold cross validation for the model are the following seen in *table 5*,

HYPERPARAMETER	DESCRIPTION	VALUE RANGE
Contamination	Anomaly score threshold to label observation as anomaly	$[0.1; 0.5]$
Maximum Features	Number of features to be drawn to train each isolation tree (as a ratio)	$[0.25; 1]$
Maximum Samples	Maximum subsample size	$[32; 1024]$
Number of Estimators	Number of isolation trees that make up the isolation forest	$[200; 2000]$

Table 5. Isolation forest hyperparameters. Source: personal collection

The ranking is done based on the mean and standard deviation of the set of cross-validation test's F1-scores for each hyperparameter combination. This is done to ensure best performance and avoid hyperparameters that cause overfitting.

Once the best set of hyperparameters is chosen the model is trained once again with the whole training dataset, and now it is ready to make predictions.

First however, it is saved using the dump function part of the joblib library. In this case, what the model stores is the hyperparameters and the structure of the isolation trees that form the isolation forest created during the training (scikit-learn developers, n.d.). To be able to load the model the load function of the joblib library can be used.

5.1.6 Process of SVM Model Development

The SVM model, is also developed with the help of the sklearn library and also uses DS23 training set to be trained. However, in this case DS23 needs to be standardised. The standardization step is needed as SVM requires all features to be centred around 0 and have a standard deviation of the same order (scikit-learn developers, n.d.).

Once the model is created it is validated, again by using a 5-fold cross-validation train/test strategy. This is done for the linear kernel and the RBF kernel to see what model returns better validation scores and thus be chosen to tune its hyperparameters.

Once this is done, the model's hyperparameters are tuned. This time the function used to perform this step is HalvingGridSearchCV. This function does two things differently than the one explained above,

1) Performs an exhaustive 5-fold cross-validations for each combination of hyperparameters
2) Performs the train/test strategy in sub-samples of the training data and iteratively choses the best combinations using bigger sub-samples

This function is chosen, given the fact SVM has fewer hyperparameters, thus it is worth to do an exhaustive grid search, and SVM takes longer to train so avoiding training the worst performing hyperparameter combinations on the whole dataset, improves time-efficiency for the part of the process.

In this case the hyperparameters to be tuned while performing a 5-fold cross-validation for the model are the following seen in *table 6*,

HYPERPARAMETER	DESCRIPTION	VALUE RANGE
C	Regularization parameter	$[0.001; 100]$
Gamma	Curvature weight of the decision boundary	$[0.001; 100]$

Table 6. SVM hyperparameters. Source: Personal collection

For each hyperparameter the combination of values tested are all values between the range in a logarithmic scale.

Again, after hyperparameter tuning the model is able to be used to make predictions. Moreover, it is also saved again using the dump function of the joblib library. In this case the support vectors obtained by minimizing eq.18 are saved.

5.1.7 ML Predictions

To make predictions with the trained ML models, an independent test set is fed to them as an input. This test set as seen in *figure 18* also goes through data pre-processing steps. Once the inputs are given to the models their predictions are stored as a CSV file. The stored files are named:

1) TimeSeries1_IForest_Classification.csv
2) TimeSeries1_SVM_Classification.csv

These data frames are structured in the same way as SPC predictions, a single column where the predicted class of each corresponding window is returned.

5.2 Results and Evaluation

In this chapter the results for each part are presented including some intermediate results from the development of the ML models. Most importantly the dataset created is commented as well as the predictions made by the models on the independent test set compared.

5.2.1 Synthetic Data Creation

A summary of the data created can be seen in *figure 24.* The blue points represent when the observations come from an IC pattern, whereas the red points represent when the OOC

pattern begins. As seen the first row only has in control patterns, the second row represents a cyclic pattern, which starts for observation number 25 in the first column, then 30 in the second column and so on. This is repeated for the systematic pattern, in the third row, the stratified in the fourth, the upward and downward trend in the fifth and sixth row and finally the upward and downward shift in the seventh and eighth row.

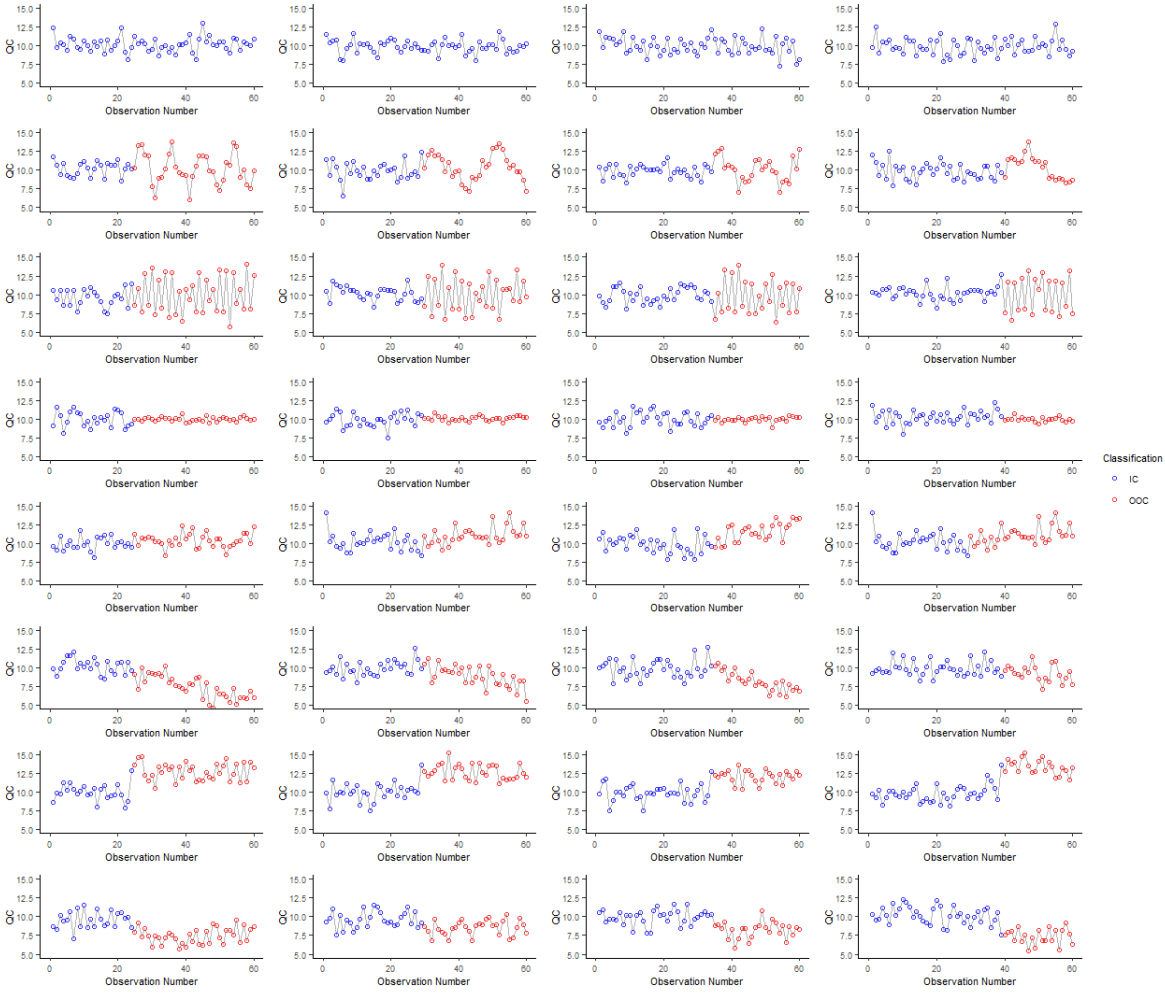


Figure 24. Summary of synthetic dataset. Source: personal collection

As explained the dataset is saved as a zip file DS1 and is composed of three CSV files. One contains 1600 series of observations of 60 observations, divided in 5 sub-blocks which have the same structure. Specifically,

- 0-39 Normal pattern
- 39-79 Cyclic pattern

- 80-119 Systematic pattern
- 120-159 Stratified pattern
- 160-199 Upward trend pattern
- 200-239 Downward trend pattern
- 240-279 Upward shift pattern
- 280-319 Downward shift pattern

The second one contains labels which label whether each observation comes from an OOC pattern or not. The third one contains the information about position of OOC pattern start and type of pattern.

5.2.2 Data Pre-processing

Data pre-processing, more importantly feature generation as a part of it, is a key part of being able to use ML algorithms to detect OOC instances. That is why, it is crucial to see if feature generation allows to differentiate windows with observations derived from OOC patterns or not. For that in following *figure 25* and *figure 26,* it is possible to view violin plots (box plot in case of discrete features) comparing the distribution of the generated features for windows with observations coming only from IC patterns and windows where at least one observation comes from OOC patterns.

Figure 25 and *26* display the features in the same order as they are presented in *table 4.* The boxplots show the distribution for discrete features which are, number of changes in the slope seen as the first boxplot and number of mean crossings seen second boxplot. The rest of the features given they are continuous are displayed using a violin plot.

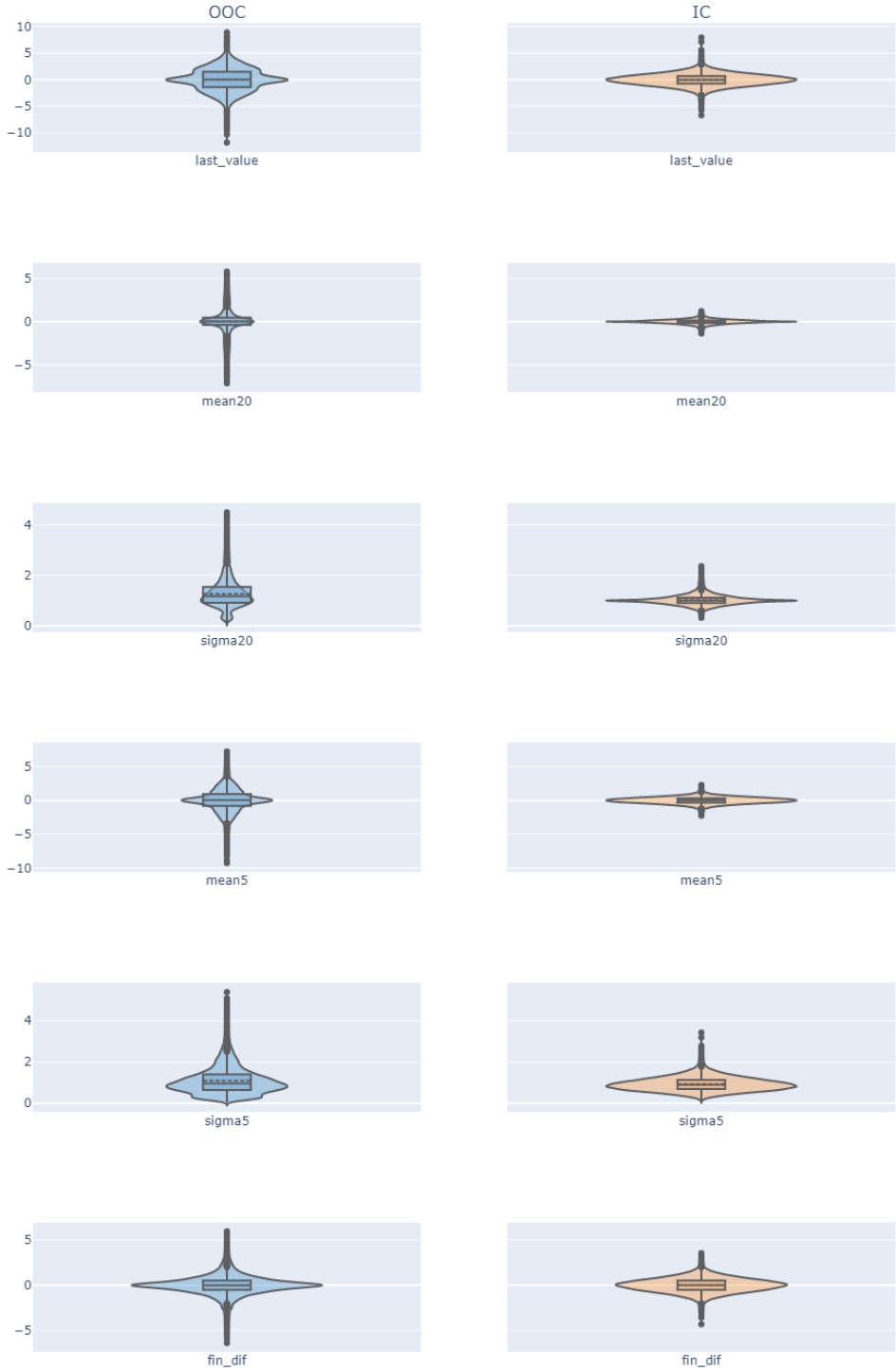


Figure 25. Features distribution for IC and OOC observations. Source: personal collection

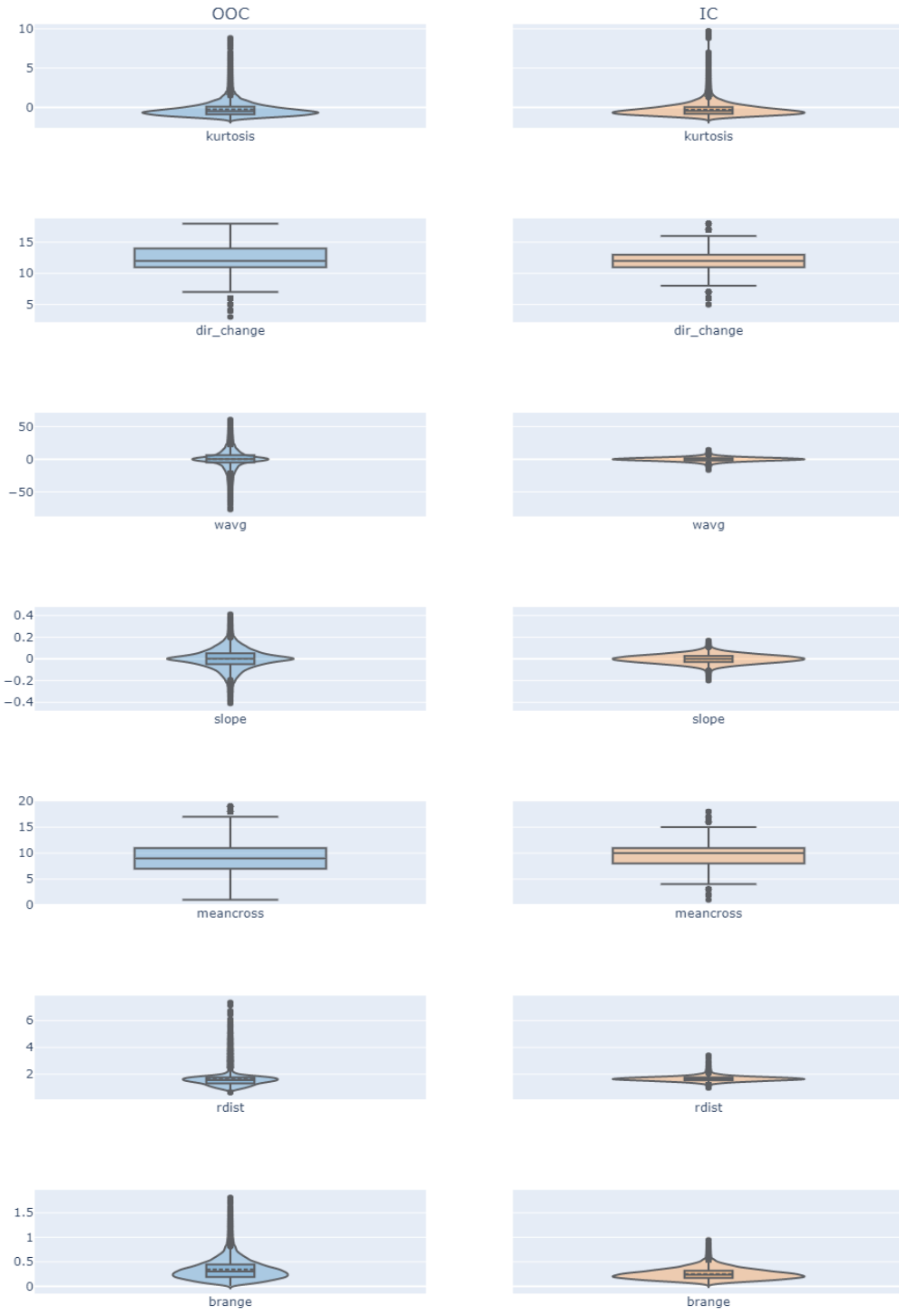


Figure 26. Features distribution for IC and OOC observations. Source: personal collection

As seen, there is some variation in feature distribution from windows with IC observations only and the ones with at least one observation from OOC patterns. However, it is also possible to see that there is a proportion of windows with observations from OOC patterns that is very similarly distributed to the IC data. The reason for this is due to the labelling method used while creating the synthetic observations. The labelling starts when the pattern changes from the IC pattern to one of the various OOC patterns, this however, doesn't mean that the OOC instance is readily detectable. Thus, even if the observation may be labelled as coming from an OOC pattern, it cannot be detected as one by any means because it acts as a normal observation. This concept is important as it will affect the way results are understood and the project evaluated.

Lastly, it must be commented that from data pre-processing a zip file is saved as DS21, which is composed of three CSV files with the following characteristics. The first contains 65600 windows of 20 observations. The second contains the label per window, which describes whether it contains at least one observation that come from OOC patterns, or all observations come from the normal pattern. The third one contains information about the corresponding series of observations pattern present in DS1 for each window.

The same is done for the non-scaled version of DS2, DS22, which has the same characteristics.

5.2.3 SPC Automation

Validation for SPC methods, as stated was done by manually testing each function created to automate SPC methods. The tests provided positive results for each function. For instance, for rule 1 and 2 the results can be seen in *figures 27* and *28* below respectively. In the figures the observations where the rule is fulfilled are flagged in red. As seen in *figure 27* both points outside of the 3σ control limits are flagged as fulfilling the rule. In *figure 28* the same occurs where the 9^{th} observation on the same side of the mean is flagged. In both cases when the rule is not fulfilled there are no observations flagged. The results for the rest of the rules can be found in the Appendix.

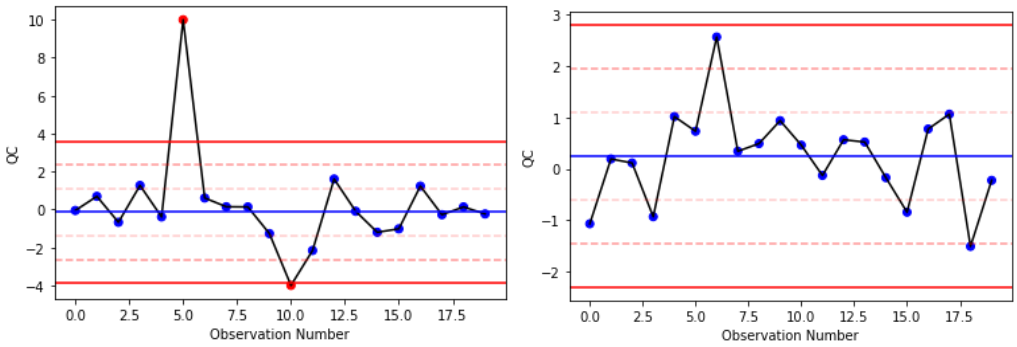


Figure 27. Test for function rule 1, above for rule fulfilling and below for rule not fulfilled. Source: personal collection

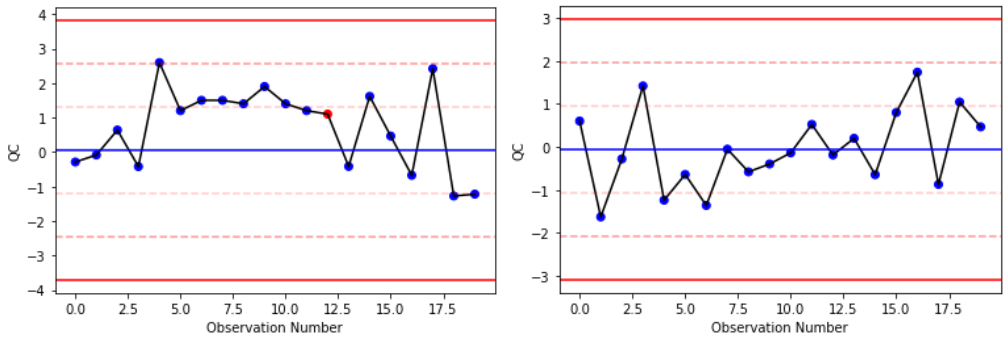


Figure 28. Test for function rule 2, above for rule fulfilling and below for rule not fulfilled. Source: personal collection

5.2.4 Process of Isolation Forest Model Development

The initial cross validation, where the hyperparameters are set by default as set in the sklearn function to create the isolation forest model (scikit-learn developers, n.d.), returns a mean F1-score of 0.773 and a standard deviation of 0.004. With these values it is possible to deduce that the model is somewhat able to make predictions correctly and that the training is performed to allow for generalizability. Moreover, the training data seems not to present any sign of greater significance in any portion, as the standard deviation of the scores is low.

Moving on to the hyperparameter tuning, RandomSearchCV function randomly evaluates the following combinations of hyperparameters with a 5-fold cross validation strategy,

COMB. NUMBER	CONTAMINATION	MAX FEAT.	MAX SAMP.	NUM. ESTIM.
1	0.5	1	32	1400
2	0.5	1	128	200
3	0.5	1	512	600
4	0.2	0.5	64	200
5	0.2	1	512	1200
6	0.5	0.5	1024	2000
7	0.5	1	32	1600
8	0.4	0.25	64	1400
9	0.4	0.25	1024	400
10	0.4	1	1024	1200

Table 7. Parameter combinations for isolation forest cross-validation. Source: personal collection

In the following *figure 29* it is possible to see the performance of the model with each combination of hyperparameters used when cross-validated, and thus the best combination of hyperparameters chosen based on it.

The eighth model is the chosen one as it has a higher F1-score and relatively low standard deviation.

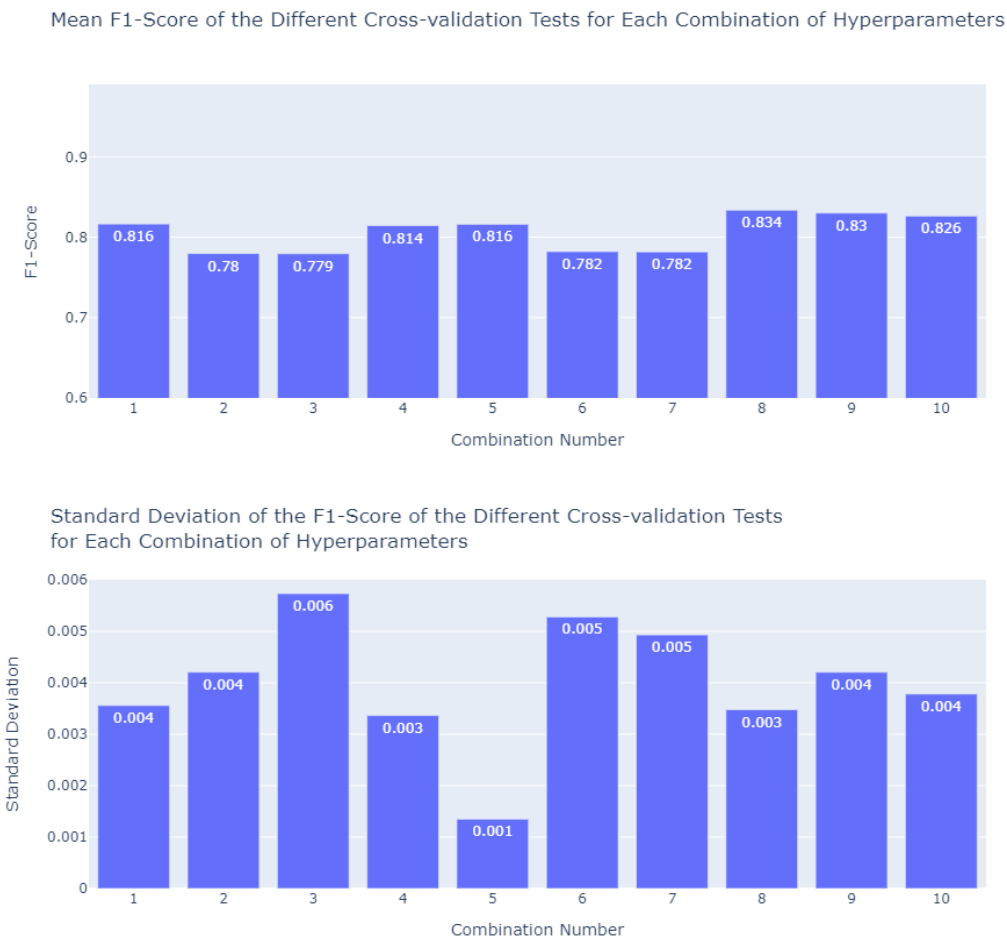


Figure 29. Isolation forest hyperparameter combination scores. Source: personal collection

5.2.5 Process of SVM Model Development

The result of the initial model validations, where the hyperparameters are set by default as set in the sklearn function to create SVM model (scikit-learn developers, n.d.), shows that the model with the RBF kernel has a better generalizability and ability to make predictions than the linear kernel model, as the mean F1-score for the RBF kernel model is 0.793 compared to 0.566 for the linear model. That is why the RBF kernel model is chosen for hyperparameter tuning.

The results obtained by the HalvingGridSearchCV function used to tune SVM model's hyperparameters are summarized in the *figure 30* below, where each graph represents the iterative test with each subsample, and the consecutive choice of the best hyperparameters to cross-validate with a bigger sub-sample.

Figure 30. SVM hyperparameter combination scores. Source: personal collection

In the first iteration there are no missing values, in fact the combinations with no bar associated, means that the F1-score for the model was of 0.

Thus, the final model's parameters chosen are:

HYPERPARAMETER	DESCRIPTION
C	10
Gamma	0.1

Table 8. Final SVM model hyperparameters. Source: personal collection

5.2.6 Predictions and Validation Metrics

ML models have been proposed, created, and trained, and SPC methods automated. Moreover, both have been used to predict whether a window of 20 observations from the testing set has observations that derive from OOC patterns or not deducing the process is OOC. Therefore, the models can be compared.

It is important to state again that the main goal is model comparison, given the data is labelled based on the start of an OOC pattern which does not mean that the specified observation is actually associated with an OOC process.

In *figure 31* it is possible to observe the performance of all models in regard to the whole data set. The graph shows the value of TP (bottom-right), TN (top-left), FP (bottom-left), FN (top-right) organised accordingly to the example of a confusion matrix in *figure 9.* The desired output would be only having TP and TN, thus minimizing the plot in the FP and FN sections. This is because it would mean that the predictions are all labelled in the correct way according to the real answer. With these criteria, when all graphs are compared the best model overall seems to be the isolation forest or SVM machine learning models, given the summed up red areas are smaller.

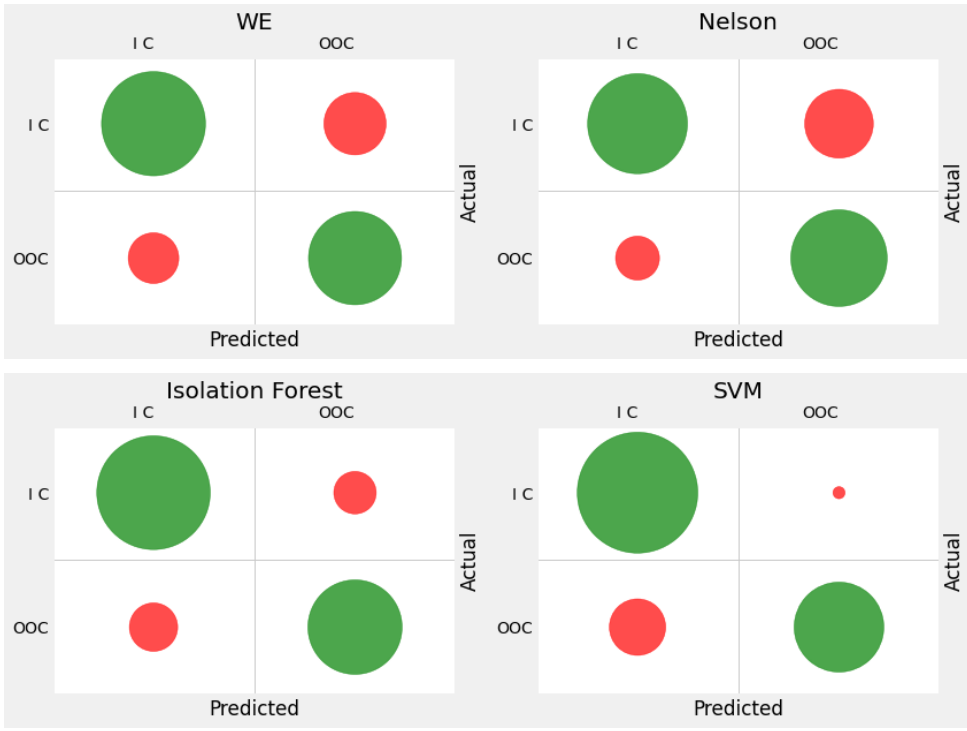


Figure 31. Confusion matrix of the model predictions compared to the synthetic data labelling for each model. Source: personal collection

Digging deeper, it is possible to notice that the SPC methods both have a lower number of true negatives, compared to the ML models. Moreover, the number of false positives is also higher than for the machine learning models. Therefore, showing that ML models have an improved behaviour at detecting when a process is IC. This can be due to the statistical distribution bias behind the construction of the SPC methods, and the way the adding more rules also increase the chances of detecting an IC process as OOC, as seen by comparing the number of FP for the WE and Nelson set of rules. On the other hand, machine learning learns the data's own distribution, diminishing the bias that comes from certain SPC model assumptions.

When it comes to detecting the OOC processes, the SVM model has the worst performance overall, even if it has the best performance dealing with IC process predictions. This is not the case with the isolation forest model as it has a similar value compared to the other SPC methods at correctly predicting a process being OOC, both of these statements can be seen in more detail by observing the precision and recall score for SVM in *table 9* which summarizes the respective precision, recall, and F1-scores for each model.

	PRECISION	RECALL	F1-SCORE
WE	0.68	0.75	0.71
NELSON	0.65	**0.80**	0.72
IFOREST	0.81	0.77	0.79
SVM	**0.93**	0.70	**0.80**

Table 9. Score summary for each model, where the best model per score is displayed in bold. Source: personal collection

Even if the models clearly have a ranking in terms of which is performs better, the variation in their ability to detect anomalies is relatively small. The same can be said in terms of incorrectly labelling OOC situations.

The SPC methods both act similar and have worse precision, given they tend to incorrectly label IC processes as OOC. However, they make that up by performing better with recall, as they are more sensible to detecting OOC situations. On the other hand, the machine learning models differ in their behaviour, when compared to each other. SVM is more has a much better precision but lacks recall of OOC instances. Isolation forest is more balanced, performing as well with predicting both classes. Comparing SPC methods to ML models, it is possible to see that SPC lack precision, but have better recall. Overall, however, when the F1-score is considered the ML models perform better than the SPC methods, and SVM model seems to be the better performing one. This, however, will depends on the situation and preferences of use as it is highly unbalanced when dealing with the different classes.

In *table 10* it is possible to see the recall value for each type of pattern within the dataset. This metric is the main one to analyse to be able to see out of the total number of windows which contain observations from OOC patterns, what proportion could the model identify correctly.

When observing the stratified results, overall, SPC methods and ML models seem to struggle to detect the pattern. This may be due to the fact for SPC it takes at least 15 points to detect an instance of the pattern, and for ML what can occur is the features' distribution is not heavily affected by these values, causing a masking error. Furthermore, it is clear that the WE rule set for SPC struggles the most given there is no direct rule to identify the pattern within the window. As for Nelson rules, they have the best recall for this pattern, followed closely by the isolation forest ML model. The SVM models does show a certain ability to detect this pattern but generally performing worse than isolation forest and Nelson rules.

RECALL PER PATTERN							
MODEL	**Stratified**	**Systematic**	**Cyclic**	**Upward Trend**	**Downward Trend**	**Upward Shift**	**Downward Shift**
WE	0.35	0.94	0.84	0.70	0.54	**0.94**	**0.96**
NELSON	**0.67**	**0.95**	**0.85**	**0.72**	**0.55**	**0.94**	**0.96**
IFOREST	0.64	0.92	**0.85**	0.58	0.53	0.93	0.94
SVM	0.59	0.81	0.78	0.46	0.45	0.91	0.90

Table 10.Recall summary for each model and pattern, where the best model is displayed in bold. Source: personal collection

Moving on to the systematic errors, SPC methods and ML models both tend to recall the patterns correctly. For ML this is probably due to the easier to learn discrimination that comes from features such as mean crossing and direction changes, which are more prominent for these types of patterns. For SPC, rule number 4 allows to easily detect the pattern. Another comment is that SPC methods and isolation forest have similar recall values, and the only model that performs differently is SVM. SVM does show a lower ability to identify windows with this pattern. This may be due to the fact SVM is more restrictive at classifying windows as anomalies as seen when commenting the general results.

In the case of cyclic patterns, they are identified with a fairly high recall. This is again probably due to the fact these patterns cause a major difference in feature distribution between IC and OOC instances, thus easing the discrimination or ML Models. Overall, SPC methods and isolation forest perform similarly, and the SVM model generally has a lower recall.

In terms of trends, they seem to be the hardest to detect patterns for all models. This makes sense as it is the type of pattern where the most masking is present due to its slow start, needing more OOC points in a window to actually detect the pattern. Nevertheless, when comparing SPC to ML models it is clear that for this type of pattern ML models are outperformed by SPC methods in terms of recall. However, this seems to be the case for the upward trend more than the downward trend. In fact, for the downwards trend the models have similar recall, which is worse than for the upward trend. This may be due to a lower average trend gradient for the downward trend data, limiting the ability to detect such patterns within the windows.

Finally, for shift patterns, again all models have high recall values, meaning they identify the majority of windows with this pattern. This is makes sense as shift can cause the highest variation in observations, allowing SPC methods to easily flag variability. Higher variation also causes higher distinction in the feature distribution from the windows with IC observations to the ones with OOC observations, allowing ML models to easily distinguish the windows with observations coming from OOC patterns.

From the results commented, the following things are clear:

1) ML models have a better balance between precision and recall
2) SPC methods are better at predicting anomalies but lack precision
3) Observation from systematic, cyclic and shift patterns allow for easier identification of OOC situations
4) Observation from stratified and trend patterns on the other hand difficult the task of correct identification of OOC situations
5) SPC methods and isolation forest have similar recall values
6) SVM generally has lower recall values

Therefore, the better model overall depends on the situation and the priorities of the user. In an industrial environment with fast-streaming data, the machine learning models may have some benefits. Mainly because they are more balanced thus avoid false alarms while still being able to detect anomalies. But also, because in reality due to the fact data streams are fast, if there is a pattern which is harder to detect by the ML model due to the fact it needs more OOC points in a window to detect the pattern, the velocity of the data input will contrast this issue.

Another important reflection is that these results have been obtained for the ML models given their hyperparameters have been tuned to achieve the highest F1-score. If the score was changed, their performance in the corresponding metric would probably be better. This thus allows for flexibility to adapt to the users' needs.

6 Financial Evaluation

The cost of the project can be estimated by adding up the cost of human and capital resources for the duration of the project. The human capital will include junior engineers, tasked with preparation, experimentation, memoir writing and presentation of the project. Moreover, it will consider senior engineer's reviewing time of the project. For both human resources the indirect costs which, include facilities maintenance and administration fees among other things, are accounted for. On the other hand, capital resources, will include the depreciation of the hardware used, the costs of the software used. All of the information is summarized in *table 11* below.

The dedicated time of the project is of 305 hours from the Junior Engineer as detailed in *figure 1* Gannt diagram, and 20 hours for presentation of the project are added. On the other hand, the time estimated for revision by a Senior Engineer is taken as a total of 40 hours. The unitary costs are obtained from *Glassdoor, 2022 and* dividing it by the total working hours in 2022 in Spain.

Human Resources	Unitary Cost (€/h)	Indirect Cost (€/h)	Hours of work (h)	Total Cost (€)
Junior Engineer	14,00	5,00	325	6.175,00
Senior Engineer	26,00	10,00	40	1.440,00
Senior Engineer	26,00	10,00	40	1.440,00
				9.055,00

Capital Resources	Total Cost (€)
Computer Depreciation (25% YoY)	100,00
RStudio	0,00
Anaconda	0,00
Office 365 license	20,00
	120,00

Total Amount without VAT	9.175,00
VAT (21%)	1.926,75
	€ 11.101,75

Table 11. Financial analysis summary

Adding the value added tax (VAT) the total cost of the project can be estimated to be 11.101,75 euros.

7 Conclusions

The project has been carried out successfully through its various stages and all objectives were achieved.

1. A representative dataset was simulated and created with 7 types of OOC patterns present in industrial environments. This allowed to train machine learning models and create a benchmark to evaluate both SPC methods and ML models' predictions.
2. Two different ML models, isolation forest and SVM, were chosen, created, trained following the correct development of a machine learning project, to be able to detect OOC instances within a quality control environment.
3. All models and automated SPC methods were used to detect when a set of observations were not following the IC control pattern. From this, the conclusion was reached that it is possible to create a machine learning model to identify OOC processes as framed by quality control environment.
4. Thanks to the correct use of all models to predict anomalies, from the same set of observations and comparison frame, it is possible to determine that machine learning models are a viable alternative to classical automated SPC methods to detect OOC processes. This is due to the fact their overall performance is of similar magnitude. It is true however, that the models do have some differences. More specifically:
    a. SPC methods are slightly better at detecting OOC processes than the isolation forest model, and better than the SVM model
    b. SPC methods tend to have more false alarms than both ML models, especially compared to SVM
    c. It can be stated that ML models have a better overall performance, to correctly classify both IC and OOC processes. However, for SVM this is because it has a very high precision which makes up for its lack of recall.

8 Bibliography

About ISO9001:2015. (n.d.). (ASQ) Retrieved May 4, 2022, from ASQ: https://asq.org/quality-resources/iso-9001#:~:text=ISO%209001%20is%20defined%20as,meet%20customer%20and%20regulatory%20requirements.

Acerta. (n.d.). *The Difference Between Machine Learning & SPC (and Why it Matters)*. Retrieved June 6, 2022, from acerta.ai: https://acerta.ai/blog/the-difference-between-machine-learning-spc-and-why-it-matters/#:~:text=Machine%20learning%20is%20not%20looking,SPC%20alone%20is%20not%20enough.

Agrawal, A. (2014, November 18). *Understanding Python pickling and how to use it securely*. Retrieved from synopsys: https://www.synopsys.com/blogs/software-security/python-pickling/#:~:text=Pickle%20in%20Python%20is%20primarily,transport%20data%20over%20the%20network.

Alcock, R. (1999, August 27). *Synthetic Control Chart Time Series Data Set*. Retrieved from UCI: https://archive.ics.uci.edu/ml/datasets/Synthetic+Control+Chart+Time+Series

American Society for Quality. (n.d.). *Quality 4.0*. Retrieved 07 26, 2022, from https://asq.org/quality-resources/quality-4-0

Anello, E. (2021, April 5). *Anomaly Detection with Isolation Forest*. Retrieved May 23, 2022, from betterprogramming: https://betterprogramming.pub/anomaly-detection-with-isolation-forest-e41f1f55cc6

Arpit. (2020, January 29). *Isolation Forest algorithm for anomaly detection*. Retrieved from towards data science: https://medium.com/@arpitbhayani/isolation-forest-algorithm-for-anomaly-detection-f88af2d5518d

Berwick, R. (n.d.). *Guide to Support vector*. (MIT) Retrieved May 16, 2022, from web.mit: https://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

Bhagwat, R., Abdolahnejad, M., & Moocarme, M. (2019). *Applied Deep Learning with Keras.* Packt. Retrieved from https://subscription.packtpub.com/book/data/9781838555078/6/ch06lvl1sec34/confusion-matrix

Bienkowski, J. (2019, September 17). *What is a Minimum Viable Model?* Retrieved June 12, 2022, from Synapsta-Blog: https://medium.com/synapsta-blog/what-is-a-minimum-viable-model-5d8166ec2e77

Bogdanovist. (n.d.). *How to choose a predictive model after k-fold cross-validation?* Retrieved from https://stats.stackexchange.com/users/10354/bogdanovist: https://stats.stackexchange.com/q/52277

Comissió de Recerca d'IQS. (2021, May 31). Normativa sobre la titularitat, protecció i explotació de drets de proprietat intel·lectual i Industrial. Barcelona, Spain: Consell de Direcció de IQS.

Cuadros, J. (2022, February 26). *Enseñanza del control estadístico de procesos – Materiales docentes*. Retrieved from https://jcuadros.github.io/emc/

Devaux, E. (2022, May 29). *Types of Synthetic Data and 4 Example of Real-life applications (2022)*. Retrieved from statice: https://www.statice.ai/post/types-synthetic-data-examples-real-life-examples#:~:text=Types%20of%20synthetic%20data%20and%204%20examples%20of%20real%2Dlife,)%2C%20and%20tabular%20synthetic%20data.

Dobilas, S. (2021, January 17). *SVM Classifier and RBF Kernel — How to Make Better Models in Python.* Retrieved from towards data science: https://towardsdatascience.com/svm-classifier-and-rbf-kernel-how-to-make-better-models-in-python-73bb4914af5b

Fletcher, T. (2008, December 23). *Support Vector Machines Explained.* Retrieved from https://www.csd.uwo.ca/~xling/cs860/papers/SVM_Explained.pdf

Free Software Foundation Inc. (2022, April 13). *Licencias*. Retrieved from GNU: https://www.gnu.org/licenses/licenses.es.html

Fumo, D. (2017, June 15). *Types of Machine Learning You Should Know*. Retrieved from towards data science: https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

Gauri, S. K., & Chakraborty, S. (2009). Recognition of control chart patterns using improved selection features. *Computer & Industrial Engineering, 56*(4), 1577-1588.

Gauri, S. K., & Chakraboty, S. (2006). Feature-based recognition of Control Chart Patterns. *Computers & Industrial Engineering*(51), 726-742.

Glassdoor. (n.d.). Retrieved from Sueldo Ingeniero Junior: https://www.glassdoor.es/Sueldos/ingeniero-junior-sueldo-SRCH_KO0,16.htm

Glassdoor. (n.d.). Retrieved from Sueldo Ingeniero Senior: https://www.glassdoor.es/Sueldos/ingeniero-senior-sueldo-SRCH_KO0,16.htm

GMcGlinn. (2008, August 12). *File:Poster - Control Charts for Nelson Rules.svg*. Retrieved from Wikipedia: https://commons.wikimedia.org/wiki/File:Poster_-_Control_Charts_for_Nelson_Rules.svg

Google. (2022, July 18). *Machine Learning Glossary*. Retrieved from https://developers.google.com/machine-learning/glossary

Hachimi, M. (2020). Multi-stage Jamming Attacks Detection using Deep Learning Combined with Kernelized Support Vector Machine in 5G Cloud Radio Access Networks. *IEEE ISNCC.* Montreal.

Hennign, C. (2022, February 1). *imbalanced classification vs anomaly detection*. Retrieved from stackexchange: https://stats.stackexchange.com/questions/562650/imbalanced-classification-vs-anomaly-detection

Hessing, T. (n.d.). *Rational Sub Grouping*. Retrieved from six sigma study guide: https://sixsigmastudyguide.com/rational-sub-grouping/

Hessing, T. (n.d.). *Statistical Process Control (SPC)*. Retrieved from six sigma study guide: https://sixsigmastudyguide.com/statistical-process-control-spc/

International Organization for Standardization. (2015, 09). Retrieved from ISO 9000:2015(es). Sistemas de gestión de la calidad — Fundamentos y vocabulario: https://www.iso.org/obp/ui/es/#iso:std:iso:9000:ed-4:v1:es

Jefatura del Estado de España. (2018, December 5). Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. España: Boletin Oficial del Estado.

Kantardzic, M. (2011). SVM. In M. Kantardzic, *Data Mining* (2nd ed., pp. 498-542). Hoboken: John Wiley and Sons, Inc.

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*, (pp. 413-422).

Lu, Z. (2019, July 1). *Printing lines with sliding windows in bash*. Retrieved May 24, 2022, from zhiganglu: https://zhiganglu.com/post/bash-print-sliding-windows/

Montgomery, D. C. (2013). *Introduction to Statistical Quality Control* (7th ed.). Uniter States: John Wiley & Sons Inc.

NCSS. (n.d.). *NCSS Statistical Software*. Retrieved from Individual and Moving Range Charts: https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Individuals_and_Moving_Range_Charts.pdf

NIST/SEMATECH. (2012, April). *e-Handbook of Statistical Methods*. Retrieved from https://www.itl.nist.gov/div898/handbook/pmc/section1/pmc14.htm

NIST/SEMATECH. (2012, April). *e-Handbook of Statistical Methods*. Retrieved from https://www.itl.nist.gov/div898/handbook/pmc/section5/pmc5436.htm

NIST/SEMATECH. (2012, April). *e-Handbook of Statistical Methods*. Retrieved from https://www.itl.nist.gov/div898/handbook/pmc/section2/pmc21.htm

NIST/SEMATECH. (2012, April). *e-Handbook of Statistical Methods*. Retrieved from https://www.itl.nist.gov/div898/handbook/pmc/section3/pmc32.htm

NIST/SEMATECH. (2012, April). *e-Handbook of Statistical Methods*. Retrieved May 14, 2022, from Table of Constants for Control Charts: http://www.world-class-quality.com/uploads/Download/20170801162202 11ControlChartConstantsandFormulae.pdf

NIST/SEMATECH. (2012, April). *e-Handbook of Statistical Methods*. Retrieved May 14, 2022, from https://www.itl.nist.gov/div898/handbook/pmc/section3/pmc322.htm

Nordby, J. (2021, March 18). *How to tune the hyperparameters for oneclass SVM while doing unsupervised learning?* Retrieved from stackexchange: https://stats.stackexchange.com/questions/402493/how-to-tune-the-hyperparameters-for-oneclass-svm-while-doing-unsupervised-learni

Oracle. (2022, August 07). Retrieved from ¿Que es el Big Data?: https://www.oracle.com/es/big-data/what-is-big-data/

Patel, H. (2021, August 30). *What is Feature Engineering — Importance, Tools and Techniques for Machine Learning*. Retrieved June 07, 2022, from towards data science: https://towardsdatascience.com/what-is-feature-engineering-importance-

tools-and-techniques-for-machine-learning-2080b0269f10#:~:text=Feature%20engineering%20is%20the%20process,design%20and%20train%20better%20features.

Pedregosa, F., Varoquax, G., & Gramfor, A. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.

Polzer, D. (2021, May 21). *A Comprehensive Beginner's Guide to the Diverse Field of Anomaly Detection*. Retrieved May 4, 2022, from towards data science: https://towardsdatascience.com/a-comprehensive-beginners-guide-to-the-diverse-field-of-anomaly-detection-8c818d153995#36dd

QMS Definition. (n.d.). (mastercontrol) Retrieved May 04, 2022, from mastercontrol: https://www.mastercontrol.com/quality/qms/definition/

Raj, R. (n.d.). *Classification of Machine Learning Models*. Retrieved from enjoy algorithms: https://www.enjoyalgorithms.com/blog/classification-of-machine-learning-models/

Ripley, B. D. (2009). Pattern Recognition and Neural Networks. *Cambridge University Press*, Glossary.

Rubiales, A. (2020, June 26). *¿Qué es Underfitting y Overfitting?* Retrieved May 15, 2022, from medium: https://rubialesalberto.medium.com/qu%C3%A9-es-underfitting-y-overfitting-c73d51ffd3f9

Savan, P. (2017, May 3). *Chapter 2 : SVM (Support Vector Machine) — Theory*. Retrieved May 15, 2022, from medium: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

scikit-learn developers. (n.d.). *Cross-validation: evaluating estimator performance*. Retrieved May 15, 2022, from scikit-learn: https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation-iterators

scikit-learn developers. (n.d.). *Pre-processing Data*. Retrieved June 11, 2022, from scikit-learn: https://scikit-learn.org/stable/modules/preprocessing.html#standardization-or-mean-removal-and-variance-scaling

scikit-learn developers. (n.d.). *sklearn.ensemble.IsolationForest*. Retrieved May 23, 2022, from scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

scikit-learn developers. (n.d.). *sklearn.model_selection.GridSearchCV*. Retrieved from sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

scikit-learn developers. (n.d.). *Support Vector Machines*. Retrieved June 12, 2022, from scikit-learn: https://scikit-learn.org/stable/modules/svm.html

Shewart, W. A. (1923). *Economic Control of Quality of Manufactured Product.* New York: D. Van Nostrad Company Inc.

Tenraj, P. (2020, June 22). *Do Decision Trees Need Feature Scaling*. Retrieved June 10, 2022, from towards data science: https://towardsdatascience.com/do-decision-trees-need-feature-scaling-97809eaa60c6

Western Electric Company. (1956). *Statistical Quality Control Handbook* (1st ed.). Indiana, Indianapolis: Western Electric Co.

Wikipedia Contributors. (2022, July 18). *Anomlay Detection*. Retrieved August 21, 2022, from Wikipedia The Free Encyclopedia: https://en.wikipedia.org/w/index.php?title=Anomaly_detection&oldid=1099088370

Wikipedia Contributors. (2022, August 12). *Feature Engineering*. Retrieved from Wikipedia The Free Encyclopedia: https://en.wikipedia.org/wiki/Feature_engineering

Wikipedia Contributors. (2022, July 13). *Nodes*. Retrieved from Wikipedia The Free Encyclopedia: https://en.wikipedia.org/wiki/Node_(computer_science)

Wikipedia Contributors. (2022, February 11). *Synthetic Data*. Retrieved August 21, 2022, from Wikipedia The Free Encyclopedia: https://en.wikipedia.org/w/index.php?title=Synthetic_data&oldid=1071178758

Willimitis, D. (2018, December 18). *The Kernel Trick*. Retrieved May 22, 2022, from towards data science: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

Worcester, P. (2019, June 5). *A Comparison of Grid Search and Randomized Search Using Scikit Learn*. Retrieved June 10, 2022, from towards data science: https://medium.com/@peterworcester_29377/a-comparison-of-grid-search-and-randomized-search-using-scikit-learn-29823179bc85

Xie, Y. (2019). A Docker Container Anomaly Monitoring System Based on Optimized. *IEEE Transactions on Cloud Computing*, 1-1.

Xu, J., Lv, H., Zhuang, Z., Lu, Z., Zu, D., & Quin, W. (2019). Control Chart Pattern Recognition Method Based on Improved One-dimensional Convolutional Neural Network. *International Federation of Automatic Control, 52*(13), 1537-1542.

Zhao, Y. N. (2019). PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of machine learning research (JMLR), 20*(96), 1-7. Retrieved 06 2022, 07, from https://pyod.readthedocs.io/en/latest/benchmark.html

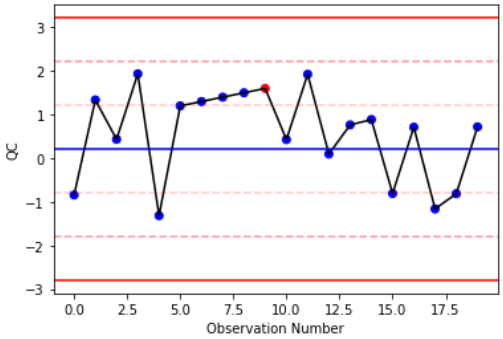
9 Appendix

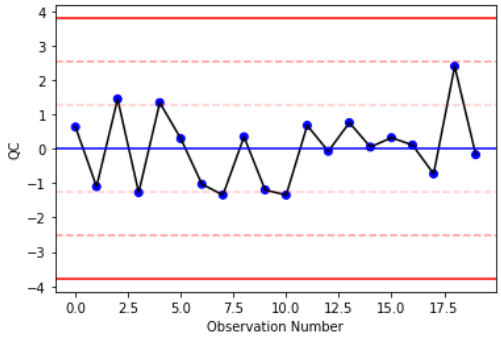
1: Code Repository

https://github.com/marcmacori/Anomaly-Detection-for-SPC

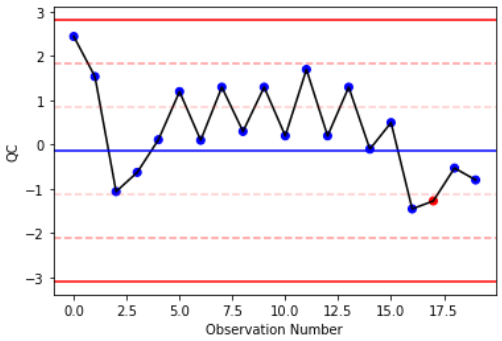
2: SPC Method Function Rule Validation

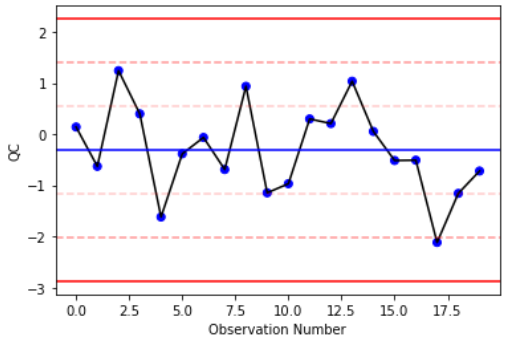
Rule 3



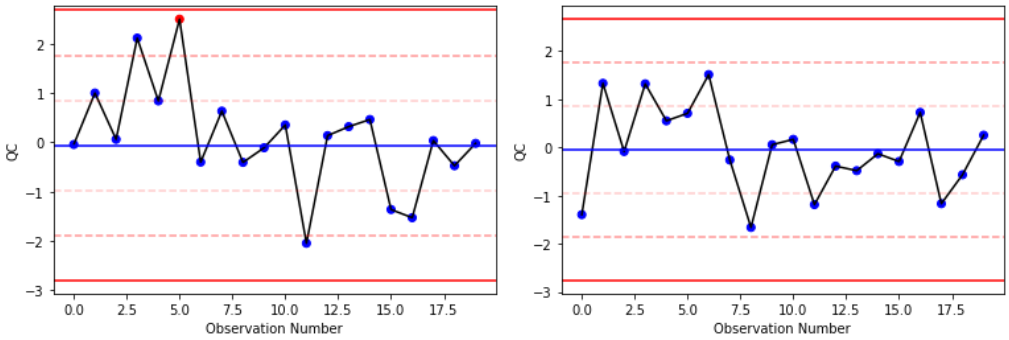


Rule 4

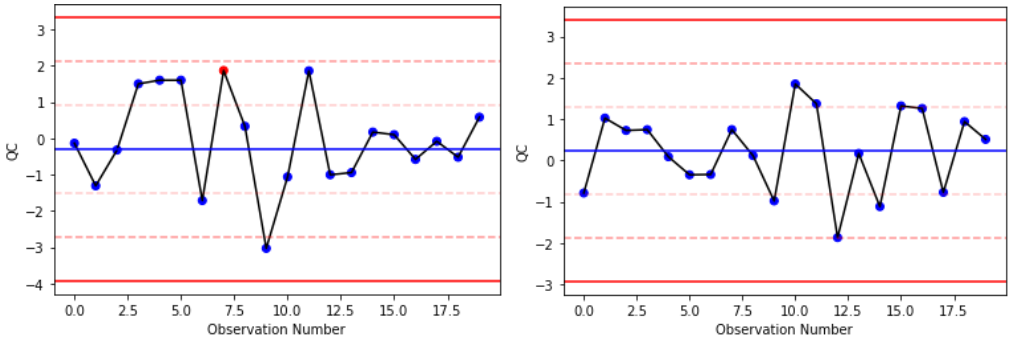




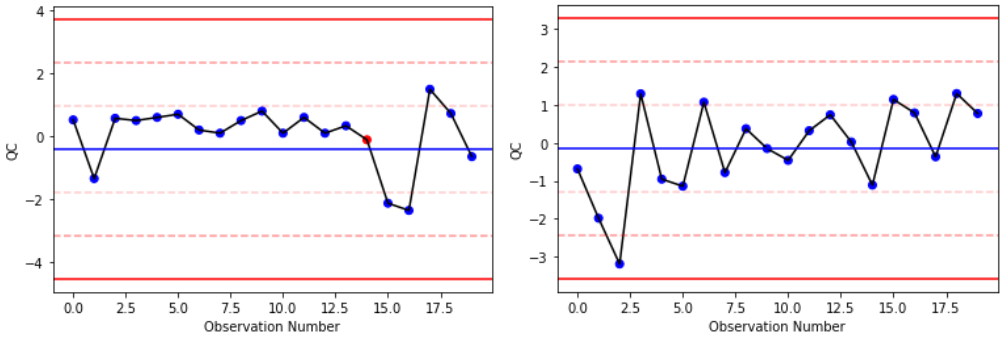
Rule 5



Rule 6



Rule 7



Rule 8