UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Gestionale

Modello Stacking di apprendimento automatico per la gestione della qualità (OQM) e la previsione dei difetti nell'Industria 4.0

A Stacking Machine Learning Model for Operational Quality Management (OQM) and Defect Prediction in Industry 4.0

Candidate:
Matteo Ciarrocchi

Advisor:
Dr. Sara Antomarioni

Academic Year 2025-2026

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Gestionale

Modello Stacking di apprendimento automatico per la gestione della qualità (OQM) e la previsione dei difetti nell'Industria 4.0

A Stacking Machine Learning Model for Operational Quality Management (OQM) and Defect Prediction in Industry 4.0

Candidate:
Matteo Ciarrocchi

Advisor:
Dr. Sara Antomarioni

Academic Year 2025-2026

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Gestionale
Via Brecce Bianche – 60131 Ancona (AN), Italy

A chi, nonostante tutto,
non smette di crederci

Abstract

Nel contesto dell'Industria 4.0, l'Operations Quality Management (OQM) sta attraversando una profonda trasformazione grazie all'adozione di tecniche di Data Analytics e Machine Learning (ML), strumenti ormai indispensabili per evolvere da un controllo qualità reattivo a un monitoraggio predittivo e proattivo. La presente tesi si pone l'obiettivo di analizzare criticamente l'impiego di tali tecnologie, sviluppando una soluzione algoritmica avanzata in grado di superare le sfide intrinseche dei dati di produzione.

Il lavoro di ricerca è strutturato in tre fasi principali. Inizialmente, a valle di un inquadramento teorico sull'OQM data-driven, è stata condotta un'approfondita Analisi Esplorativa (EDA) e di Feature Engineering su un dataset manifatturiero open source. Questa fase ha consentito di identificare i principali driver operativi della difettosità e di ingegnerizzare nuovi KPI aziendali.

Successivamente, la ricerca si è concentrata sulla replica sperimentale e sull'analisi critica di quattro studi di riferimento presenti in letteratura. Tale passaggio è stato cruciale non solo per stabilire una baseline prestazionale oggettiva, ma anche per far emergere e superare specifiche criticità metodologiche degli approcci esistenti, quali il rischio di Data Leakage o la perdita di informazioni derivante da tecniche di undersampling.

Per superare i limiti individuati, nella terza fase è stata progettata, sviluppata e testata un'architettura predittiva innovativa basata su uno Stacking Ensemble Eterogeneo che combina la robustezza del Random Forest, la precisione dell'XGBoost e l'approssimazione non lineare di una Rete Neurale (Multi-Layer Perceptron), orchestrati da un Meta-Learner logistico. Per garantire l'assoluta validità statistica, l'intera pipeline è stata incapsulata in una Nested Cross-Validation che integra la tecnica di sovracampionamento SMOTE per la gestione del Reverse Imbalance. L'architettura è stata infine potenziata da un algoritmo di Threshold Tuning dinamico, calcolato massimizzando il Matthews Correlation Coefficient (MCC) esclusivamente sul set di addestramento.

I risultati sperimentali confermano l'efficacia superiore dell'approccio proposto, distanziandosi dal paradosso dell'Accuratezza, pertanto il sistema ha registrato metriche al di sopra degli standard esistenti. Il confronto con lo Stato dell'Arte certifica che l'integrazione di validazioni rigorose e strategie cost-sensitive permette di ottenere un modello altamente affidabile, in grado di soddisfare i severi requisiti qualitativi dello Zero-Defect Manufacturing.

Contents

List of Figures

List of Tables

Chapter 1

Introduzione

Operations Quality Management (OQM) represents the set of practices, methodologies and tools aimed at ensuring that an organization's operational processes produce outputs that comply with the required quality requirements and customer expectations, all this to achieve a competitive advantage through the minimization of errors [1]. It is configured as a systemic approach that combines management principles, digital tools and advanced analytical techniques. Unlike traditional quality control, which is mainly focused on the detection and correction of defects in the final product, OQM focuses on the entire operational cycle. This approach integrates organizational, technical and cultural aspects, aiming not only at the efficiency of the individual process, but also at the overall reliability of the production system. With the advent of Industry 4.0 [2], OQM has taken on an even more strategic role as new technologies (Internet of Things (IoT), smart sensors, cyber-physical systems and cloud platforms) now make it possible to collect an unprecedented amount of data, from production processes and the supply chain, which, after subsequent analysis, allow for real-time visibility of operations, detecting anomalies before they generate defects, and optimizing the planning of activities. The integration of OQM with Industry 4.0, therefore, shifts the focus from ex-post control to predictive and proactive monitoring, strengthening companies in markets characterized by high complexity and variability. In this context, Data Analytics (DA) and Machine Learning (ML) techniques play a central role: through advanced statistical models and machine learning algorithms, they make it possible to extract significant patterns from data, develop defect prediction systems, estimate the risk of non-compliance and identify the most critical process variables. For example, supervised algorithms can be used to classify compliant and non-compliant products based on sensory data, while unsupervised algorithms allow the detection of clusters of anomalies or non-evident correlations; the use of time-series techniques and neural networks allows the anticipation of variations in operating parameters. Despite the opportunities offered by Industry 4.0 and advanced analytics tools, the implementation of a fully integrated Total Quality Management (TQM) still encounters some limitations: the management of large volumes of heterogeneous data (big data) requires robust technological infrastructures and specialized analytical skills not always present in organizations; the implementation of ML algorithms can be hampered by issues related to data availability and quality: incomplete, noisy, or unrepresentative datasets reduce the reliability of models. In addition to this, there are organizational and cultural criticalities, such as resistance to change, the difficulty of integrating new tools with legacy systems and the need to ensure transparency and interpretability of the solutions adopted. The full realization of a data-driven TQM remains an open challenge, which requires the combination of different types of solutions.

1.1 History of Operations Quality Management

Quality management has undergone a significant evolution over time, reflecting technological, organizational and cultural changes in production systems. The first forms of quality control,

developed between the twenties and forties of the twentieth century, were carried out by figures dedicated to inspection and focused essentially on the finished product, on which the main objective was to identify defects already present, in a strictly reactive logic that did not allow intervention upstream of the process. Starting from the fifties, thanks to the contributions of scholars such as Walter A. Shewhart, W. Edwards Deming and Joseph Juran, the statistical process control (SPC) approach was established, which allowed a systematic approach to monitoring the variability of processes, shifting quality from a mere ex-post verification to a prevention activity [3]. In the eighties and nineties, the perspective was further expanded with the concept of Total Quality Management (TQM), which placed quality as a goal shared by the entire organization, including not only production processes, but also management, decision-making and related processes. Tools such as the PDCA cycle, benchmarking and continuous improvement (Kaizen) have helped to instill a culture of quality extended to all levels of the company. At the same time, in the same years, the Six Sigma methodology took hold, which introduced a rigorous and quantitative approach to reducing process variability, structured around the DMAIC (Define, Measure, Analyze, Improve, Control) cycle. In the early 2000s, the progressive spread of the Lean philosophy led to a strong integration between waste reduction and statistical quality methodologies, giving rise to the so-called Lean Six Sigma. A further step forward was marked by the introduction of international standards, including the ISO 9000 standards (since 1987), ISO 13485 for medical devices and models of excellence such as the EFQM or the Baldrige Model. These tools have made it possible to standardize and disseminate consolidated quality management practices globally, laying the foundations for a common language and integration with other management systems (safety, environment, social responsibility). With Industry 4.0, quality management has taken on a profoundly different connotation; the introduction of digital technologies such as the Internet of Things, cyber-physical systems, cloud platforms and advanced sensors has made it possible to collect and analyze large amounts of data from production processes. This new context has shifted the focus from ex-post control to real-time monitoring and prediction of phenomena. Recent literature highlights how Data Analytics and Machine Learning tools are becoming central to OQM: Luckow et al. (2018) evaluated architectures, models, and implementation challenges related to the use of deep learning techniques in automotive manufacturing quality and logistics, focusing on computer vision problems [4]; Nalbach et al. (2018) proposed a machine learning-based system that links quality assurance to the design phase in order to support predictive quality [5]; Wuest, Irgens, and Thoben (2014) have proposed a clustering and supervised learning approach to manage the complexity and high dimensionality of product state data [6]; Lieber et al. (2012) investigated how data mining techniques and intelligent machine telematics can be used to predict internal quality problems of intermediate products [7]. The perspective of quality has been extended beyond the boundaries of production, to involve the entire supply chain. Issues such as supplier reliability, punctuality of deliveries and traceability of materials are increasingly considered decisive to ensure the overall quality of the final product. In this context, machine learning algorithms have been applied to predict delays, evaluate supplier performance, and analyze supply chain operational risks, as in the study proposed by Steinberg, F. (2023) in which multiple regression/classification algorithms are compared to predict the severity of supplier delays [8]. Despite this progress, the literature still shows some critical issues: many studies are focused on specific industrial contexts, such as automotive or electronics, and are difficult to generalize to other sectors, in particular to small and medium-sized enterprises; a significant part of the research is based on experimental or simulated data, which does not fully reflect the complexity of real systems. A further limitation concerns the "black-box" nature of some algorithms, such as deep neural networks which, while guaranteeing high accuracy, pose difficulties in terms of interpretability and therefore managerial acceptance. Finally, the integration of the OQM with sustainability and

resilience objectives, which are essential for companies today, still remains little explored. Today, therefore, quality is no longer just compliance with requirements, but a real dynamic system of organizational learning, which however still has open challenges. In light of these considerations, the present thesis aims to contribute to the ongoing debate by critically analyzing the existing models of data-driven OQM, replicating a reference study present in the literature and applying methodologies similar to an open-source dataset. The goal is to verify the effectiveness of Data Analytics and Machine Learning techniques in predicting quality, while offering operational indications that can also be useful in real business contexts.

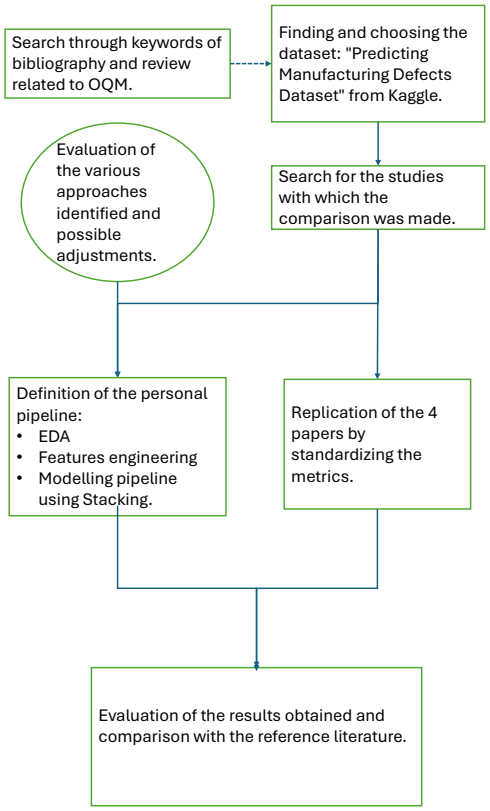

Figure 1.1: Block diagram illustrating the structure of the thesis

The block diagram illustrated in Figure 1.1 summarizes the logical architecture and methodological flow that guided the development of this thesis. The research path is structured into sequential and parallel phases, designed to ensure scientific rigor and objective validity of the results obtained. The research begins with the definition of Operations Quality Management (OQM) in the context of Industry 4.0; in this preliminary phase, the fundamental requirements of modern quality control are established, characterized by the transition to a proactive approach oriented towards Zero-Defect Manufacturing. This theoretical framework directly motivates the choice of the practical case study: the acquisition of the Predicting Manufacturing Defects Dataset from the Kaggle platform, characterized by a marked Reverse Imbalance, constitutes the ideal testbed for testing algorithmic solutions under industrial stress conditions.

Starting from the dataset, the study interfaces with recent scientific literature, isolating four reference articles that have addressed the same predictive task. At this point, the methodological flow splits to ensure an objective analysis:

- **Experimental Replica:** The investigation does not simply accept the authors' declared results, but proceeds with a rigorous replication of the algorithms in a controlled environment; through standardization of evaluation metrics, an objective and comparable performance baseline was recreated.
- **Evaluation and Adjustments (Circular Branch):** In parallel, as highlighted by the circular block, a critical analysis of the literature methodologies was conducted, which allowed us to highlight systemic gaps in the reference papers, such as exposure to the risk of data leakage, the ineffectiveness of destructive techniques, and the inability to manage probabilistic distortions through rigid decision thresholds.

The critical issues that emerged from the evaluation phase provided the guidelines for designing the innovative solution, whose implementation was divided into three operational sub-phases:

- **EDA (Exploratory Data Analysis):** A univariate and bivariate analysis that allowed us to validate the absence of multicollinearity and to identify the primary predictors of defectiveness.
- **Feature Engineering:** The translation of raw data into high-value management KPIs, enriching the information space available to algorithms.
- **Modelling Pipeline:** The development of the thesis's algorithmic core. A Heterogeneous Stacking Ensemble architecture was engineered that combines the robustness of bagging, the precision of boosting, and the nonlinear approximation of neural networks, orchestrated by a logistic meta-learner. The entire architecture was encapsulated within a nested validation framework, in synergy with the SMOTE technique, ensuring complete segregation of the test data.

The methodological path converges in the final block, where the results of the "Personal Pipeline" are directly compared with the baselines obtained from replicating the studies; the framework's final output confirms the achievement of the research objective: the performance superiority of the personal pipeline over all reference papers. As summarized in the diagram, the scientific value of this achievement lies not only in achieving a recall rate greater than 99% and an average MCC of 0.84, but above all in having obtained these metrics through a model structurally free from data leakage and immune to probabilistic biases thanks to the implementation of an innovative adaptive threshold tuning algorithm.

Chapter 2

State of the Art

In this thesis we used the open-source dataset available on Kaggle called "Predicting Manufacturing Defects Dataset" [9], which will be analyzed later. From the DOI we found all the articles and studies that use it in the context of the OQM, reviewing the most widespread and relevant algorithms used, of which the operating principles, typical applications, advantages, limits and potential in the context of Industry 4.0 will be described. Supervised approaches, unsupervised approaches and the most recent methods of Deep Learning will be analyzed; the aim is to provide a critical overview of the current techniques, thus laying the foundations for understanding the methodological choices adopted in the subsequent research work.

2.1 Literature regarding data-driven techniques in OQM

k-Nearest Neighbors (KNN)

The first method that is analyzed is the k-Nearest Neighbors (KNN) [10], one of the most immediate and intuitive supervised learning algorithms that do not build a complex mathematical model or "learn" an explicit function during an intensive training phase but is based on the simple idea that similar examples produce similar responses. In practice, to classify (or predict a value, in the case of regression) a new sample, the KNN first stores all the labeled samples of the training dataset, then, when a new example arrives to be classified, calculates the distance (typically Euclidean, but also Manhattan, Mahalanobis, etc.) between the new example and all the examples in memory; it selects the K closest examples (the "k-neighbors") based on that distance, aggregates the neighbor labels; for classification, it takes the majority class (majority vote), while for the regression it averages the values and returns the aggregate class or value as the prediction.

Two aspects strongly influence the performance of KNN: the choice of k, since for small values it gives very "local" and noise-sensitive decisions, while large values make decisions more stable but less able to capture complex class boundaries; the representation and scaling of features, because the algorithm is based on distances, variables with larger scales or irrelevant characteristics can dominate, so the normalization, selection or weighting of features are almost always necessary steps.

The KNN in the field of Operations Quality Management finds different types of applicability regarding the prediction of defects in products; in the study by Muhammad David and Muhammad Azka Firdaus [11], they test distances:

$$d_{euclid}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.1}$$

$$d_{manhattan}(x, y) = \sum_{i=1}^{n}|x_i - y_i| \tag{2.2}$$

$$d_{minkowski}(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \tag{2.3}$$

The aim is to predict defects in the production process, and the best accuracy value is achieved with the Euclidean distance, equal to 86.41%.

"Fault Detection Using the k-Nearest Neighbor Rule for Semiconductor Manufacturing Processes" introduces an innovative method of Fault Detection based on the k-Nearest Neighbors (FD-kNN) rule applied to semiconductor manufacturing processes, with the aim of improving operational quality and reducing scrap and downtime. The algorithm, adapted to work in unsupervised mode, uses only process data under normal conditions to build a reference model and calculate anomaly thresholds based on the local distance between samples, making it more effective than traditional PCA methods in detecting failures in environments characterized by nonlinearity, multimodality and temporal variability [12].

Naive Bayes

Continuing with classification algorithms, Naive Bayes (NB) [13] is a simple but surprisingly effective probabilistic approach, based on Bayes' theorem and the hypothesis of conditional independence of features given the value of the class. In operational terms, the model estimates the posterior probability of each class "C_k" for an observation described by an attribute vector "x" through Bayes' formula, assigning the class with the highest posterior probability:

$$P(C_k|x) = \frac{P(x|C_k)\, P(C_k)}{P(x)} \tag{2.4}$$

For continuous data, parametric versions (Gaussian Naive Bayes) are commonly used, while for discrete data, frequencies and Laplace estimates are used to avoid zeros. Despite the radical simplification of the assumption of independence, it has been shown that Naive Bayes can perform competitively; Domingos & Pazzani [14] and Rish [15] explain the conditions and reasons why NB tends to work well even when the independence condition is violated, and in many cases its accuracy is comparable to that of more complex methods, especially when the number of examples is limited or when the relationships between variables are not so complex.

From a computational point of view, it is extremely light; in fact, the training essentially consists of counting occurrences and estimating the few variances and means, and the prediction requires the calculation of conditional probability products; these probabilistic outputs can be used as scores for operational decisions and are often used as baselines of larger systems for quality control and line supervision.

In fact, typical Naive Bayes applications involve tasks where it is necessary to quickly classify process outcomes or predict the presence of defects from sensor-derived features, acceptance tests, or process indicators. A significant application work in the industrial field is that of Zhang et al. [16], which proposes variants of Naive Bayes for bearing fault diagnosis: in the article, the authors combine feature selection and pre-processing techniques to reduce the correlation between variables and improve the discriminating capacity of NB, showing that, suitably adapted, NB can achieve highly competitive performance in industrial condition monitoring scenarios; this is representative of how it is integrated into OQM pipelines.

However, there are practical limitations: the assumption of independence is often violated in process data, since sensory signals and physical measures are typically correlated, and this can degrade the quality of estimated probabilities and predictions; for this reason, various strategies are

adopted in industrial practice, such as transformation and selection of features to reduce dependency, use of more sophisticated variants or the use of NB as a hybrid component [16].

Logistic Regression

The next algorithm is logistic regression, a statistical reference model for the prediction of binary outcomes and often the first disciplinary approach adopted when the goal is to estimate the probability of an event occurring; its foundations go back to the classical developments of binary sequence analysis [17] and the operational and practical treatment has been consolidated in reference texts such as *Applied Logistic Regression* [18], still the standard manual for the correct application of the method.

Logistic regression models the probability "p" as a function of a vector of predictors "x" via the logistic function; it is conventionally written in terms of log-odds (logarithm of odds):

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{2.5}$$

from which the explicit form of probability is derived:

$$p = \frac{1}{1 + e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_n x_n)}} \tag{2.6}$$

The model is trained by estimating the coefficients "β_i" so that the predicted probabilities are as close as possible to the observed values; this is done by optimizing an error function, which measures the distance between predictions and real observations. Basically, the process looks for coefficient values that maximize consistency between the model and the data.

These mathematical properties make logistic regression particularly attractive for OQM, as they provide interpretable probabilistic estimates. The odds ratio "e^{β_i}" quantifies the multiplier effect on the probability ratio associated with a unit increment of the variable x_i, information that process engineers frequently use to prioritize corrective actions and to draw up operating rules.

After a careful feature engineering phase, given that in the industrial field logistic regression is often used as a baseline that can be interpreted in quality analysis pipelines, a logistic model is built to estimate the probability of defect and to define operational thresholds. Applications of LR for quality control are documented in the work of M. Sharp et al. [19] in which a good initial compromise between interpretability and performance is represented, especially when the amount of labeled data is not huge or when traceability of decisions is required.

However, the practical limitations of the technique should be emphasized; in fact, the linear relationship between log-odds and predictors can be too restrictive in the presence of complex interactions and non-linearities typical of industrial processes, and in such cases it is necessary to expand the model with polynomial terms, interactions, or to switch to nonlinear versions or to more flexible models. When defects are very rare events, so there is a strong imbalance, classical logistic estimation can suffer; practical strategies include weighting observations, SMOTE or the use of cost-sensitive loss to incentivize recall; the presence of strong multicollinearity between sensors or related features requires selection or regularization techniques to ensure stable estimates.

Random Forest

The next algorithm we want to analyze is the Random Forest, which represents one of the most widespread and reliable techniques of supervised machine learning for classification and regression in the industrial field. It was introduced by Leo Breiman in 2001 [20] and is an ensemble model,

i.e., it is based on the idea of combining a large number of independent decision trees, each of which is trained on a different sample of the original data, to obtain a more stable and accurate overall forecast than that provided by a single tree. This approach is based on two fundamental mechanisms: bagging, which consists of generating multiple trees on random subsets of the dataset, and the random selection of a subset of variables at each decision node. This double randomness reduces the correlation between trees and limits the risk of overfitting, improving the model's ability to generalize.

Each decision tree in the forest recursively divides the variable space into regions that are increasingly homogeneous with respect to the target variable, using criteria such as reducing Gini impurity or entropy in the case of classification, or minimizing variance in the case of regression. The final prediction of the Random Forest is obtained by aggregating the results of all the trees: in the case of a classification problem, the most voted class is chosen, while for the regression the average of the predicted values is calculated.

In the OQM context, Random Forest is widely used thanks to its robustness, flexibility and ease of interpretation; industrial data are often complex, noisy and characterized by many variables of a heterogeneous nature (sensors, process measurements, environmental parameters, quality tests), and the Random Forest adapts perfectly to these conditions without requiring excessive data preparation or normalization, effectively managing the presence of missing values, categorical variables and non-linear relationships between process variables and product quality.

One of the main advantages, compared to other machine learning algorithms, lies in the possibility of extracting measures of importance of variables (*feature importance*), which quantify the contribution of each parameter to the final decision. From a quality management perspective, this allows critical process variables to be identified and provides production engineers with practical guidance to optimize processing parameters, reduce scrap, and improve process stability.

Chen et al. [21] document how this algorithm is frequently adopted for quality prediction problems, as well as for the classification of compliant and non-conforming products in industries such as automotive and electronics; moreover, research shows, for example, how a Random Forest model used to classify four typical types of welding defects achieved an accuracy of 97% [22] or how a method based on Random Forest and feature selection predicted defects in hot rolling [23].

At the same time, some limitations must be considered: the model, while providing aggregate indications on the importance of the features, is less interpretable in detail than a single decision tree, and in an extremely large dataset or with real-time inference needs, the training and prediction phases can be computationally expensive. However, there are parallel and optimized versions that mitigate these aspects, keeping Random Forest among the most balanced methods in terms of accuracy, robustness and ease of use.

Extreme Gradient Boosting (XGBoost)

Another algorithm being analyzed is Extreme Gradient Boosting (XGBoost) [24], an ensemble model, which stands out as one of the most effective and widespread tools thanks to its combination of predictive power, robustness and scalability. In its essence, it implements the gradient boosting paradigm on decision trees by building an additive model of the type

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F} \tag{2.7}$$

where each "f_k" is a decision tree that is trained sequentially with the aim of correcting the residual errors of the previous model. The objective function to be minimized takes the form

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{2.8}$$

where "l" is the loss (e.g. log-loss for classification, MSE for regression) and "Ω" is a regularization term that penalizes the complexity of each tree (number of leaves, weight of nodes, depth). In this way, XGBoost not only learns a good approximation of the target function, but does so in a controlled way, reducing the risk of overfitting, i.e., that the model fits too closely to the data and performs worse.

In the OQM field, XGBoost is particularly suitable for binary classification (compliant or defective part, defect types), regression (yield prediction, estimation of critical process variables), and failure prediction (when targets are built as binary labels or as RULs); however, its effectiveness depends heavily on the quality of the feature engineering and the correctness of the validation.

What makes it a very suitable algorithm for the OQM theme are the excellent performance on tabular data typical of production logs, the robustness thanks to the built-in regularization, the management of missing values, the ability to provide scores of importance for the features during analysis and the compatibility with post-hoc interpretability techniques (e.g., SHAP) that allow the translation of predictions into insights that can be technically interpreted by process engineers [24].

Using XGBoost typically requires integration into complex pipelines that include feature engineering, data skew management (via class weighting or SMOTE), hyperparameter optimization, and robust validation. In this context, S. Lee's [25] study proposes a hybrid methodology structured in two phases. Initially, the model is used to predict the outcome of the process based on input and control variables. Subsequently, a meta-heuristic algorithm (Differential Evolution) is applied to the resulting predictive model to identify the optimal combination of process variables. The ultimate goal is not only to predict the outcome, but to optimize the entire system configuration to minimize (or maximize) a specific user-defined objective function.

Isolation Forest (iForest)

Among the unsupervised learning algorithms is the Isolation Forest (iForest) [26], an algorithm designed specifically for the detection of anomalies, which is based on an intuition very different from traditional techniques: instead of explicitly modeling "normality" and looking for deviations, iForest tries to isolate anomalous examples by exploiting the fact that outliers, being few and characterized by atypical values, are easier to separate than regular points. The construction of the model takes place by creating a forest of random binary trees, called isolation trees, in which each node is obtained by randomly choosing a feature and a split threshold; a sample is "isolated" when, going down the tree, it reaches a leaf that contains it by itself and the length of the path from the root to the leaf is therefore an indication of the ease of isolation; therefore, the anomalous points tend to have, on average, shorter paths compared to normal points. By aggregating the length of the paths on multiple trees, an anomaly score is obtained for each point, and thresholds on this score allow anomalous vs normal instances to be classified.

From a mathematical point of view, there is no single complex "objective function" to optimize as in parametric models, but the statistical effect on which the detection is based is that the average length of the path $h(x)$ for a point x in the sample is related to the probability that that point is anomalous; in practice, the average depth $E(h(x))$ is calculated on random t trees and normalized with respect to the expected length for a sample in a random subset of data of dimension n. The usual definition of the score is the form

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{2.9}$$

where $c(n)$ is a normalization term that depends on the sample size n. This term is defined as:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}$$

where $H(k) = \sum_{i=1}^{k} \frac{1}{i}$ is the k-th harmonic number. Values of $s(x,n)$ close to 1 indicate a strong anomaly, significantly lower values indicate more "normal" behavior.

Isolation Forest has quickly become a natural choice for monitoring and diagnosis because many OQM problems lend themselves to the "find the few measures that behave differently" formulation, which has fostered its adoption in pipelines where automatic and continuous anomaly detection is desired with minimal human intervention. In typical industrial applications, from the early detection of machine faults to the detection of wafers or out-of-specification batches in semiconductor processes, the unsupervised nature and scalability make iForest particularly practical; moreover, since the algorithm directly isolates outliers without requiring strong assumptions about data distribution, it is robust in the presence of nonlinearity, multimodality, and process variability [26].

The efficacy described in production environments is documented by application works and methodological extensions such as Puggini and McLoone [24], who proposed to integrate a targeted selection of variables (feature selection) with Isolation Forest for optical spectroscopy data used in plasma monitoring. By combining this dimensionality reduction with iForest, they improved the ability to find fault-related anomalies in etching processes, overcoming the limitations of traditional approaches. Similar studies apply on-line or incremental versions of iForest for data-streaming processes, as shown in work on anomaly detection in etch tools and at other stages of semiconductor manufacturing, where low latency and the ability to react to rapid changes are essential requirements [27].

However, in the OQM environment it is also necessary to be aware of the limitations of this algorithm, which can confuse regions with a low density of normal points with real anomalies if the data have scattered clusters, and it does not explain the physical causes of the anomaly; for this reason, in industrial applications, it is often integrated with explainability methods or with subsequent phases of diagnosis that associate anomalous patterns with known root causes.

Support Vector Machine (SVM)

Following an unsupervised anomaly detection method, we transition to a supervised method that learns the optimal boundary between known classes: the Support Vector Machine (SVM). It represents a family of supervised learning algorithms originally designed for binary classification, but extendable to regression and one-class applications for anomaly detection. The fundamental intuition behind SVMs is the search for a separating hyperplane in the feature space that maximizes the margin, defined as the minimum distance between the hyperplane itself and the closest points of the two classes. This margin maximization criterion is linked to the Structural Risk Minimization principle, which endows SVMs with strong generalization properties even in the presence of noisy data, as formalized in the seminal work by Cortes and Vapnik [28]. By solving a constrained quadratic optimization problem, this approach yields a classifier with excellent theoretical and practical properties.

To convey this concept mathematically, in the linearly separable formulation, an SVM aims to solve:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.10}$$

subject to the constraints

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, n \tag{2.11}$$

where $\mathbf{x}_i$ represents the training sample, y_i denotes the label, $\mathbf{w}$ is the weight vector, and b is the bias; minimizing the norm of the weight vector is equivalent to maximizing the margin.

These properties make SVMs particularly suitable for various OQM tasks. In manufacturing contexts, SVMs are commonly applied for the classification of compliant vs. defective parts, the recognition of anomalous patterns in process measurements, fault diagnosis from vibrational or spectral signals, and as a regressor (Support Vector Regression, SVR) to estimate critical process variables. Consequently, SVMs are among the most widely used methods due to their ability to handle non-linear relationships while maintaining robust generalization capabilities, even with moderately sized datasets [29]. Numerous studies on rotating machinery diagnostics, tool wear, and chemical processes report strong SVM performance—even when compared to alternative methods—provided that feature representation is carefully engineered. A specific reference work is the survey by Widodo and Yang [30], which documents the application of SVMs in condition monitoring and fault diagnosis, demonstrating concrete case studies and practical advantages.

Onel et al. [31] propose a non-linear SVM-based feature selection method for fault detection and diagnosis, demonstrating that pairing targeted feature selection with an SVM enhances the ability to detect faults in complex industrial signals, thereby reducing false alarms and improving accuracy. Similarly, for scenarios lacking fault labels, the One-Class SVM has been successfully applied to anomaly detection tasks in manufacturing, as seen in works such as Shin et al. [32].

From an operational perspective, the adoption of SVMs requires attention to several practical aspects: first, it is standard practice to perform hyperparameter optimization via cross-validation; furthermore, SVMs do not inherently provide feature importance metrics comparable to tree-based models. Therefore, practitioners tend to combine SVMs with interpretability tools or utilize them as a high-accuracy component within a pipeline that includes an interpretable model to facilitate AI communication with process engineers.

Fuzzy C-Means (FCM)

Within the analysis of the articles related to the dataset under consideration, a clustering algorithm is also present: Fuzzy C-Means (FCM). This method represents an extension of the traditional k-means clustering—which is characterized by a crisp and rigid data classification—by inheriting its core partitioning logic while introducing a fundamental element of flexibility: the possibility for each data point to belong simultaneously to multiple clusters with varying degrees of membership. This characteristic makes it uniquely suited for the analysis of real-world manufacturing processes. Indeed, it is common for certain operating conditions to share similar features but with different intensities; the algorithm allows for the representation of this gradual transition by assigning a membership vector to each point, indicating the degree or probability of its belonging to each cluster. This is useful for defining soft labels that describe transitional states or intermediate process conditions, thereby providing richer information compared to a simple hard assignment.

The FCM algorithm was formalized by Bezdek [33], who defined its objective function to be minimized—which combines the distance between data points and cluster centroids with the membership degree of each point—along with the iterative update rules. In general terms, the function is:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m \|x_i - c_j\|^2 \tag{2.12}$$

where u_{ij} represents the degree of membership of point x_i in cluster c_j, c_j is the cluster center, and m is the fuzziness parameter, which controls how "soft" or blurred the assignment of points is. Values close to 1 make the clustering crisper, resembling k-means, whereas higher values allow for greater overlap between clusters. During training, FCM iteratively updates the centroids and membership degrees until the solution converges, thereby balancing the average representation of the data and its distribution across clusters.

In the field of Operations Quality Management, one of the most common areas of application is fault diagnosis in industrial machinery: Liu et al. [34] utilized FCM for fault diagnosis in mechanical systems, combining it with fuzzy measures and dimensionality reduction techniques, demonstrating that it can effectively discriminate among different operating conditions, even in the presence of noise and variability typical of industrial environments. In a more recent study, Li et al. [35] demonstrated the effectiveness of various fuzzy clustering variants for bearing fault diagnosis, achieving promising results in early anomaly detection.

Naturally, like any method, Fuzzy C-Means has certain limitations, as it is sensitive to outliers, which can significantly shift the centroids. Furthermore, it requires an a priori selection of the number of clusters and the fuzziness parameter, both of which heavily influence the results. To mitigate these issues, several robust or optimized variants have been proposed, and for large datasets, scalable and incrementally updatable versions exist, enabling application in real-time manufacturing environments.

Feed Forward Neural Network (FFNN) and Multilayer Perceptron (MLP)

We now address the more complex models encountered in the literature: neural networks. In this context, it is appropriate to analyze the family of Feed Forward Neural Networks (FFNNs), which represents the most basic and conceptually transparent form of artificial neural networks. It is important to clarify that, although these terms are often used interchangeably in the literature, the Multilayer Perceptron (MLP) constitutes the most widespread and practical specific architecture within the broader FFNN family.

In the former, information flows in a strictly unidirectional manner, from the input nodes toward the output nodes, traversing one or more layers of artificial neurons without feedback loops. This simple yet powerful structure forms the foundation for numerous, more complex architectures in modern machine learning. Theoretically, FFNNs are inspired by the functioning of biological neurons, where each node linearly combines the received input signals and subsequently applies a non-linear activation function. Mathematically, the output of a neuron can be expressed as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.13}$$

where x_i represents the input, w_i denotes the synaptic weight that determines the importance of each input, b is the bias, and $f(\cdot)$ is an activation function. The weights are updated during the training phase with the objective of minimizing the error between the network's output and the target values, according to a cost function. The learning process is carried out via gradient descent or its derived algorithms; in the case of multilayer networks (MLPs), it is executed through the well-known backpropagation technique, formalized in the foundational texts by Rumelhart et al. [36]

and Haykin [37].

The addition of one or more hidden layers enables the network to learn complex non-linear relationships, making it an extremely powerful tool for identifying product quality indicators. In this regard, the universal approximation theorem [38] explains why even an MLP with a single hidden layer and a sufficient number of neurons can approximate arbitrarily complex continuous functions, justifying its adoption in cases where the relationships between process inputs and final quality cannot be easily modeled using linear statistical methods.

In Operations Quality Management, these networks are successfully deployed across several concrete areas: Widodo and Yang [30] demonstrated the effectiveness of FFNNs in rotating machinery fault diagnosis, achieving high accuracy even in the presence of noise; similarly, Andersen et al. [39] utilized them in electronics manufacturing (SMT) by integrating sensor data, while Zhou et al. [40] applied feed-forward models to predict quality deviations in steel production. In domains such as injection molding, the MLP allows for the prediction of geometric defects starting from pressure and temperature sensors [41]; conversely, G. A. Susto et al. [42] employed it in virtual metrology for semiconductors, estimating variables that would otherwise require expensive and time-consuming physical measurements. Due to their ability to recognize the normal "signature" of a system, these networks are also integrated into real-time monitoring pipelines to instantaneously flag anomalous deviations.

From a practical standpoint, the adoption of MLP architectures requires a careful preliminary pre-processing phase, including normalization, dimensionality reduction, and feature identification, as accuracy heavily depends on data quality and proper hyperparameter tuning. Compared to models such as Random Forest or SVM, feed-forward networks are less interpretable due to their black-box nature, but they offer greater flexibility in the presence of highly interdependent variables. To mitigate this critical issue, explainability techniques such as SHAP or LIME are adopted, along with regularization and robust validation schemes. Finally, the MLP is often incorporated into hybrid architectures: for instance, clusters derived from fuzzy algorithms (FCM) can be used as conditional inputs for local MLP models, thereby enhancing operational robustness [43]. As highlighted by recent reviews [44], these networks remain a premier choice for predictive quality projects, provided they are embedded within operational pipelines that include continuous performance monitoring and controlled retraining mechanisms as process conditions vary.

Table 2.1: Comparison of Machine Learning Algorithms for OQM (Operations Quality Management)

Method	**Applications**	**Pros (Advantages)**	**Cons (Disadvantages/Limits)**	**Performance**
KNN	Product defect prediction and fault detection in semiconductor manufacturing processes.	Very immediate and intuitive. Does not build a complex mathematical model. Adaptable to unsupervised mode.	Performance influenced by choice of 'k' (sensitive to noise for small 'k', less precise on boundaries for large 'k'). Requires feature normalization.	The study [11] achieved an accuracy of 86.41% (with Euclidean distance). More effective than PCA in nonlinear environments.
Naive Bayes	Rapid classification of process outcomes. Defect prediction from sensors, tests, and indicators.	Simple but surprisingly effective. Extremely computationally lightweight. Probabilistic outputs useful as decision scores.	The feature independence assumption is often violated in process data, which can degrade estimates.	In study [16], an identification accuracy of 97.92% is reached.
Logistic Regression	Prediction of binary outcomes. Estimation of defect probability in quality control.	Often used as an interpretable baseline. Provides probabilistic estimates. The Odds Ratio quantifies the impact of each variable.	The linear relationship can be too restrictive for complex nonlinearities. Can suffer with rare events (imbalanced classes).	
Random Forest	Classification and regression. Quality prediction (compliant/non-compliant) in automotive/electronics. Welding defects.	Widespread and reliable. More stable than a single tree. Limits overfitting. Handles heterogeneous, missing, and non-linear data.	Less interpretable than a single tree. Can be computationally expensive on extremely large datasets or in real-time.	In study [22], a model for welding defects achieved an accuracy of 97%.

Continued on next page...

Table 2.1 – continued from previous page

Method	Applications	Pros (Advantages)	Cons (Disadvantages/Limits)	Performance
XGBoost	Binary classification, regression (yield prediction), failure prediction. Process optimization.	Combination of predictive power, robustness, and scalability. Built-in regularization. Handles missing values.	Effectiveness depends heavily on feature engineering and validation.	Study [24] demonstrates that the model performs calculations over 10 times faster than existing solutions.
Isolation Forest	Anomaly detection (unsupervised). Early detection of machine failures and out-of-spec batches.	Intuitive (isolates outliers). Scalable and unsupervised. Robust to nonlinearity and multimodality.	Can confuse low-density normal regions with anomalies. Does not explain physical causes (requires extra diagnosis).	In study [27], the model obtained a sensitivity of 100% and false positives at 2.8%.
SVM	Binary classification, regression (SVR), and anomaly detection (One-Class) for fault diagnosis.	Strong generalization properties. Handles nonlinear relationships (kernels). Good with small/medium datasets.	Requires hyperparameter optimization. Does not provide direct feature importance measures.	In study [31], the Fault Detection Rate was improved, increasing the detection rate to over 94%.
Fuzzy C-Means	Clustering and fault diagnosis (e.g., bearings). Definition of "soft labels" for transition states.	Flexible extension of k-means. Allows partial membership (degrees of belonging).	Sensitive to outliers. Requires prior choice of cluster number and fuzziness parameter.	
FFNN	Quality prediction, anomaly detection (e.g., steel, SMT electronics).	Autonomously learns nonlinear correlations. Flexible in the presence of strong nonlinearity.	Requires accurate pre-processing. Accuracy depends on data quality and hyperparameters. "Black box" nature.	

Continued on next page...

Table 2.1 – continued from previous page

Method	Applications	Pros (Advantages)	Cons (Disadvantages/Limits)	Performance
MLP	Modeling nonlinear relationships, Virtual Metrology, quality prediction (e.g., injection molding).	Universal approximator. Often the initial choice for predictive quality projects.	Less interpretable than linear/tree models. Quality depends on label quantity. Critical hyperparameter optimization.	In article [41], the model achieved a mean relative error lower than 0.2% and a correlation index (R^2) close to 0.98.

2.2 Literature regarding the target dataset

A significant portion of recent literature dedicated to *Operations & Quality Management* systems, supported by machine learning techniques, has utilized the *Predicting Manufacturing Defects* dataset [9] published on Kaggle. This dataset has been employed across heterogeneous methodological approaches, ranging from simple comparative evaluations of algorithms to the development of advanced hybrid models and interpretability frameworks for analyzing factors determining production process quality. The analysis of these works allows for the delineation of a performance and methodological benchmark for this thesis.

One of the highest-performing approaches was proposed by Mehregan et al. (2025) [45], who developed a hybrid model aimed at maximizing defect prediction accuracy, overcoming the limitations of individual traditional algorithms. The authors' approach is based on the *Stacking Ensemble* technique, a methodology combining predictions from various "base" models to train a final "meta-model". The architecture consists of four *base learners* (XGBoost, LightGBM, CatBoost, and an Artificial Neural Network), constituting "level 0" to capture diverse data facets, and a meta-model, "level 1", which uses the previous level's predictions as input for a Random Forest acting as the final binary classification decision-maker.

A crucial aspect of this study is rigorous data management. To address class imbalance, given the dataset's skew between defective and non-defective products, the authors applied the SMOTE technique to generate synthetic minority class samples, balancing the training set. *Feature selection* based on Random Forest feature importance was applied to reduce dimensionality, selecting only the top 7 most influential variables (*Maintenance Hours*, *Defect Rate*, *Quality Score*, *Production Volume*, *Additive Material Cost*, *Stockout Rate*, and *Energy Efficiency*), which alone explain approximately 80% of the information. Finally, to maximize performance, hyperparameters for all models were optimized using the Optuna framework. This approach yielded the highest results among analyzed studies, achieving an accuracy of 96.06% and an F1-Score of 96.20%, demonstrating the superiority of complex ensemble techniques on this specific dataset. This highlights how integrating heterogeneous, appropriately calibrated models can significantly improve the identification of defective products within complex, highly non-linear production contexts.

In parallel, Marín Díaz (2025) [46] proposed a dual framework integrating supervised learning with unsupervised clustering techniques and structural exploration, placing strong emphasis on decision-making transparency via *Explainable AI* (XAI) techniques. A methodology articulated on

two parallel tracks was developed to ensure not only accuracy but also operational interpretability. The first track, dedicated to supervised prediction, compared various algorithms, including SVM, KNN, and Random Forest, identifying XGBoost as the best-performing model for binary defect classification, with an accuracy of 95.37% and an F1-Score of 97.30%. The second track, unsupervised in nature, aimed to segment latent production profiles using clustering algorithms. The author tested both *Fuzzy C-Means* (FCM) and *K-Means*, finding that the latter, configured with $k = 3$, offered sharper operational segmentation than the fuzzy approach, which failed to adequately separate structural data profiles.

Regarding dataset preparation, numerical variables were normalized to the $[0, 1]$ range via Min-Max scaling to ensure compatibility with clustering algorithms. To manage class imbalance, a stratified dataset split (80/20) and evaluation metrics sensitive to unbalanced classes, such as precision and recall, were chosen. The most significant contribution of this work lies in the extensive application of post-hoc interpretability techniques: the author employed five distinct methods (SHAP, LIME, ELI5, PDP, and ICE) to audit the model. Cross-analysis of these methods confirmed that *Maintenance Hours*, *Defect Rate*, and *Quality Score* are the dominant predictors. Specifically, PDP charts highlighted a critical non-linear relationship: defect probability increases dramatically when normalized maintenance hours exceed the 0.4 threshold. This article confirms XGBoost as an extremely robust algorithm for this data, serving as an excellent baseline if complex ensembles are not used; it further suggests that analysis should extend beyond prediction to production process segmentation via clustering.

Goetzinger (2025) [47] aimed to investigate the *root causes* of manufacturing defects using a *data-driven* approach, shifting focus from simple defect detection to understanding the generating production factors. The study confirmed the effectiveness of gradient boosting-based algorithms by proposing a hybrid model: XGBoost-FNN. The former is used as a base model for its ability to handle tabular data and provide clear, interpretable feature importance scores, while the latter captures complex sequential and non-linear relationships between input parameters and defect states. In addition to these main models, the study explored Isolation Forest for anomaly detection, though with limited results due to the dataset's specific nature, containing a high frequency of simulated defects, rendering them less "anomalous" statistically. The analysis strongly focused on causal factors, highlighting through SHAP (*SHapley Additive exPlanations*) values how increased maintenance hours and production volumes positively correlate with increased defects, while high *Quality Score* and *Stockout Rate* act as risk reduction factors. Despite the interpretability focus, the hybrid XGBoost-FNN model demonstrated top-tier performance, with an accuracy of 95.37% (identical to Marín Díaz), an ROC-AUC of 87.38% (confirming excellent class discrimination), and a Recall of 98.72%, a critical value for quality control, indicating very few missed real defects. The study underscores the importance of analyzing impact directionality, not just generic variable importance.

Finally, David and Firdaus (2024) [11] explored a more traditional approach, evaluating the effectiveness of a geometric proximity-based method for defect prediction by testing different distance metrics and *K-Nearest Neighbor* (KNN) algorithm configurations to identify optimal operating conditions. The investigation focused on the impact of distance metric choice on predictive capacity. Specifically, three metrics were evaluated with k values between 2 and 14: Euclidean distance (classical linear distance in vector space), Manhattan distance (sum of absolute coordinate differences, often more robust in high dimensions), and Minkowski distance (a generalization of the former, including an adjustable parameter for data space curvature). A critical point radically distinguishing this work is the pre-processing strategy for class imbalance, using *Random Under-Sampling*. This technique balances the dataset by randomly eliminating majority class samples (non-defective

products) until parity with the minority class (defective ones) is reached. This choice involves a massive loss of potentially useful information, significantly reducing the algorithm's learning base and resulting in lower performance. The results reflect the limitations of the geometric approach on this specific complex dataset; the best accuracy of 86.41% was obtained using Minkowski distance with $k = 10$. This suggests that for this dataset, more complex algorithms capable of capturing non-linear relationships, such as XGBoost, are preferable, and that *Under-sampling* is a suboptimal strategy compared to *Over-sampling*, favoring techniques that preserve or augment data rather than reducing it.

Chapter 3

Materials and Methods

3.1 Dataset exploration and preparation

This chapter describes the methodological approach adopted for the exploratory analysis of the "manufacturing_defect_dataset.csv" [9] dataset, which contains information related to several factors influencing the defect rate in a simulated production context. This has made it possible to build a clean, consistent and representative database useful for extrapolating strategic information and actionable insights, analyzing the relationships between operational variables and the quality of the final product and supporting the subsequent phases of modeling and validation of predictive models.

The methodological approach was divided into three progressive and logically linked phases, described below.

3.1.1 Numerical and Graphical EDA

The first phase of the `I_Feature_analysis.py` analysis was dedicated to the rigorous validation of the dataset, preparatory to any subsequent investigation, as it ensures that the conclusions are based on intact, clean and correctly interpreted data. The process began with the acquisition of the dataset and a preliminary inspection of its structure and dimensionality and concluded that it is composed of 3,240 observations and 17 variables, of which 16 represent the independent features and one the target variable (DefectStatus), which indicates whether a product is compliant (0) or defective (1).

In detail, the 17 features analyzed are:

- **ProductionVolume:** number of units produced in a given period;
- **ProductionCost:** total production cost per batch;
- **SupplierQuality:** quality index of materials supplied by suppliers;
- **DeliveryDelay:** days of delay in deliveries;
- **DefectRate:** defect rate detected in the production process;
- **QualityScore:** overall product or process quality score;
- **MaintenanceHours:** total hours of maintenance on equipment;
- **DowntimePercentage:** percentage of machine downtime;
- **InventoryTurnover:** number of inventory turnovers in a period;
- **StockoutRate:** percentage of out-of-stock events;
- **WorkerProductivity:** average index of worker productivity;

- **SafetyIncidents:** number of accidents or safety events recorded;
- **EnergyConsumption:** total energy consumption (kWh);
- **EnergyEfficiency:** energy efficiency of the process;
- **AdditiveProcessTime:** time spent in additive processes;
- **AdditiveMaterialCost:** cost of materials used in additive processes;
- **DefectStatus:** Binary target variable (0 = compliant, 1 = defective).

A completeness analysis was then conducted with which the absence of missing values was certified, which would have required potentially distorting imputation techniques, thus ensuring the quality of the information base. At the same time, the target variable DefectStatus was examined through a grouping, in order to understand its distribution and identify a possible class imbalance; This phenomenon is common in defect datasets and its quantification is crucial for the subsequent predictive modeling phase. In fact, the number of compliant products was significantly lower than that of defects (usually the opposite situation occurs), so it will be decided to apply, in the subsequent modeling phases, the *SMOTE (Synthetic Minority Over-sampling Technique)* oversampling technique to balance the classes and improve the predictive capacity of the algorithms.

In order to understand the central tendency (mean, median), dispersion (standard deviation) and distributional form of each individual process parameter, we continued with the calculation of the fundamental descriptive statistics for each numerical variable of the dataset, allowing to obtain an initial overview of the distribution of the data and identifying any anomalies and ranges of variability that are not consistent with the simulated production context. Later, a systematic outlier analysis based on the Interquartile Range (IQR) method was implemented and for each variable the following were calculated:

- the first and third quartiles (Q1, Q3);
- the interquartile range (IQR = Q3 – Q1);
- the lower and upper limits for the definition of outliers (Q1 – 1.5·IQR, Q3 + 1.5·IQR), these values are set to achieve a compromise between sensitivity and robustness.

	Total_outlier	Percentage	Lower_limit	Upper_limit
DefectStatus	517.0	15.96	1.00	1.00
StockoutRate	0.0	0.00	-0.05	0.15
AdditiveMaterialCost	0.0	0.00	-117.46	715.56
AdditiveProcessTime	0.0	0.00	-3.54	14.51
EnergyEfficiency	0.0	0.00	-0.10	0.71
EnergyConsumption	0.0	0.00	-1006.83	6979.76
SafetyIncidents	0.0	0.00	-5.50	14.50
WorkerProductivity	0.0	0.00	70.37	109.86
ProductionVolume	0.0	0.00	-357.88	1455.12
ProductionCost	0.0	0.00	-2364.62	27217.91
DowntimePercentage	0.0	0.00	-2.50	7.54
MaintenanceHours	0.0	0.00	-11.12	33.88
QualityScore	0.0	0.00	39.73	120.73
DefectRate	0.0	0.00	-1.86	7.36
DeliveryDelay	0.0	0.00	-3.50	8.50
SupplierQuality	0.0	0.00	69.99	109.67
InventoryTurnover	0.0	0.00	-2.12	14.15

Figure 3.1: Outlier analysis results.

The results were summarized in a dataframe 3.1 in which it is noted that none of the numerical predictive variables have anomalous values, which from a management perspective can represent both measurement errors to be filtered out, and rare but significant process events that deserve a dedicated investigation; for all these features, the percentage of outliers was equal to 0.00%. The only variable that reports a value of 15.96% of "outlier" is DefectStatus, but this should not be interpreted as a presence of anomalies, but as a methodological artifact, i.e., since DefectStatus is the target variable of binary nature, the application of a statistical technique designed for continuous variables (such as IQR) erroneously identifies the minority class as statistical "outliers".

The second `II_EDA_grafica.py` phase has translated the analysis from the purely numerical to the visual level, with the aim of identifying patterns, latent correlations and the main drivers of defectiveness. The investigation started from the understanding of the distributive shape of the individual variables through the generation of histograms, one for each variable.

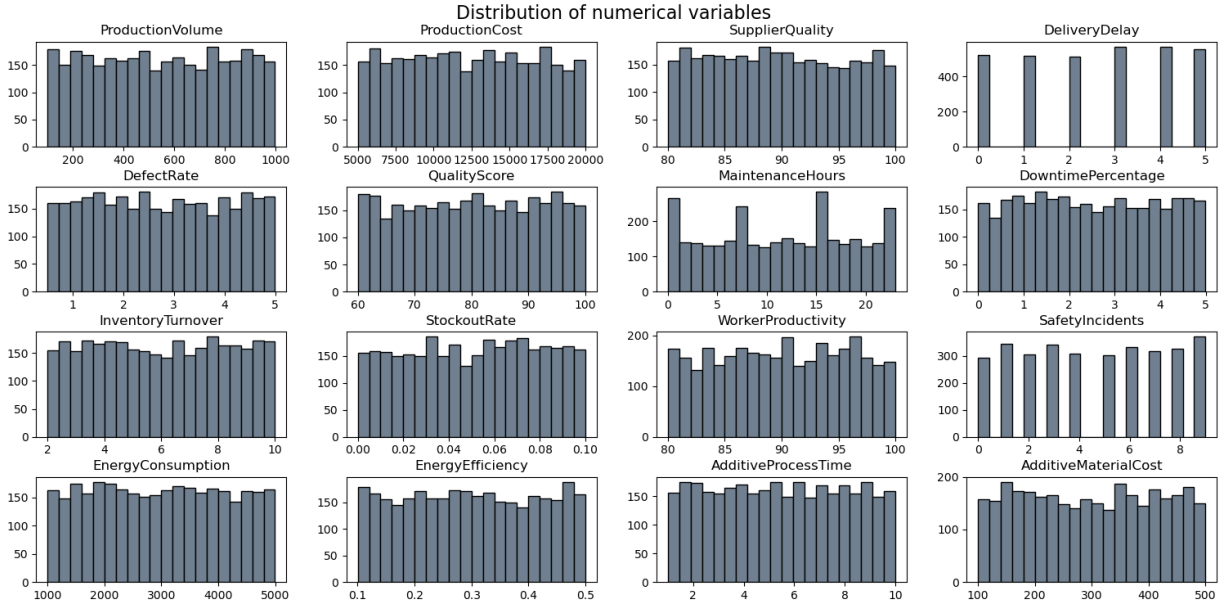

Figure 3.2: Distribution of numerical variables.

A predominantly uniform distribution is noted 3.2, indicating that the simulated data cover the entire range of each feature with an almost constant frequency, without showing a central tendency or significant asymmetries; there are discrete variables, "DeliveryDelay" and "SafetyIncidents", whose histograms do not have a continuum, but distinct and separate bars, centered on integer values; the irregular distribution of the variable "MaintenanceHours", neither uniform nor normal, but irregular and potentially multi-modal, with different peaks and valleys, indicates a non-constant frequency of maintenance interventions at different hourly intervals.

Once the univariate analysis was completed, the exploratory investigation continued with the study of multivariate relationships, using the matrix of Pearson correlation coefficients 3.3 (graphically displayed through a *heatmap*), crucial to understand the interdependencies between the process variables and the target variable and identify the main predictors of defectivity.

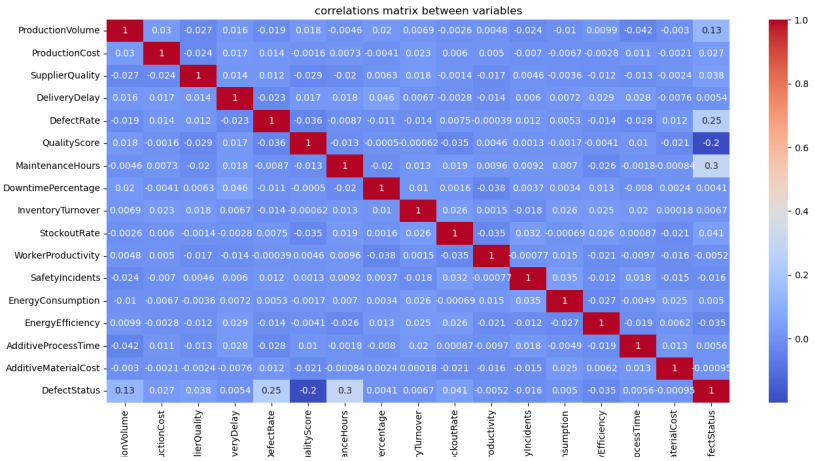

Figure 3.3: Correlations matrix between variables.

The analysis is aimed at diagnosing the possible presence of multicollinearity, i.e. strong correlations between the features themselves, a phenomenon that can affect the stability and interpretability of some predictive models, in particular those of a linear nature. The heatmap represents the correlation

coefficients "ρ" (rho), which vary in a range $[-1, +1]$. Values close to +1 (in red) indicate a strong positive correlation, values close to -1 (in blue) a strong negative correlation, while values close to zero denote an absence of linear dependence. From the last line relating to DefectStatus, a picture of generally weak linear correlations with the target variable emerges, in fact the most significant relationships, although of moderate magnitude, are: QualityScore ($\rho = -0.3$); DefectRate ($\rho = 0.25$); MaintenanceHours ($\rho = -0.2$). These correlations will be analyzed in detail later. Examining the internal correlations between the 16 predictive variables, it is evident, from the coefficients almost universally close to zero, a substantial absence of multicollinearity. This implies that all features carry non-redundant information and can be included in subsequent training phases without risking numerical instability in the model coefficients.

We then moved on to analyze in detail the distribution of "DefectStatus", the target variable, quantifying the number of observations for each of the two classes.

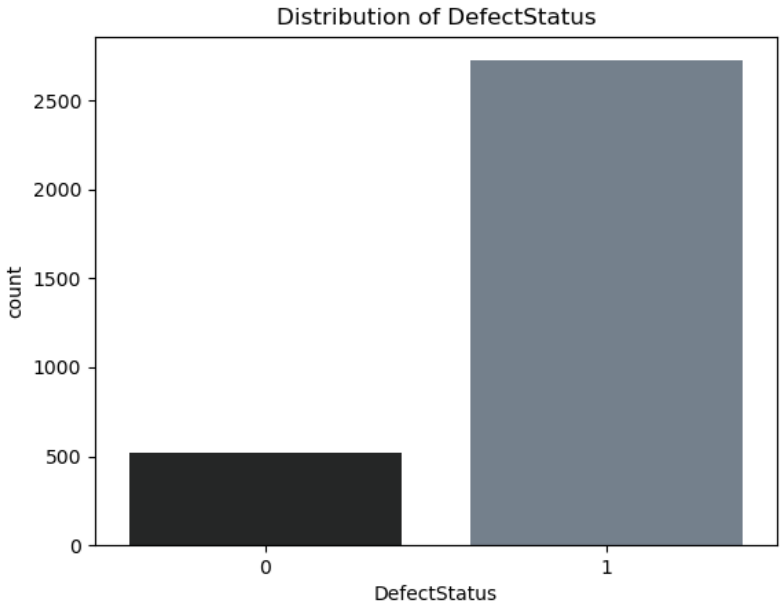

Figure 3.4: Distribution of DefectStatus.

As can be seen 3.4, there is a strong imbalance between the classes, as the majority class, corresponding to defective products (1), dominates the dataset with a count of 2723 observations and the minority class, corresponding to compliant products (0), is significantly underrepresented, counting only 517 samples. A model trained on this data would tend to develop a strong *bias* towards the majority class, maximizing accuracy simply by always predicting "defective", but failing to identify compliant cases. This analysis, therefore, unequivocally confirms the need to apply resampling techniques in the subsequent pre-processing phases.

Then we move on to the hierarchical identification of the factors of greatest impact, useful for subsequent graphical analysis, we then want to isolate and quantify the linear relationships between the predictive variables and the target variable.

```
DefectStatus            1.000000
MaintenanceHours        0.297107
DefectRate              0.245746
ProductionVolume        0.128973
StockoutRate            0.040574
SupplierQuality         0.038184
ProductionCost          0.026720
InventoryTurnover       0.006733
AdditiveProcessTime     0.005619
DeliveryDelay           0.005425
EnergyConsumption       0.005039
DowntimePercentage      0.004128
AdditiveMaterialCost   -0.000953
WorkerProductivity     -0.005224
SafetyIncidents        -0.016039
EnergyEfficiency       -0.035031
QualityScore           -0.199219
Name: DefectStatus, dtype: float64
```

Figure 3.5: Correlation with DefectStatus.

The 4 features with the highest absolute value were chosen 3.5; in fact, a positive correlation means that the two variables move in the same direction, therefore a higher probability of defect, while a negative one means that the two variables move in opposite directions, therefore less probability:

- **MaintenanceHours ($\rho = 0.297$):** it is the factor with the strongest positive correlation, counterintuitively since more maintenance should reduce defects;
- **DefectRate ($\rho = 0.246$):** it is the second factor with a positive impact, in fact as the general defect rate of the process increases, the probability that a single sample is defective increases
- **QualityScore ($\rho = -0.199$):** represents the most significant negative correlation, the higher the quality score, the less likely it is that the product is defective;
- **ProductionVolume ($\rho = 0.129$):** The third most relevant positive correlation, as production volume increases, the probability of defective parts increases.

All other variables show a much weaker linear association, with coefficients rapidly approaching zero; On the basis of these results, it was decided to focus the bivariate graphical analysis with the four variables indicated above and the target variable, to investigate, through the use of boxplots, whether the distribution of these specific predictors differs in a statistically significant way between compliant products (0) and defective ones (1).

Figure 3.6: MaintenanceHours based on defectStatus.

Figure 3.6 shows the analysis of the variable *MaintenanceHours*, through a boxplot; the analysis shows that there is a substantial difference between the two groups, visually confirming the positive correlation previously calculated. Defective products, which make up the majority of the dataset, show a median maintenance hours (approximately 13 hours) that is more than double that of compliant products (approximately 6 hours). The entire distribution of data for the defective class is visibly shifted to higher values, in fact the interquartile interval, i.e. the "box", extends approximately from 6 to 18 hours, while for the compliant it is in a lower range, between 3 and 9 hours. This result is of particular interest, suggesting that a high number of maintenance hours is strongly associated with the detection of defects; it is plausible that this data reflects reactive maintenance, i.e. the process generates a defect which in turn triggers a maintenance intervention, which is then recorded and associated with that production batch. It is noted that there are several outliers in the group of compliant products alone, this indicates that, although rare, there are cases in which a very high number of maintenance hours is associated with a final result that is still compliant, going against the general trend.

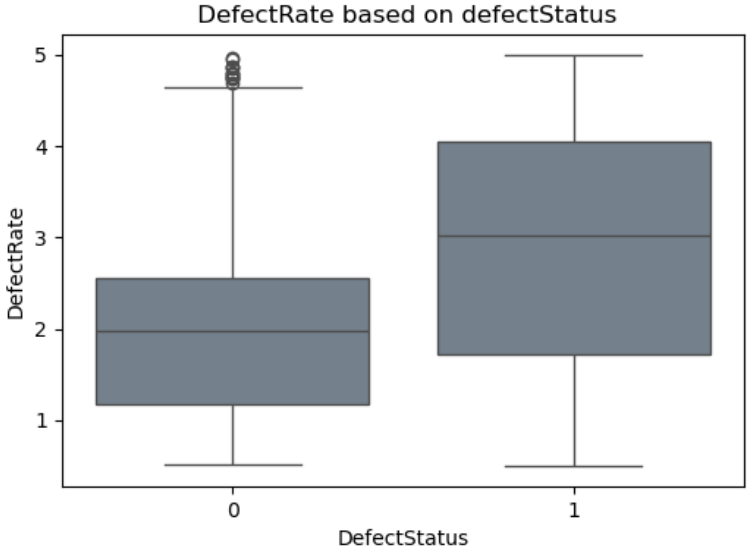

Figure 3.7: DefectRate based on defectStatus.

We continue with the graph illustrating the distribution of the DefectRate variable 3.7, the defect rate, which is the second factor for correlation impact. Also in this case, the comparison between the two classes by boxplot is clear and confirms the positive correlation, so that class 1 has a median defect rate, about 3.0, significantly higher than that of class 0, about 2.0; similarly to what has been seen for maintenance, the entire IQR of the "defective" Class is shifted to higher values (about 1.8 - 4.1) than the "compliant" Class (about 1.2 - 2.6), this indicates that the former are consistently associated with a higher rate of process defects. The result is consistent and serves as a validation for the dataset; in fact, it is entirely expected that an increase in the general defectiveness rate of the process is directly associated with a greater probability that the individual product analyzed is itself defective. In this case, some outliers for Class 0 can be seen, indicating that, although on rare occasions, it was possible to obtain a compliant product even in production batches that were recording a very high defect rate.

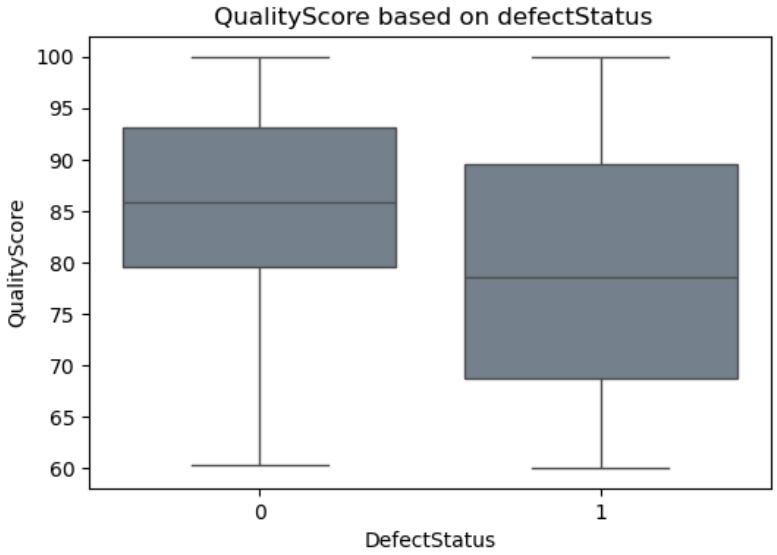

Figure 3.8: QualityScore based on defectStatus.

The third graph created concerns the QualityScore variable 3.8, which has the strongest negative correlation with the target and the boxplot visually confirms this inverse relationship. Analysis of the graph shows a behavior opposite to that of the previous two predictors, as the minority class has a significantly higher median of the quality score, about 86, compared to the median of the majority class, about 78-79. The range of Class 0 is visibly shifted to higher values than that of Class 1, logically *compliant products* are associated with higher quality scores, while defective products are associated with *lower scores*. Unlike the other graphs, in this case there are no outliers in either class, indicating that the quality scores are all within the statistical limits calculated for both groups.

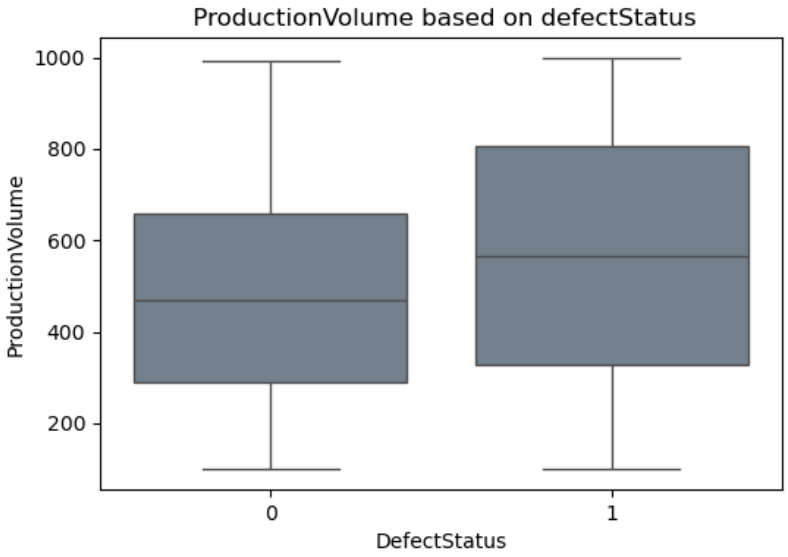

Figure 3.9: ProductionVolume based on defectStatus.

Finally, the ProductionVolume variable was examined, which ranked fourth in correlation impact 3.9. Although less markedly than the other variables, the weak positive trend detected and a difference in the medians are confirmed, in fact the defective class has a median production volume higher than that of the compliant class; the interquartile range is also shifted to higher values in the former than in the latter and appears slightly wider, indicating greater variability. It is hypothesized that higher production volumes are weakly associated with a higher incidence of defects, and that pushing the process to higher volumes may induce mild instability or stress on machinery, marginally increasing the likelihood of a non-conforming outcome. Similar to what we saw for QualityScore, there are no outliers in either distribution.

To overcome the limitations of linear correlation analysis and capture more complex interactions, an ensemble model was trained: the Random Forest Classifier. Although not used for predictive purposes at this stage, it has been used as an inferential tool to estimate Feature Importance; by analyzing the average reduction of the impurity attributable to each predictor, with the exploratory purpose of calculating the contribution of each feature, through the Gini index, it was possible to draw up an objective ranking of the most influential variables of the entire dataset.

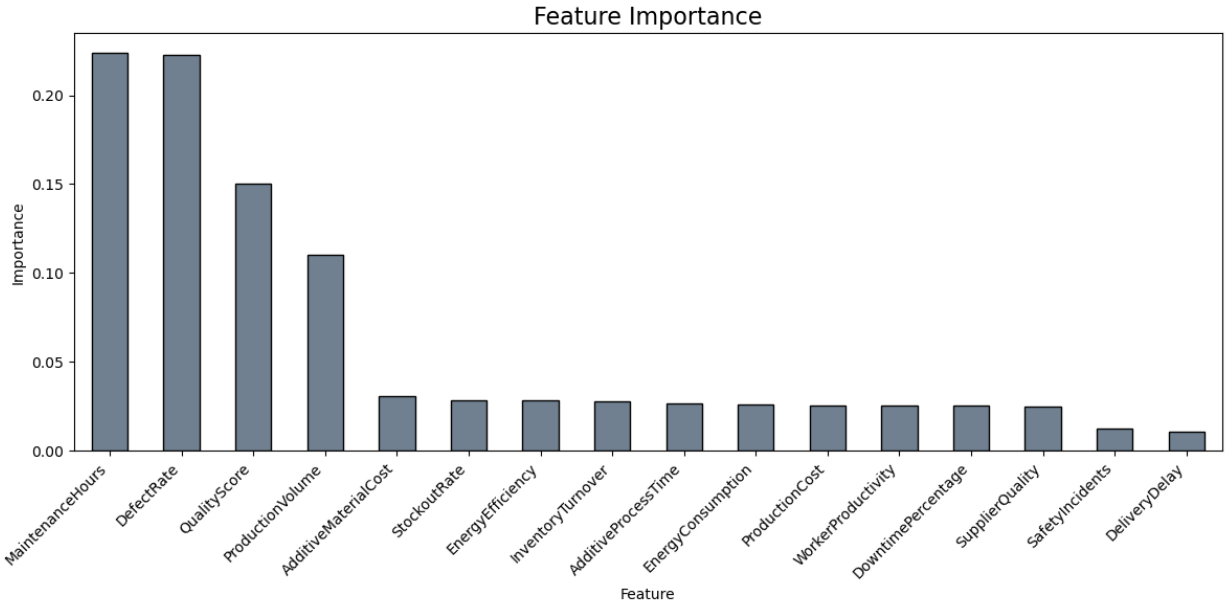

Figure 3.10: Feature Importance.

Figure 3.10 is the result and shows the hierarchy of importance of the 16 features; From the analysis we obtain strategic information of particular importance, i.e. a strong convergence with what emerged from the correlation analysis, confirming that the first variables in importance are the same as those identified previously. In particular, the dominance of the two main predictors, MaintenanceHours and DefectRate, which alone account for about 44% of the total predictive importance of the model, is noteworthy; after these first four features, a clear "elbow" is observed in the graph, so the importance of all the other twelve variables, from AdditiveMaterialCost down, collapse to very low and marginal values (all below 0.03). This double confirmation, from linear and non-linear analysis, provides an extremely solid basis for the subsequent phases, clearly indicating that any effective predictive model will have to rely primarily on the four key factors, identified as the predictors that show the clearest separation between the two classes.

The importance analysis of the predictors was then completed with a cumulative importance graph that allowed to apply a "Pareto Principle" to the process drivers, identifying the minimum subset of features that, if monitored and controlled, allows one to explain almost all the variability that leads to defects. While the graph above identified which variables were individually most important, this graph therefore answers a different and strategic question: "How many variables are needed to explain most of the model's predictive capacity?" This approach is critical for feature selection, reducing dimensionality, with the goal of finding an optimal subset of features that balances the complexity of the model with its predictive performance, reducing the risk of overfitting. The graph was obtained starting from the same "feature_importances" calculated by the RandomForestClassifier model and, after being sorted in descending order, they were aggregated through a cumulative sum; The result is a graph that shows how the percentage of total importance increases as you add variables, from the most important to the least important.

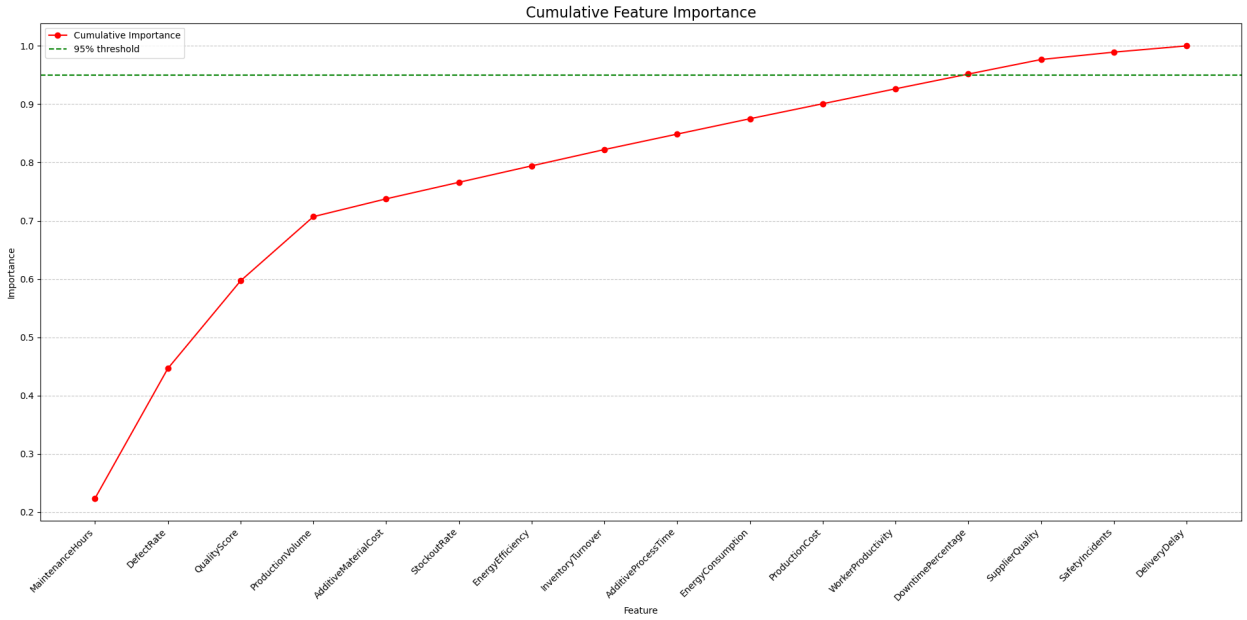

Figure 3.11: Cumulative Feature Importance.

The trend of the cumulative importance curve (red line) in Figure 3.11 in relation to the 16 features confirms, as already seen, the disproportionate contribution of the first four variables that collectively explain more than 70% of the total importance of the model; following the curve, it can be seen that it intersects a reference threshold (dotted green line) at the 13th variable, DowntimePercentage, at 95%, a value commonly used to identify the set of variables that explains almost all of the predictive variance of the model. This result is of extreme importance for the modeling phase by suggesting that the last three variables (SupplierQuality, SafetyIncidents and DeliveryDelay), which cumulatively contribute less than 5% to the total importance, could be excluded.

3.1.2 Feature Engineering

The third and final `III_explorations.py` phase represented the transition from exploratory synthesis to the generation of management insights, focusing on the creation of new metrics and new datasets specialized in the economic quantification of the observed phenomena. First, by applying the principles that emerged from the feature importance analysis, a reduced dataset called "df_subset" was generated; this subset includes only the variable "DefectStatus" and the four features with the highest predictive impact with the aim of preparing a clean and highly relevant database for model training and reducing dimensionality.

A central element of this phase was Feature Engineering, i.e. the engineering of new variables (KPIs) not present in the original dataset, but with a higher management information value. Indicators such as:

- **CostPerUnit:** calculated as "ProductionCost / ProductionVolume", it is a fundamental KPI to normalize the cost of production with respect to volumes;
- **ProductivityEfficiency:** calculated as "WorkerProductivity * EnergyEfficiency", it is a composite indicator to measure global operational efficiency;
- **MaintenanceRatio:** calculated as "MaintenanceHours / DowntimePercentage", it is an advanced metric to evaluate the effectiveness of maintenance by relating the hours of intervention to machine downtime.

The survey then focused on the strategic segmentation of the dataset to analyze cohorts of specific management interest; by applying logical filters, two distinct subdatasets were created: the first, `high_quality`, to isolate the "golden standard" products, with QualityScore > 75 and DefectRate $<$ 1; the second, `low_efficiency`, to group the processes with the most critical energy performance with EnergyConsumption > 5000 and EnergyEfficiency < 0.6.

Finally, to statistically quantify the impact of defectiveness, comparative analyses were conducted using the `groupby('DefectStatus')` function. The following values were calculated and compared 3.12: Average Cost of Non-Quality, comparing the average ProductionCost of defective products compared to compliant ones, and "Quality Gap", comparing the average values of SupplierQuality and QualityScore between the two groups, thus validating the link between quality metrics and the final state of the product.

```
DefectStatus
0    12158.877326
1    12473.169403
Name: ProductionCost, dtype: float64
              SupplierQuality  QualityScore
DefectStatus
0                   89.328686     85.442375
1                   89.929096     79.126454
```

Figure 3.12: Comparative analysis for DefectStatus.

This three-step methodological approach made it possible to move from a raw dataset to a structured understanding of production phenomena, quantifying the economic impact of defects and identifying the drivers on which to act to implement continuous improvement strategies.

3.2 Methodology adopted

Figure 3.13: PyCharm Logo

For the development of the project, **PyCharm**, an integrated development environment (IDE) created by JetBrains and optimized specifically for the Python language, was chosen. This tool provides a comprehensive set of features that support the entire software lifecycle, including writing, testing, and maintaining code.

A strength of PyCharm is its ability to manage development environments. It greatly simplifies the creation and configuration of isolated environments through tools such as `virtualenv`, `conda`, and package management with `pip`, while also allowing the installation of the necessary libraries directly from its graphical interface.

The adoption of a virtual environment for the project was a strategic choice. This practice guarantees complete isolation of dependencies: each project on the machine can thus have its own set

of libraries and specific versions, eliminating the risk of conflicts and ensuring that the development environment is consistent and reproducible.

3.2.1 Python libraries

Pandas

Figure 3.14: Pandas Logo

Pandas is a fundamental Python library, designed specifically for data manipulation and analysis. Its strength is its ability to efficiently and intuitively manage structured data, such as those organized in tables, spreadsheets, or databases. The name itself comes from the econometric term "panel data," which refers to multidimensional datasets.

The library architecture is based on two main data structures:

- **The Series**, which can be thought of as a single column of data, similar to a one-dimensional array but with labels (indexes).
- **The DataFrame**, which is the most widely used structure and represents an entire two-dimensional table, consisting of rows and columns; each column of a DataFrame is, in fact, a Series.

The popularity of Pandas derives from the simplicity with which it allows you to perform complex operations; in fact, with a few lines of code, it is possible to read, explore, clean, and transform data. Furthermore, Pandas is optimized for working with large datasets and is a pillar of the Python data science ecosystem, integrating perfectly with other libraries. Finally, it offers robust features for importing and exporting data from a wide range of formats, such as CSV, Excel, JSON, and SQL databases. This makes it an indispensable tool for both exploratory data analysis (EDA) and the preprocessing phase required to prepare data for machine learning models.

Scikit-learn

Figure 3.15: Scikit-learn Logo

Scikit-learn has established itself as one of the go-to libraries in the Python ecosystem for machine learning, thanks in large part to a unified and consistent application programming interface (API). This means that, regardless of the algorithm used—be it a linear regression, a random forest, or

a support vector machine—the syntax for training and using the model remains almost identical. This design consistency greatly simplifies experimentation, allowing you to quickly test different approaches for the same problem without having to learn new implementations.

The library offers a comprehensive arsenal for supervised learning, including models for classification and regression problems, while fully supporting unsupervised learning by providing algorithms for clustering and dimensionality reduction. Another pillar of Scikit-learn is data preprocessing. The library integrates essential functionality to normalize, scale, and transform features before feeding them into models, which is often crucial to ensure that algorithms function correctly and achieve optimal performance. Finally, Scikit-learn provides a robust framework for model evaluation, as it includes utilities to easily divide data into training and test sets, to perform cross-validation, and to calculate a wide range of performance metrics.

Matplotlib

Figure 3.16: Matplotlib Logo

Matplotlib is a Python library that is considered a cornerstone for data visualization and is designed to give you complete and flexible control over the creation of static, animated, and interactive charts. Its importance is such that it is often referred to as the "base library" of Python visualization.

The main advantage of Matplotlib lies in its extraordinary versatility. It allows you to produce a wide range of charts: from the most standard ones such as line charts, bar charts, histograms, and scatter plots, to more complex visualizations such as heatmaps, pie charts, and three-dimensional plots. However, its real strength is the almost unlimited level of customization; in fact, the user has the ability to modify every single detail of a chart, such as colors, axis labels, titles, grids, line styles, and markers. In the field of data analysis, Matplotlib is a crucial tool that allows you to quickly translate data, such as data contained in a Pandas DataFrame, into a visual representation, making it easier to identify trends, correlations, or anomalies. An additional significant benefit is the ability to export charts in a variety of high-quality formats, making it the ideal solution for preparing figures for inclusion in reports, scientific papers, or presentations.

Seaborn

Figure 3.17: Seaborn Logo

Seaborn is a data visualization library for Python that operates as a high-level interface based on Matplotlib. It is designed to work seamlessly with Pandas' data structures, and its main purpose is to make creating statistical charts not only easier but also more aesthetically pleasing.

Seaborn's fundamental goal is to make it easy to visualize complex relationships between variables. It allows you to generate, with a few intuitive lines of code, sophisticated graphs that show

distributions, trends, and correlations, without requiring the user to manually define every stylistic detail such as colors, labels, or line styles. The resulting charts are generally more elegant and readable than standard Matplotlib ones; the aesthetic quality of the outputs is a distinguishing feature of Seaborn. In addition, it automatically handles many complex operations, such as adding legends or assigning different colors based on categorical variables.

Among the most common types of charts that can be produced with ease, we find:

- **Scatterplots:** to investigate the relationship between two variables;
- **Distribution charts** (such as column charts): to visualize how the data is distributed;
- **Boxplots:** to obtain a summary of the variability and distribution of the data;
- **Bar charts:** used to compare values between different categories, often with automatic aggregations.

SymPy

Figure 3.18: SymPy Logo

SymPy is one of the most important libraries for symbolic mathematics in Python. Unlike number libraries like NumPy, which work with numbers (floating-point approximations), the key feature of SymPy is its ability to work with symbols. This allows it to perform calculations exactly, just as you would on paper, manipulating algebraic expressions and finding analytical solutions instead of numerical ones.

It offers an incredibly wide range of tools for algebra and mathematical analysis. It allows you to define symbols (such as x, y or α) and to use them for tasks such as simplifying complex expressions, expanding polynomials, factorizing, and solving equations (both algebraic systems and differential equations). But its support also extends to all infinitesimal calculus: SymPy can compute limits, derivatives (partial and total), integrals (definite and indefinite), and series developments (such as Taylor series) symbolically.

Another strength is its "pretty printing" capability: the library can format the output in a clear and readable way, identical to how formulas would be written in a textbook or on a whiteboard (often using Unicode or generating LaTeX code). Moreover, SymPy doesn't just find the exact formula; it also provides tools to "translate" these symbolic expressions into high-performance numerical code (e.g., for NumPy or Fortran), allowing you to move from a theoretical solution to an efficient numerical calculation.

3.2.2 Experimental modeling pipeline/solution proposed

The scientific literature in the field of Defect Prediction for Industry 4.0 proposes numerous algorithmic solutions, often accompanied by excellent performance metrics. However, in the transition from theory to industrial application, it is fundamental to verify the reproducibility of such results, which is a pillar of the scientific method: a result is valid only if it can be independently replicated using the parameters and methods declared by the authors.

Table 3.1 shows the experimental replication of the four articles using the same dataset as this paper, searched through the DOI. The goal is not merely confirmatory, but critical. By reconstructing the algorithms in a controlled development environment, this work has three specific objectives:

- **Validation of Metrics**, i.e., verifying whether the declared accuracies, ranging from 86% to 96%, are obtainable by strictly applying the described methodologies.

- **Robustness Analysis**, namely assessing how the different models react to the strong imbalance of the dataset.

- **Identification of the Baseline**, establishing with empirical data the current real "state of the art" that will serve as the starting point and benchmark for the innovative algorithm to be proposed in Chapter 5.

All experiments were conducted using a unified development environment based on the Python language; the replication process followed a rigorous protocol for each article. The process began with a faithful preprocessing phase, where data were treated following the specific indications of each author; it continued with the configuration of hyperparameters, which were implemented exactly as those cited in the papers (e.g., $K = 10$ for KNN) to minimize confounding variables; furthermore, the same evaluation metrics were used for each replication, namely Accuracy, Precision, Recall, and F1-Score, with the addition of ROC-AUC and Cohen's Kappa for the more advanced studies, in order to have a complete view of performance on both the majority class and the critical defect class.

The choice to replicate these specific four works responds to the need to cover the main families of algorithms used today in industry, from the "complex ensemble" approach to the "distance-based" one, from "Tree-based" to "Deep Learning," with the aim of obtaining a clear overview of how the models perform on this specific dataset and evaluating which ones work best, which is useful for the subsequent construction of the algorithm that will be proposed in the next chapter. Below, the results obtained for each article are presented in a table, highlighting the confirmations, where the replication was successful, and the discrepancies, where the replicated results deviate from the declared ones, providing a technical analysis of the probable causes for the latter.

Table 3.1: Comparison of Reference Studies and Replication Results

Reference Study	**Proposed Algorithm**	**Declared Metrics**	**Critical Notes**
1. Mohammad Reza Mehregan et al. [45]	**Stacking Ensemble**: the decisions of four base learners are combined: XGBoost, LightGBM, CatBoost, and ANN. These predictions are then aggregated by a Meta-learner, Random Forest, which makes the final decision.	Accuracy: 96.06% Precision: 96.00% Recall: 96.00% ROC-AUC: 0.87 MCC: 0.78 Kappa: 0.78	Very robust model, but characterized by high computational complexity, justified by the need to capture every nuance of the data, which results in a long training time.
2. Muhammad David and Muhammad Azka Firdaus [11]	**KNN**: configured with $K = 10$ and Minkowski distance as the metric. The authors declare the use of a Random Under-Sampling (RUS) technique to reduce the majority class and balance the dataset before training. The objective is to verify if a "lightweight" geometric model can compete with more complex algorithms.	Accuracy: 86.41% Precision: 91.23% Recall: 96.78% ROC-AUC: 0.53 MCC: 0.03 Kappa: 0.02	The experimental replication found that the result is not obtained according to the reported methodologies, but an accuracy of 89% is approached by not applying balancing techniques to the dataset.
3. Gabriel Marín Díaz [46]	**Optimized XGBoost**: the predictive model is integrated with an advanced Explainable AI (XAI) module, using techniques such as SHAP, LIME, and ELI5 to identify which variables most influence the decision. Parallelly, the identification of latent profiles in the data is studied using algorithms such as Fuzzy C-Means and K-Means.	Accuracy: 95.37% Precision: 94.99% Recall: 99.27% ROC-AUC: 0.88 MCC: 0.84 Kappa: 0.83	Excellent balance between performance and interpretability. It serves as confirmation of the reference model to be used for prediction on this dataset.
4. Philipp Goetzinger [47]	**Hybrid Comparison**: two distinct architectures are put into competition on the same dataset, proposing a comparative analysis: on one hand, an XGBoost model is implemented; on the other, a Feed Forward Neural Network is developed. It also focuses on the identification of causal factors of defects, highlighting which variables negatively impact quality.	XGBoost: Accuracy: 95.37% Precision: 95.02% Recall: 97.98% ROC-AUC: 0.89 MCC: 0.83 Kappa: 0.82	The results of the XGBoost model are reported, as it has proven to be successful, confirming that on these tabular data, tree-based models (XGB) outperform Neural Networks (FNN).

In this study, a classification architecture based on a Heterogeneous Stacking Ensemble approach is proposed. The decision to combine multiple learning algorithms, or Base Learners, stems from the need to overcome the limitations of individual classifiers, especially in contexts such as the dataset

under examination, which is characterized by a severe class imbalance ("Reverse Imbalance") and the imperative to minimize false negatives within a Zero-Defect Manufacturing framework. The proposed framework is designed to maximize robustness and generalization capabilities by integrating advanced preprocessing techniques, dynamic imbalance management, and adaptive optimization of the decision threshold; unlike traditional approaches based on single classifiers, it combines the generalization capabilities of Bagging, the precision of Boosting, and the non-linear approximation capabilities of Neural Networks, all orchestrated by a logistic Meta-Learner. The entire pipeline was designed with a rigorous validation protocol to prevent Data Leakage. Data Leakage, which consists of the infiltration of information from the test set during the training phase, is indeed one of the critical aspects to prevent in the design of Machine Learning pipelines. To mitigate this risk, the entire processing chain was encapsulated within the cross-validation using ImbPipeline from the imbalanced-learn library, ensuring that the transformations are learned exclusively on the training folds during cross-validation.

The numerical features of the dataset exhibit heterogeneous orders of magnitude; therefore, a StandardScaler ($\mu = 0, \sigma = 1$) was applied as the first step of the pipeline. This operation is fundamental for two reasons:

- ANN Convergence: Neural networks require normalized data for stable weight updates via backpropagation; without this step, variables with larger magnitudes would dominate the calculation of gradients and distances.

- SMOTE Effectiveness: The SMOTE algorithm calculates nearest neighbors based on Euclidean distance; without scaling, features with higher absolute values would dominate the distance calculation, generating geometrically incorrect synthetic samples.

To counteract class imbalance, the SMOTE algorithm was employed, which, unlike simple Random Oversampling that duplicates existing data thereby increasing the risk of overfitting, generates new synthetic samples by interpolating between nearest neighbors of the minority class in the feature space. Unlike many studies in the literature that apply it to the entire dataset prior to splitting, resulting in a methodological error known as Data Leakage, in this pipeline, SMOTE is embedded within the ImbPipeline. This ensures the generation of synthetic data solely on the training folds, leaving the test folds intact and representative of the true distribution. The optimal configuration of the k parameter (number of neighbors) underwent hyperparameter tuning within the nested cross-validation, resulting in optimal values between $k = 3$ and $k = 5$, thus preserving the local structure of defects without introducing excessive noise.

The core of the system is a two-level Stacking classifier.

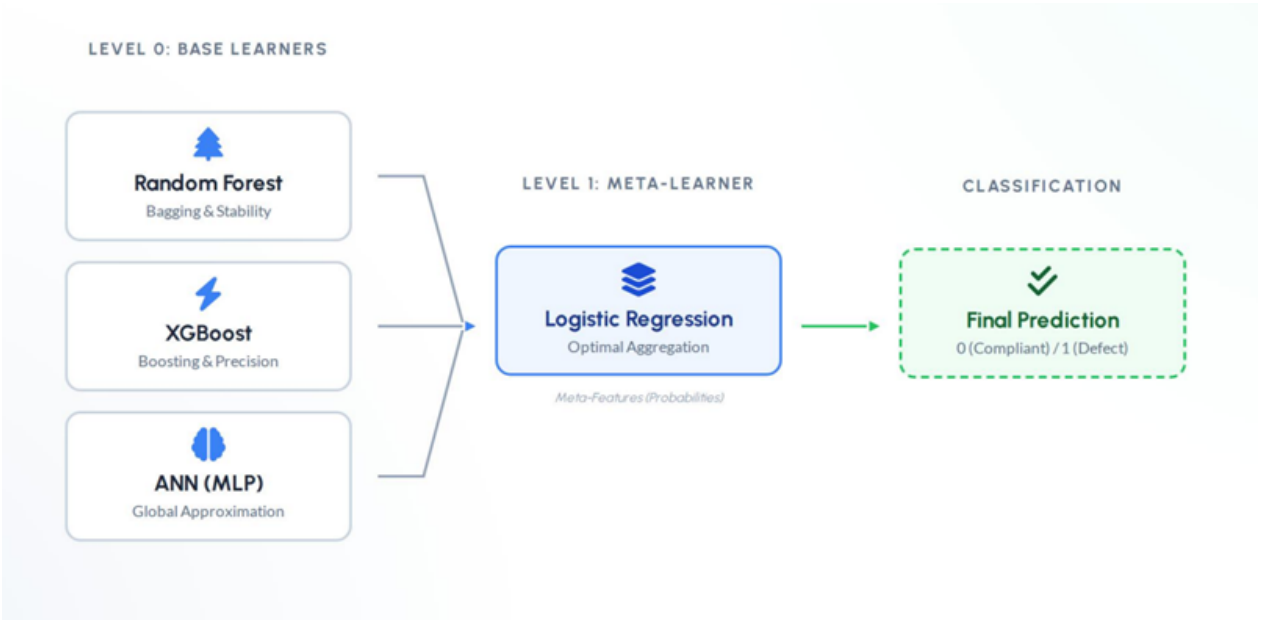

Figure 3.19: Schematic of the proposed stacking architecture

Level 0: Heterogeneous Base Learners

At the first level of the Stacking architecture, three models belonging to different algorithmic families were selected to ensure maximum diversity in the ensemble and to maximize error diversity:

Random Forest (Bagging): The first Base Learner selected for Level 0 of the Stacking is the Random Forest algorithm, which aims to reduce variance and improve robustness to outliers, adopting the Bagging (Bootstrap Aggregating) paradigm: by training a large number of independent decision trees on randomly drawn samples with replacement (bootstrap samples) and aggregating their results, the model intrinsically mitigates the overfitting problem and proves particularly robust against the presence of outliers or noise in the manufacturing dataset. To ensure the statistical stability of the prediction without incurring computational bottlenecks, the hyperparameter regarding the number of estimators was set to 300 trees, "n_estimators=300". The most innovative element in this algorithmic instance concerns the management of class imbalance; indeed, rather than applying a global and static penalty, the "class_weight='balanced_subsample"' strategy was implemented, which calculates class frequencies and their respective corrective weights not on the entire global training set, but dynamically within each individual bootstrap sample provided to the respective tree. Given the Reverse Imbalance characteristics of the problem under examination, this calculation granularity ensures that even trees exposed to marginal fractions of the minority class apply locally maximized corrective weights. This approach translates into a locally adaptive Cost-Sensitive Learning strategy, significantly improving the recall on the minority class compared to a conventional balancing approach.

XGBoost (Boosting): The second Base Learner integrated into the architecture is XGBoost, an advanced implementation of the Gradient Boosting paradigm, which, unlike Random Forest that reduces variance by creating trees in parallel, operates by constructing an ensemble of decision trees sequentially. Each new estimator is trained to minimize the gradient of the loss function ("logloss") calculated on the residuals of the previous tree; this iterative error correction mechanism makes the algorithm exceptionally proficient at reducing systematic bias and modeling non-linear boundary patterns between classes. To mitigate the intrinsic risk of overfitting in high-capacity models, the

architecture was configured following the Slow Learning principle. Therefore, in a separate script, a pre-tuning phase was conducted, leading to the selection of a moderately high maximum tree depth, "max_depth=8", to capture complex interactions among features, counterbalanced by a very conservative learning rate, "learning_rate=0.03", distributed over a large number of iterations, "n_estimators=400". However, the most critical methodological aspect regarding XGBoost lies in the management of Reverse Imbalance; generally, algorithms of this family tend to asymptotically favor the majority class, but, to counter this phenomenon algorithmically, a direct intervention on the cost function was made via the "scale_pos_weight" hyperparameter. Since the class of interest, "Defects" (label 1), represents the numerical majority, the parameter was calculated as the ratio between negative and positive instances (N_{neg}/N_{pos}). The result, being a scale factor less than one, acts as a penalizer for the majority class: it scales down the gradients generated by defective samples, mathematically forcing the algorithm to concentrate its learning capacity on the minority class samples. This inverse Cost-Sensitive Learning implementation ensures that the model does not trivially optimize global accuracy while ignoring minority anomalies.

Multi-Layer Perceptron (Neural Network): The third and final Base Learner comprising Level 0 of the Stacking is an Artificial Neural Network, specifically a Multi-Layer Perceptron (MLP) that addresses the theoretical principle of Ensemble Diversity through global non-linear approximation. While Random Forest and XGBoost approximate the classification function through orthogonal hyperplanes, with rigid partitioning of the feature space, the neural network models complex interactions by combining features in a continuous and non-linear manner; this distinct difference in inductive bias ensures that the errors made by the ANN are statistically uncorrelated with those of the trees, enriching the latent space upon which the Meta-Learner will operate. Architecturally, a separate script was used here as well for pre-tuning, aimed at countering the high risk of overfitting induced by class imbalance; a deliberately shallow and convergent topology was therefore chosen, featuring two hidden layers consisting of 32 and 16 neurons respectively ("hidden_layer_sizes=(32, 16)"). This funnel-like structure acts as a regularizing filter, forcing the network to discard statistical noise and extract exclusively the macro-representations, or latent features, essential for class discrimination; non-linearity is guaranteed by the use of the ReLU activation function, which ensures numerical stability during backpropagation. Finally, the control of adaptation to the training data was entrusted to a dynamic strategy via the "early_stopping=True" parameter. In combination with a calibrated initial learning rate ("learning_rate_init=0.01") and a maximum limit of 500 iterations, this mechanism internally monitors the validation error and halts training as soon as the model's generalization capacity stops improving, guaranteeing a further line of defense against the memorization of majority instances.

Level 1: The Meta-Learner

The proposed Stacking architecture culminates in a Level 1 Meta-Learner, tasked with optimally combining the predictions of the three base classifiers. The transfer of information between the two levels occurs by leveraging the predicted probabilities method ("predict_proba"): the Meta-Learner does not have access to the original feature space, but rather to a very low-dimensional latent space, consisting exclusively of the confidence scores—continuous values in the range $[0, 1]$—emitted by Random Forest, XGBoost, and the ANN. Logistic Regression was selected as the algorithm for this level; this design choice, seemingly contrary to the complexity of the Level 0 models, is dictated by the need to prevent overfitting phenomena. Indeed, since the Base Learners have already learned the complex non-linear relationships from the raw data, the Meta-Learner's task is reduced to a

calibration problem, i.e., identifying the linear coefficients that minimize the log-loss by combining the opinions of the "experts". The use of a high-capacity model, such as XGBoost, was also attempted in this phase, but relatively poorer results were obtained, explained by an over-adaptation to the training predictions. Furthermore, it should be noted that the input data to Level 1 were not subjected to further standardization procedures, as they are natively homogeneous and confined within the probabilistic range. During the nested cross-validation, targeted optimization was performed on the C parameter of the Logistic Regression, representing the inverse of regularization strength; by exploring the neighborhood of moderate values ($C \in [0.5, 1.0, 2.0]$), it was ensured that the L2 penalty forced the model to distribute the decision weights equitably, preventing the Meta-Learner from collapsing by assigning exclusive trust to a single Base Learner and thus preserving the intrinsic advantage of the Ensemble architecture. Experimental results showed that C values close to 1.0 or 2.0 offer the best compromise, allowing the meta-model to trust the base learners' predictions without overfitting the training data.

3.2.3 Solution validation and optimization

Nested Cross-Validation

To ensure a rigorous evaluation of the proposed architecture's generalization performance, overcoming the limitations of the standard hold-out validation frequently used in the literature, a stratified Nested Cross-Validation with 5 Outer Folds and 3 Inner Folds was implemented. The search and optimization of hyperparameters, such as the number of neighbors k for the SMOTE algorithm and the regularization parameter C of the Meta-Learner, was strictly relegated to the inner validation loop (Inner Loop) using a "RandomizedSearchCV". This approach ensures that the model is tested in the outer loop (Outer Loop) on strictly independent data partitions that were never observed during the tuning phase. The conceptual separation between Model Selection and Model Assessment fundamentally eliminates the risk of Optimism Bias, providing an empirical and reliable estimate of how the system would operate in a real manufacturing environment.

Threshold Tuning (Threshold Calibration)

The combined use of Cost-Sensitive Learning strategies and synthetic oversampling deeply alters the posterior probability distribution emitted by the Meta-Learner. Consequently, the standard assumption of adopting a fixed decision threshold at 0.5 proves suboptimal, as it is not aligned with the new probabilistic densities induced by rebalancing. To overcome this probabilistic distortion, an adaptive Threshold Tuning algorithm was introduced, whereby, at the end of training for each fold, the system extracts the out-of-fold probabilistic predictions from the training set using the "cross_val_predict" function. An exhaustive grid search is conducted on these probabilities to identify the optimal operational threshold t_{opt} that maximizes the Matthews Correlation Coefficient (MCC). The MCC metric was chosen as the objective function because it is the only one capable of returning a robust and symmetric correlation measure, simultaneously balancing true positives, true negatives, false positives, and false negatives. Once the optimal threshold t_{opt} is calculated exclusively on the training data, thus preventing any form of probabilistic Data Leakage, it is deterministically applied to the test set probabilities for final classification; this design choice effectively decouples the model's discriminative capability from the operational decision threshold, allowing the architecture to self-calibrate based on the specific local distributions of each fold.

Chapter 4

Experimental Results and Evaluation

4.1 Selection of Evaluation Metrics

The evaluation of a classification model in an industrial setting, especially with this type of dataset, requires a careful selection of metrics. For this reason, the performance of the proposed architecture was measured using a set of robust and complementary metrics:

- **Matthews Correlation Coefficient (MCC):** Chosen as the primary reference metric. Ranging between -1 and +1, the MCC returns a high value only if the model achieves good results across all four categories of the confusion matrix (TP, TN, FP, FN). It is considered the gold standard for imbalanced datasets.

- **Accuracy:** Calculated as the ratio of correct responses to total data, it provides a global measure of the model's ability to correctly classify observations, considering both classes simultaneously. Its value ranges between 0 and 1, where 1 indicates perfect classification, and is expressed as a percentage.

- **Cohen's Kappa:** Measures the agreement between the model's predictions and the true labels, correcting for the probability of agreement occurring by pure chance.

- **Recall on Defects:** In a Zero-Defect Manufacturing perspective, the absolute priority is to minimize False Negatives—undetected defects that reach the customer; Recall exactly quantifies this capability.

- **Precision and F1-Score:** To monitor the incidence of False Positives, i.e., the rejection of good parts, and overall balance.

- **ROC AUC:** The Area Under the ROC Curve evaluates the global discriminative capability of the model independently of the operational decision threshold.

4.2 Quantitative Results of Nested Cross-Validation

The execution of the Nested Cross-Validation, with 5 outer folds, provided a rigorous and optimism bias-free estimate of the system's generalization performance. Figure 5.1 summarizes the mean results and relative standard deviations calculated on the test folds.

```
=== RISULTATI ===
Accuracy        : 0.9599 (+/- 0.0094)
Precision       : 0.9607 (+/- 0.0103)
Recall          : 0.9930 (+/- 0.0029)
F1              : 0.9766 (+/- 0.0053)
ROC AUC         : 0.8904 (+/- 0.0332)
MCC             : 0.8439 (+/- 0.0394)
Kappa           : 0.8376 (+/- 0.0437)
```

Figure 4.1: Mean results and relative standard deviations calculated on the test folds.

As can be seen in 4.1, the model achieves high "Accuracy," averaging "0.9599". However, in this discussion, it was deliberately chosen not to use it as the primary performance indicator since, in the presence of highly imbalanced datasets, it is subject to the well-known Accuracy Paradox: a trivial classifier that systematically predicts the majority class would still obtain a misleadingly high score, masking the total inability to identify the minority class. For this reason, the analysis of the model's value shifts to metrics structurally immune to this distortion, demonstrating an excellent performance level of the model. The Recall value (99.30%) immediately stands out, which, in an industrial context oriented towards Zero-Defect Manufacturing, is critical, as it indicates that the model is capable of intercepting nearly all defective components, drastically minimizing the risk of a non-conforming product reaching the end user. This data, combined with a Precision of 96.07%, confirms that the system is not merely "over-notifying" defects, but maintains high operational reliability, limiting False Positives. Furthermore, the intrinsic robustness of the architecture is certified by the values of the Matthews Correlation Coefficient (MCC) and Cohen's Kappa, equal to "0.8439" and "0.8376" respectively. Achieving such high scores in these two metrics confirms that the integration of SMOTE, weight correction in XGBoost and Random Forest, coupled with adaptive threshold balancing, enabled the model to perfectly discriminate both classes, without simply "leaning" in favor of the majority. Further confirmation of the proposed architecture's quality derives from the analysis of the Area Under the ROC Curve (ROC-AUC), which averages "0.8904"; unlike the point metrics discussed previously, which strictly depend on the threshold value adopted for final classification, the ROC AUC evaluates the intrinsic ability of the model to separate the two classes across all possible thresholds. Obtaining a value close to 0.9 demonstrates that the Meta-Learner produces extremely coherent and well-calibrated probability estimates; this excellent ranking capability constitutes the fundamental mathematical basis that allowed the subsequent Threshold Tuning algorithm to identify an optimal cutoff point, maximizing the operational effectiveness of the system. Finally, it is crucial to observe the narrow standard deviations, which are minimal across the 5 independent folds of the Cross-Validation, demonstrating high architectural stability: the model is not sensitive to specific and fortuitous combinations of data, but learns structural and repeatable patterns, an essential requirement for actual deployment in production.

4.3 Visual Analysis of the "Best Fold"

```
Fold 4/5
  -> Migliori parametri trovati: {'smote__k_neighbors': 4, 'model__final_estimator__lr__C': 2.0}
  -> MCC Fold: 0.8699 (Best Params: {'smote__k_neighbors': 4, 'model__final_estimator__lr__C': 2.0})
  Salvataggio grafici (MCC: 0.8699)...
```

Figure 4.2: Inner Loop training log for the Best Fold

To gain a deeper understanding of the model's decision-making behavior under operational conditions, it is useful to graphically analyze the performance relating to the best iteration of the cross-validation (Best Fold). As highlighted by the Inner Loop training logs, the performance peak was recorded during Fold 4 4.2, reaching an MCC of 0.8699. For this specific iteration, hyperparameter optimization selected a configuration with $k = 4$ for the SMOTE algorithm and an inverse regularization $C = 2.0$ for the Meta-Learner's Logistic Regression.

Confusion Matrix and Industrial Safety

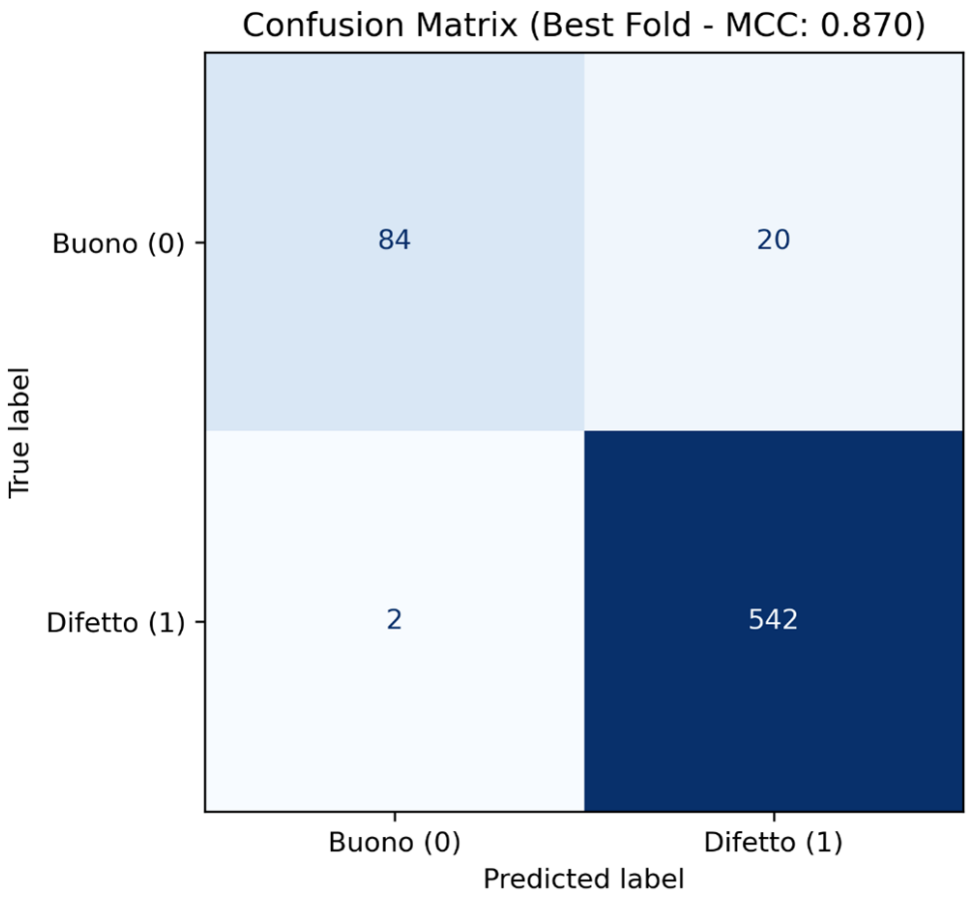

Figure 4.3: Confusion Matrix on the test set

Figure 4.3 shows the Confusion Matrix generated on the test set of this optimal fold. The most relevant aspect from an industrial application perspective lies in the lower-left quadrant: the model generated only 2 False Negatives out of a total of 542 actual defective components, translating to a defect interception rate of 99.6%. This asymmetric behavior is the direct and intended result

of the MCC-oriented Threshold Tuning strategy and the Cost-Sensitive Learning applied during the training phase. The system, in fact, accepts a moderate and manageable number of False Positives, 20 precautionary rejections of sound parts out of a total of 104, in order to guarantee the near-total blockage of critical defects. From a Zero-Defect Manufacturing perspective, where the cost of delivering a defective part to the customer is vastly higher than the cost of a false alarm manageable internally, this matrix represents the perfect decision-making compromise and fully satisfies the quality requirements of modern production.

Discriminative Capability: ROC and Precision-Recall Curves

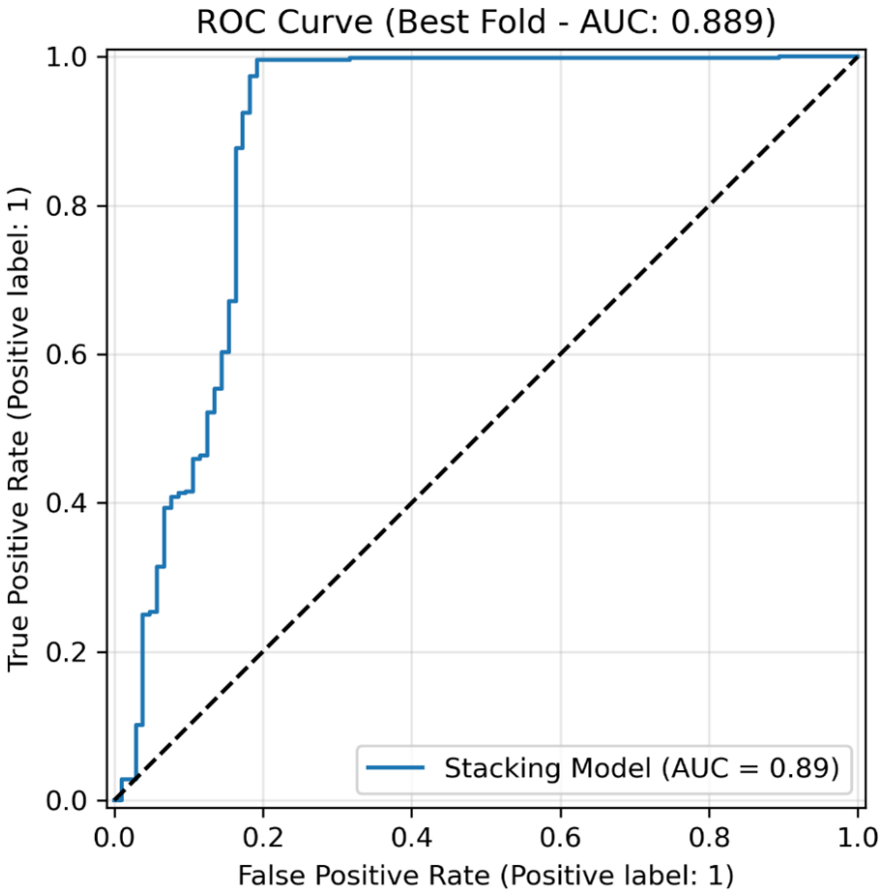

Figure 4.4: Ensemble ROC Curve

The ROC Curve 4.4 visually highlights the discriminative effectiveness of the Ensemble; given the rapid ascent of the curve toward the upper-left corner, specifically the sudden increase in the True Positive Rate for False Positive Rate values below 0.2, this behavior translates to an Area Under the Curve (AUC) of 0.889. This trend confirms that the Meta-Learner is capable of correctly ordering probabilities, cleanly separating the distribution of good parts from that of defects, even prior to the application of an operational cutoff threshold.

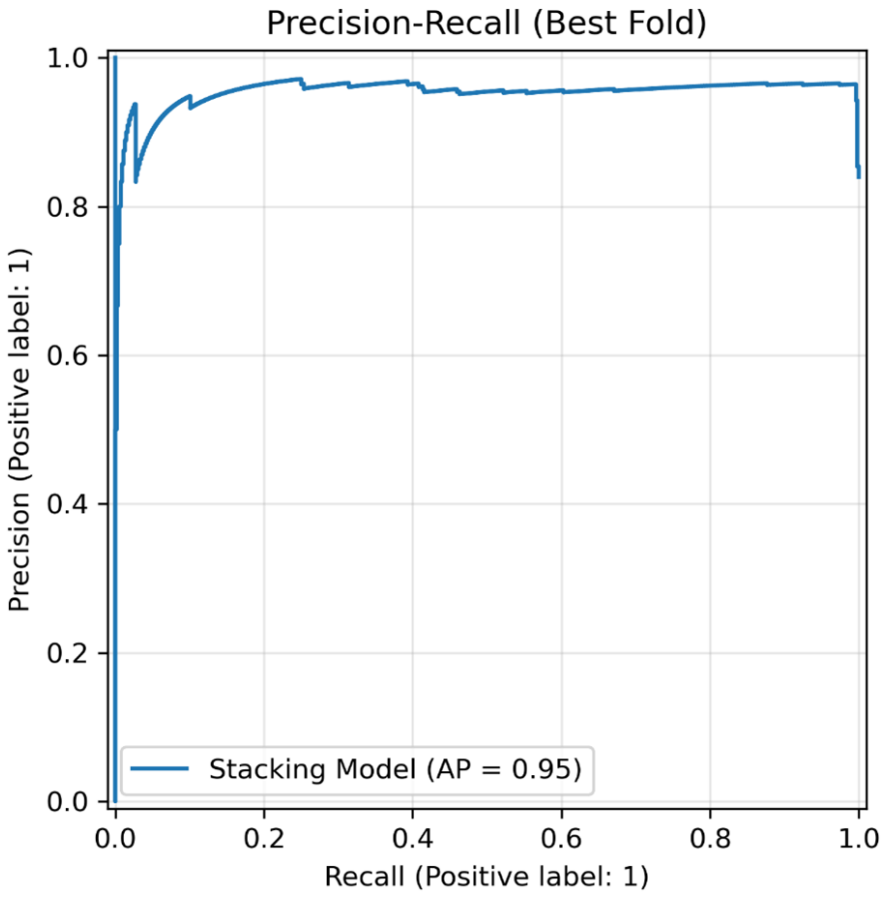

Figure 4.5: Precision-Recall Curve

Completing the analysis on discriminative capability, the Precision-Recall Curve 4.5 provides a perspective specifically focused on the predictive stability of the class of interest, label 1. With an overall Average Precision (AP) score of 0.95, the graph documents near-ideal behavior for the manufacturing context. Observing the topology of the curve, it is evident how it remains flat and tenaciously adhering to the upper limit of the y-axis (Precision > 0.95) for almost the entire Recall spectrum. From an operational and engineering standpoint, this "delayed step" trend assumes crucial significance: the Stacking architecture is capable of progressively increasing its sensitivity, reliably intercepting up to over 95% of real defects, without suffering any significant degradation in terms of Precision. The vertical collapse of the curve occurs quite physiologically, exclusively at the extreme right edge of the graph. Only when the classifier is mathematically forced to intercept the absolute totality of anomalies (Recall $= 1.0$) in the most complex marginal cases, does the system begin to generate False Positives. This dynamic unequivocally certifies the exceptional robustness of the proposed pipeline, as the model succeeds in isolating and identifying defective patterns ensuring the highest quality standards, without needlessly compromising production yield volumes with premature false alarms.

4.4 Conclusions and Comparison with the State of the Art

To objectively validate the results obtained and measure the scientific contribution of the proposed architecture, the pipeline's performance was compared with the most recent and relevant studies in the literature, previously analyzed in section 3.1. The comparative results are visualized in the

graph in Figure 5.6, which traces the performance profiles of the various approaches across the entire set of evaluation metrics.

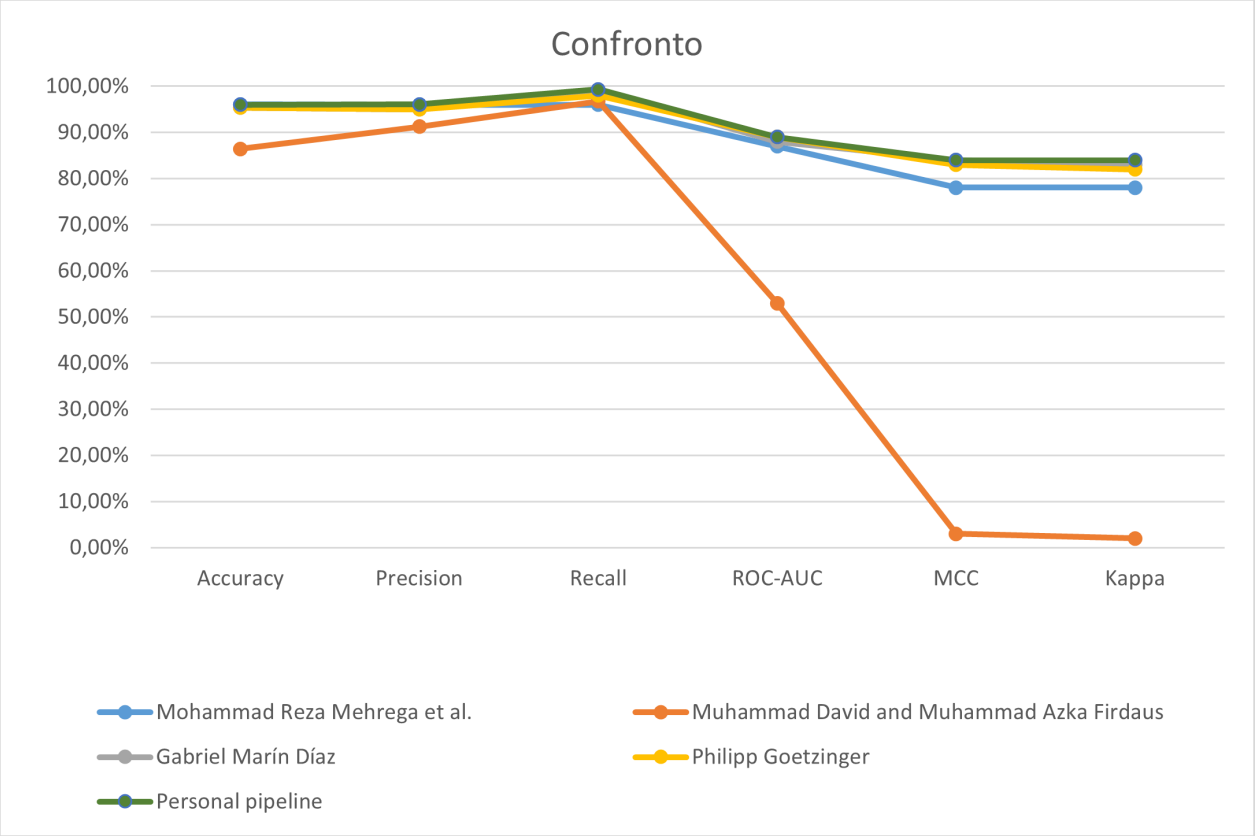

Figure 4.6: Comparative results and performance profiles of the different approaches

From the analysis of the graph, fundamental considerations emerge not only regarding the positioning of the proposed pipeline, designated as Personal pipeline (green line), but also concerning the methodological pitfalls of the dataset. Indeed, the graph emblematically highlights the risk associated with a superficial evaluation of imbalanced data. It can be noted that the study by Muhammad David and Muhammad Azka Firdaus [11] (orange line) shows seemingly valid levels of Accuracy and Recall, but records a vertical collapse corresponding to ROC-AUC, MCC, and Kappa (close to zero); this profile is the unmistakable symptom of a model distorted by class imbalance, which merely predicts the majority class constantly, which is why the inclusion of robust metrics such as MCC is essential for unmasking operationally ineffective models. Regarding the approaches of Mehregan et al. [45] (blue line), Marín Díaz [46] (grey line), and Goetzinger [47] (yellow line), these present much more solid and balanced profiles, confirming themselves as high-level benchmarks. However, the Personal pipeline systematically dominates the comparison, positioning itself at the upper bound of the graph across the entire spectrum of the metrics. The gap, although seemingly subtle in "easy" metrics like Recall, becomes pronounced and statistically significant precisely on the most severe metrics: the MCC and Cohen's Kappa; Maintaining these metrics above 84%, clearly distancing the competition, demonstrates a real superiority in the ability to separate classes without relying on chance. This positioning as the new State of the Art is not fortuitous, but is the direct result of the rigorous architectural choices described in the previous chapters. In fact, while many studies naively apply oversampling to the entire dataset, risking overfitting, the encapsulation of SMOTE within a nested validation guaranteed that the performance of the Personal pipeline was authentic and unvitiated by Data Leakage. Furthermore, the integration of adaptive Threshold Tuning based on MCC made it possible to extract the maximum discriminative value from the

Meta-Learner, correcting the probabilistic distortion ignored by other authors, and the heterogeneity of the Stacking Ensemble (RF, XGB, ANN) provided superior decision-making flexibility compared to the use of isolated algorithms. In conclusion, the architecture engineered in this study not only equals the best standards of current academic research, but surpasses them, offering a safer, more robust, and immediately scalable solution for modern industrial quality control systems.

Chapter 5

Conclusions and Future Developments

The primary objective of this thesis work was the design, development and validation of an advanced Machine Learning architecture for the identification of defects in the modern industrial context, oriented toward the Zero-Defect Manufacturing paradigm, which imposes extremely strict reliability requirements, made even more complex by the inherently imbalanced nature of production data. To address these challenges, a pipeline based on a Heterogeneous Stacking Ensemble was proposed, integrating the predictive capabilities of tree-based models and neural networks, orchestrated by a logistic Meta-Learner. The most relevant scientific and technical contributions of this study can be summarized as follows:

- Prevention of Data Leakage: The encapsulation of the standardization and synthetic oversampling (SMOTE) phases within a rigorous Nested Cross-Validation, which guaranteed the absolute validity and replicability of the obtained metrics, overcoming the methodological fallacies present in several sector studies.
- Advanced Imbalance Management: The adoption of targeted Cost-Sensitive Learning techniques (such as the inverted scale_pos_weight hyperparameter in XGBoost and balanced_subsample in Random Forest) forced the architecture to focus on the minority class of interest without degrading global stability.
- Dynamic Threshold Calibration: The implementation of a Threshold Tuning algorithm based on the maximization of the MCC exclusively on the training set allowed for the correction of probabilistic distortions (Probability Shift), decoupling the ranking capacity from the operational decision.

The experimental results have amply validated the proposed approach, since, distancing itself from the misleading Accuracy metric, the system recorded an average Matthews Correlation Coefficient (MCC) of 0.84 (with peaks of 0.87), an excellent class separation (ROC-AUC of 0.89) and, above all, a defect interception rate (Recall) exceeding 99%. The comparison with the State of the Art confirmed the superiority of the pipeline, positioning it as a highly competitive solution ready for industrial application.

5.1 Future Developments

Although the proposed architecture has achieved excellent performance, the transition from a research environment to a real production system opens the way to multiple promising future developments; one of these could be a Real-Time implementation, a logical next step consisting of the deployment of the model directly on the production line; indeed, by optimizing the code via containerization and microservices, the classifier could analyze sensor data in real-time, halting machinery or alerting operators in fractions of a second at the first sign of an anomaly. In a manufacturing context,

an operator could trust the algorithm through Explainable AI (XAI) for interpretability; in fact, the integration of Explainable AI libraries, such as SHAP (SHapley Additive exPlanations) or LIME, would allow "opening the black box" of Stacking and, instead of merely outputting a "defect" prediction, the system could explain which sensors or which features caused the anomaly, greatly facilitating preventive maintenance. Finally, since factories are dynamic environments, machinery wears out, and raw materials change, the current static architecture—i.e., trained on a historical snapshot of data—is not well suited to the new concept of Industry 4.0. For this reason, "Continual Learning and Concept Drift Management" is proposed, through which approaches such as Active Learning or Continual Learning could be integrated, allowing the model to incrementally retrain on new data, automatically adapting to the physiological drifts of production processes without forgetting historical patterns.

Bibliography

[1] G. Sariyer, S. K. Mangla, Y. Kazancoglu, C. Ocal Tasar, and S. Luthra, "Data analytics for quality management in industry 4.0 from a msme perspective," *Annals of Operations Research*, vol. 350, no. 2, pp. 365–393, 2025.

[2] H. Kagermann, W.-D. Lukas, and W. Wahlster, "Industrie 4.0: Mit dem internet der dinge auf dem weg zur 4. industriellen revolution," *VDI nachrichten*, vol. 13, no. 1, pp. 2–3, 2011.

[3] D. C. Montgomery and C. M. Borror, "Systems for modern quality and business improvement," *Quality Technology & Quantitative Management*, vol. 14, no. 4, pp. 343–352, 2017.

[4] A. Luckow, K. Kennedy, M. Ziolkowski, E. Djerekarov, M. Cook, E. Duffy, M. Schleiss, B. Vorster, E. Weill, A. Kulshrestha, *et al.*, "Artificial intelligence and deep learning applications for automotive manufacturing," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 3144–3152, IEEE, 2018.

[5] O. Nalbach, C. Linn, M. Derouet, and D. Werth, "Predictive quality: Towards a new understanding of quality assurance using machine learning tools," in *International conference on business information systems*, pp. 30–42, Springer, 2018.

[6] T. Wuest, C. Irgens, and K.-D. Thoben, "An approach to monitoring quality in manufacturing using supervised machine learning on product state data," *Journal of Intelligent Manufacturing*, vol. 25, no. 5, pp. 1167–1180, 2014.

[7] D. Lieber, B. Konrad, J. Deuse, M. Stolpe, and K. Morik, "Sustainable interlinked manufacturing processes through real-time quality prediction," in *Leveraging Technology for a Sustainable World: Proceedings of the 19th CIRP Conference on Life Cycle Engineering, University of California at Berkeley, Berkeley, USA, May 23-25, 2012*, pp. 393–398, Springer, 2012.

[8] F. Steinberg, P. Burggräf, J. Wagner, B. Heinbach, T. Saßmannshausen, and A. Brintrup, "A novel machine learning model for predicting late supplier deliveries of low-volume-high-variety products with application in a german machinery industry," *Supply Chain Analytics*, vol. 1, p. 100003, 2023.

[9] R. E. Kharoua, " predicting manufacturing defects dataset," 2024.

[10] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[11] M. David and M. A. Firdaus, "Defect rate prediction in manufacturing process using k-nearest neighbor algorithm," *International Journal of Informatics, Economics, Management and Science*, vol. 3, no. 2, pp. 161–173, 2024.

[12] Q. P. He and J. Wang, "Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes," *IEEE transactions on semiconductor manufacturing*, vol. 20, no. 4, pp. 345–354, 2007.

[13] R. O. Duda and P. E. Hart, *Pattern classification and scene analysis*. New York: Wiley, 1973.

[14] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine learning*, vol. 29, no. 2, pp. 103–130, 1997.

[15] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, pp. 41–46, Seattle, USA, 2001.

[16] N. Zhang, L. Wu, J. Yang, and Y. Guan, "Naive bayes bearing fault diagnosis based on enhanced independence of data," *Sensors*, vol. 18, no. 2, 2018.

[17] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 20, no. 2, pp. 215–232, 1958.

[18] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013.

[19] M. Sharp, R. Ak, and T. Hedberg Jr, "A survey of the advancing use and development of machine learning in smart manufacturing," *Journal of manufacturing systems*, vol. 48, pp. 170–179, 2018.

[20] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[21] T. Chen, V. Sampath, M. C. May, S. Shan, O. J. Jorg, J. J. Aguilar Martin, F. Stamer, G. Fantoni, G. Tosello, and M. Calaon, "Machine learning in manufacturing towards industry 4.0: From 'for now'to 'four-know'," *Applied Sciences*, vol. 13, no. 3, p. 1903, 2023.

[22] S. Wang, Y. Cui, Y. Song, C. Ding, W. Ding, and J. Yin, "A novel surface temperature sensor and random forest-based welding quality prediction model," *Journal of Intelligent Manufacturing*, vol. 35, no. 7, pp. 3291–3314, 2024.

[23] Y.-j. JI, X.-y. YONG, Y.-l. LIU, and S.-x. LIU, "Random forest based quality analysis and prediction method for hot-rolled strip," *Journal of Northeastern University (Natural Science)*, vol. 40, no. 1, p. 11, 2019.

[24] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[25] S. Lee, J. Park, N. Kim, T. Lee, and L. Quagliato, "Extreme gradient boosting-inspired process optimization algorithm for manufacturing engineering applications," *Materials & Design*, vol. 226, p. 111625, 2023.

[26] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining (ICDM '08)*, pp. 413–422, IEEE, 2008.

[27] G. A. Susto, A. Beghi, and S. McLoone, "Anomaly detection through on-line isolation forest: An application to plasma etching," in *2017 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, pp. 89–94, IEEE, 2017.

[28] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[29] T. T. Ademujimi, M. P. Brundage, and V. V. Prabhu, "A review of current machine learning techniques used in manufacturing diagnosis," in *IFIP International Conference on Advances in Production Management Systems*, pp. 407–415, Springer, 2017.

[30] A. Widodo and B.-S. Yang, "Support vector machine in machine condition monitoring and fault diagnosis," *Mechanical systems and signal processing*, vol. 21, no. 6, pp. 2560–2574, 2007.

[31] M. Onel, C. A. Kieslich, and E. N. Pistikopoulos, "A nonlinear support vector machine-based feature selection approach for fault detection and diagnosis: Application to the tennessee eastman process," *AIChE Journal*, vol. 65, no. 3, pp. 992–1005, 2019.

[32] H. J. Shin, D.-H. Eom, and S.-S. Kim, "One-class support vector machines—an application in machine fault detection and classification," *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 395–408, 2005.

[33] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.

[34] X. Liu, L. Ma, and J. Mathew, "Machinery fault diagnosis based on fuzzy measure and fuzzy integral data fusion techniques," *Mechanical Systems and Signal Processing*, vol. 23, no. 3, pp. 690–700, 2009.

[35] C. Li, M. Cerrada, D. Cabrera, R. V. Sanchez, F. Pacheco, G. Ulutagay, and J. Valente de Oliveira, "A comparison of fuzzy clustering algorithms for bearing fault diagnosis," *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 6, pp. 3565–3580, 2018.

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[37] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice hall PTR, 1994.

[38] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[39] R. E. Andersen, S. Madsen, A. B. Barlo, S. B. Johansen, M. Nør, R. S. Andersen, and S. Bøgh, "Self-learning processes in smart factories: Deep reinforcement learning for process control of robot brine injection," *Procedia Manufacturing*, vol. 38, pp. 171–177, 2019. 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing.

[40] H. Sun, L. Li, and C. Liu, "Novel opposite stirring mode in bloom continuous casting mould by combining swirling flow nozzle with ems," *Metals*, vol. 8, no. 10, 2018.

[41] K.-C. Ke and M.-S. Huang, "Quality prediction for injection molding by using a multilayer perceptron neural network," *Polymers*, vol. 12, no. 8, p. 1812, 2020.

[42] G. A. Susto, S. Pampuri, A. Schirru, A. Beghi, and G. De Nicolao, "Multi-step virtual metrology for semiconductor manufacturing: A multilevel and regularization methods-based approach," *Computers & Operations Research*, vol. 53, pp. 328–337, 2015.

[43] C. A. Escobar and R. Morales-Menendez, "Machine learning techniques for quality control in high conformance manufacturing environment," *Advances in Mechanical Engineering*, vol. 10, no. 2, p. 1687814018755519, 2018.

[44] H. Tercan and T. Meisen, "Machine learning and deep learning based predictive quality in manufacturing: a systematic review," *Journal of Intelligent Manufacturing*, vol. 33, no. 7, pp. 1879–1905, 2022.

[45] M. R. Mehregan, A. Rezasoltani, and A. M. Khani, "A novel hybrid machine learning model for defect prediction in industrial manufacturing processes," 2025.

[46] G. Marín Díaz, "Comparative analysis of explainable ai methods for manufacturing defect prediction: A mathematical perspective," *Mathematics*, vol. 13, no. 15, p. 2436, 2025.

[47] P. Goetzinger, K. Huffstadt, and S. Noy, "Could industry 4.0 benefit from hybrid ai analytics to enhance smart manufacturing, defect detection, and prediction?," *The International Journal of Advanced Manufacturing Technology*, pp. 1–14, 2025.