Automating Fault Detection and Quality Control in PCBs: A Machine Learning Approach to Handle Imbalanced Data

Mehrnaz Mirzaei

A Thesis

in

The Department

of

Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

September 2023

© Mehrnaz Mirzaei, 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mehrnaz Mirzaei**

Entitled: **Automating Fault Detection and Quality Control in PCBs: A Machine Learning Approach to Handle Imbalanced Data**

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

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Mohsen Ghafouri

_____ Examiner
Dr. Arash Mohammadi

_____ Examiner
Dr. Mohsen Ghafouri

_____ Supervisor
Dr. Farnoosh Naderkhani

Approved by _____
Dr. Chun Wang, Chair
Department of Institute for Information Systems Engineering

_____ 2023

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Automating Fault Detection and Quality Control in PCBs: A Machine Learning Approach to Handle Imbalanced Data

Mehrnaz Mirzaei

Printed Circuit Boards (PCBs) are fundamental to the operation of a wide array of electronic devices, from consumer electronics to sophisticated industrial machinery. Given this pivotal role, quality control and fault detection are especially significant, as they are essential for ensuring the devices' long-term reliability and efficiency. To address this, the thesis explores advancements in fault detection and quality control methods for PCBs, with a focus on Machine Learning (ML) and Deep Learning (DL) techniques. The study begins with an in-depth review of traditional approaches like visual and X-ray inspections, then delves into modern, data-driven methods, such as automated anomaly detection in PCB manufacturing using tabular datasets. The core of the thesis is divided into three specific tasks: firstly, applying ML and DL models for anomaly detection in PCBs, particularly focusing on solder-pasting issues and the challenges posed by imbalanced datasets; secondly, predicting human inspection labels through specially designed tabular models like TabNet; and thirdly, implementing multi-classification methods to automate repair labeling on PCBs. The study is structured to offer a comprehensive view, beginning with background information, followed by the methodology and results of each task, and concluding with a summary and directions for future research. Through this systematic approach, the research not only provides new insights into the capabilities and limitations of existing fault detection techniques but also sets the stage for more intelligent and efficient systems in PCB manufacturing and quality control.

Acknowledgments

I am deeply grateful to my supervisor, Prof. Farnoosh Nadarkhani, for her unwavering support, vast knowledge, and insightful guidance throughout this thesis journey. Her mentorship and encouragement have been pivotal in successfully completing this work. Additionally, I would like to express my heartfelt appreciation to Prof. Arash Mohammadi and Prof. Mohsen Ghafouri for graciously serving as committee members and providing valuable insights.

I am also thankful for my mom and dad. Their love and support mean everything to me and have pushed me to go far in my studies. I am also grateful for my little sister. Her love and the joy she brings into my life keep me going and inspire me every day. Furthermore, I am thankful for my friends and all those who have offered their support and companionship during this phase of my life. Their belief in my abilities and encouragement has been instrumental in helping me overcome challenges and achieve this significant milestone.

Lastly, I want to express my gratitude to Canada and Concordia University for warmly welcoming me as an international student. They have provided me with the opportunity to embark on a new journey, experience a new life, and follow my dreams. Their support and openness have made this endeavor possible, and I am forever grateful for this transformative experience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's rapidly advancing world, the indispensability of electronic devices has surged (Kaliyavaradhan, Prem, Ambily, & Mo, 2022; C. Wu, Awasthi, Qin, Liu, & Yang, 2022). This burgeoning significance has driven the expansion of the electronics industry, with Printed Circuit Boards (PCBs) standing as an integral component across a spectrum of devices, from computers and medical equipment to smartphones, automotive systems, aerospace technology, industrial machinery, and household appliances like televisions and microwaves. The detailed manufacturing and assembly of PCBs demand substantial time and resources (Cui & Anderson, 2016; Nguyen & Bui, 2022). Consequently, accurate identification of recurring PCB defects is imperative to eradicate the costly and wasteful practice of discarding faulty manufactured boards. Hence, developing innovative and resilient defect detection methods is vital, ensuring the optimal performance of electronic devices, cost savings, and a more sustainable technological future (Spinzi, 2017).

Recognizing the pivotal role that a PCB's quality plays in determining its performance, safety, and reliability, the significance of effective defect detection becomes even more pronounced. Defects in a PCB can result in performance issues or even lead to device failure. A high-quality PCB is one that meets all the necessary specifications and standards, covering aspects such as design, material selection, fabrication, and assembly. Striving for superior quality and defect-free manufacturing not only boosts customer satisfaction but also serves as a preventive measure against potential losses in terms of time and money. As a result, the task of detecting defects in PCB production lines

has attracted significant attention from both businesses and researchers. The goal is to minimize instances of non-conformities and the wastage of valuable resources during the manufacturing process (Verna, Genta, Galetto, & Franceschini, 2023). The sophistication of this task can vary significantly depending on the specific stages and procedures involved in the PCB production line.

Generally speaking, the production line for circuit boards can vary depending on several factors, such as the complexity of the board, the technology being used, and the application's specific requirements. Production lines can exhibit variations such as multi-layer lamination or specialized coating procedures, depending on complexity, while others might streamline the process for simpler boards. Nonetheless, the standard PCB production process comprises five core stages. Illustrated in Fig. 1.1, the production line initiates with the printing machine and Solder Paste Inspection (SPI), followed by Surface Mount Device (SMD) placement. Subsequently, components traverse a reflow oven and undergo Automatic Optical Inspection (AOI). Each of these stages generates diverse data types, enabling customized analyses aligned with a company's specific requirements and ultimate objectives. The complexity and variability in these production stages underscore the necessity for robust quality control methods.

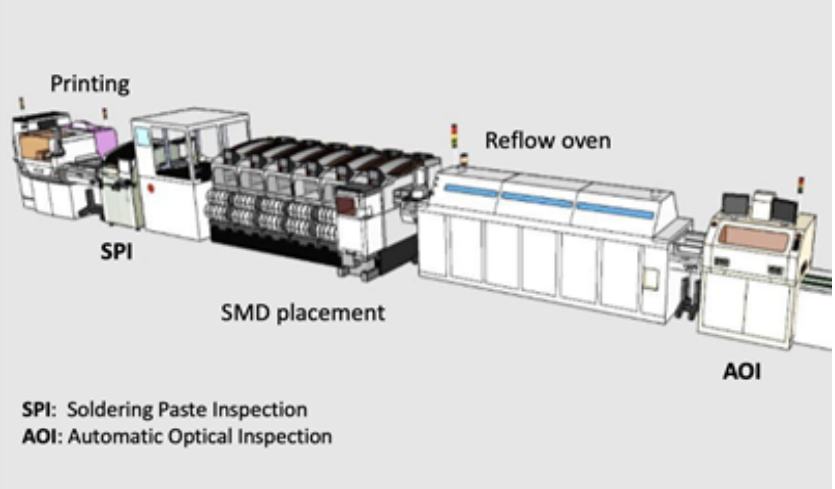


Figure 1.1: Printed circuit board production line (PHM_Society, 2022)

As mentioned, for the smooth and efficient functioning of PCB production and assembly, it is crucial to implement strong quality control methods. In this regard, various techniques are in use to spot defects, such as visual inspection, AOI, and X-ray screenings. In visual inspection,

a trained examiner looks for visible problems like missing parts or cracks and categorizes them to find the root causes (Chaudhary, Dave, & Upla, 2017; Galetto, Verna, Genta, & Franceschini, 2020). AOI uses specialized cameras and software to find issues like misplaced components or poor soldering. X-ray techniques help identify defects not visible to the eye, like hidden soldering errors or microscopic cracks. Besides these traditional methods, data analysis and ML can identify defects in real-time on PCB production lines. As components and possible defects grow in number and size, the shortcomings of older inspection techniques become increasingly evident (Dai, Mujeeb, Erdt, & Sourin, 2020). Therefore, given the advancements in production technology and the availability of large datasets, Artificial Intelligence (AI) and ML-based algorithms are progressively being used as practical solutions for automated defect detection. These algorithms can analyze data from multiple sources, like sensors, images, and electrical tests, to spot irregularities that may signal a defect. By integrating these algorithms into the production process, manufacturers can quickly identify and diagnose undesired defects, reducing the risk of faulty PCBs entering the market. In addition, they offer several distinct advantages when compared to conventional methods. These benefits include reducing production costs, as well as the ability to detect issues rapidly with high levels of accuracy (Putera & Ibrahim, 2010; Yang et al., 2020). Thus, as technology evolves, the demand for more advanced, data-centric approaches for defect detection is rising.

In response to this rising demand for sophisticated solutions, knowledge-based or data-driven methods stemming from ML and AI realms demonstrate the capability of not only adapting to adverse conditions but also exhibiting a stable level of high accuracy and efficiency over an extended period of time. These methods aim to automatically detect and diagnose faults in machines by analyzing collected data during the production and inspection phases. The key to data-driven strategies is the ability of AI algorithms to analyze large amounts of data and recognize the flawless status of the machines without the need for prior knowledge of the signals or features associated with faults (Mazzoleni et al., 2022).

The primary aim of this research study is to meticulously evaluate the dataset made available by the Prognostics and Health Management (PHM) challenge, which originates from Bitron Spa company, an industry leader in the development and manufacturing of mechatronic devices (PHM_Society, 2022). This investigation seeks explicitly to devise ML and DL solutions that are

tailored for detecting faults at various stages in the production line of PCBs. By exploring a range of ML and DL methodologies, the study aims to offer an in-depth analysis of how each approach can contribute to enhanced fault detection during the manufacturing process. A summary outlining the pivotal contributions of this thesis research can be found in the subsequent section.

1.1 Contributions

As we navigate through the complexities of PCB manufacturing, the importance of leveraging advanced ML and DL techniques cannot be overstated. These technologies offer a pathway to greater efficiency, cost-effectiveness, and sustainability in an industry that is both competitive and resource-intensive. Against this backdrop, we have conducted our research, aiming to push the boundaries of what is possible in quality control and automation.

(1) **Addressing Imbalanced Data Challenges in PCB Defect Detection:** In real-world scenarios, defective pieces are far fewer than non-defective ones, making defect detection a challenging task due to imbalanced data. This study focuses on this issue, specifically within the context of ML and DL models for PCB defect detection. Our goal is to balance the dataset using techniques ranging from data-level to algorithm-level methods. Moving beyond traditional oversampling techniques, we offer a more comprehensive set of solutions. We categorize these solutions and uncover the potential for more effective ML algorithms, the ones that can be fine-tuned to recognize minority classes without bias towards the majority. We delve into the specifics of well-known resampling methods like SMOTE and discuss alternative approaches such as ensemble methods. These insights contribute to a holistic understanding of data imbalance and offer practical tools for model improvement.

(2) **Unlocking TabNet: A Step Forward in Tabular Data Analysis:** In this study, we present a thorough exploration and evaluation of TabNet, a deep-learning model specifically designed for tabular data. This innovative approach surpasses traditional ML algorithms in terms of accuracy and interpretability while maintaining computational efficiency. Through rigorous experimentation, we demonstrate how TabNet can tackle complex nonlinear relationships in data, opening up new avenues for solving real-world problems. Our analysis thus serves as

a comprehensive guide to implementing TabNet, effectively bridging the gap between traditional tabular methods and advanced DL techniques.

(3) **Hybrid Methods for Feature Selection:** We introduce an innovative approach to feature selection through hybrid methods that synergize various traditional techniques. Our method demonstrates notable performance enhancement across multiple metrics compared to standard practices, serving as a methodological advancement in the field of machine learning. The versatility of the approach is emphasized through its successful application to diverse types of datasets. Additionally, a comparative analysis with existing methods substantiates the advantages of our hybrid approach.

1.2 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter 2, goes through a literature review on the thesis topic, along with a brief overview of the background needed for the following chapters.
- Chapter 3, introduces the datasets and tools used in the study. It covers essential dataset preparation steps, including addressing data imbalance experimental results for fault detection in PCBs at the solder paste stage.
- Chapter 4, explores using ML and DL methods for defect detection during the AOI stage.
- Chapter 5, focuses on automating the determination of repair status for PCBs using ML and DL models, aiming to replace manual evaluation and enhance both efficiency and sustainability in PCB manufacturing.
- In Chapter 6, the thesis is wrapped up, and potential future research directions in this area of study are explored in conclusion.

Chapter 2

Background and Literature Review

2.1 Fundamentals of PCBs and Fault Detection

This section provides a general overview of PCBs, their components, their manufacturing processes, as well as common types of faults or defects that can occur in PCBs. Additionally, the challenges and complexities involved in detecting faults in PCBs are explained.

PCBs, or Printed Circuit Boards, serve as the backbone of electronic devices, providing a platform for connecting and supporting electronic components. PCBs consist of layers of insulating material with conductive tracks scratched onto them, forming a circuitry network. The components, such as resistors, capacitors, diodes, transistors, relays, connectors, and integrated circuits, are mounted on the PCB and interconnected through these conductive tracks. During the manufacturing process of PCBs, various techniques are employed, including design, fabrication, assembly, and testing. Design involves creating the layout of the PCB and determining the placement and routing of the components and tracks. Fabrication involves manufacturing the PCB by engraving the conductive tracks onto the insulating layers and creating the necessary vias and pads. Assembly refers to mounting the components onto the PCB and soldering them in place. Finally, testing is conducted to ensure the functionality and reliability of the PCB. However, PCBs are susceptible to faults or defects that can occur during the manufacturing process or through environmental factors over time. The focus of these defects is typically on surface defects or components and joint defects (Ling & Isa, 2023). There are around 34 common types of faults or defects in PCBs, including open

circuits (where a connection is broken), short circuits (where unintended connections are formed), soldering defects (such as insufficient or excessive solder), component misplacements, defects associated with temperature inconsistency, PCB contamination (Sankar, Lakshmi, & Sankar, 2022). Detecting faults in PCBs is a complex task due to several challenges. The increasing complexity and density of PCB designs, with complex layouts and compact components, make it challenging to identify faults visually. Additionally, the limited accessibility of certain areas of the PCB prevents manual testing and inspection. The need for non-destructive testing methods is also crucial to avoid damaging the PCB during the detection process. Moreover, the presence of noise and variability in PCB signals further complicates fault detection, requiring sophisticated techniques and algorithms to differentiate between normal and faulty behavior.

2.2 Traditional Fault Detection Techniques

This section discusses conventional methods used for fault detection in PCBs, such as visual inspection, manual testing, and functional testing. Also, it summarizes the advantages and limitations of these traditional techniques and highlights the need for more advanced and automated approaches. Traditional fault detection techniques in PCBs can be classified into seven main categories as follows:

2.2.1 Visual Inspection

This technique is the simplest method and involves visually examining the PCB for any physical defects, such as misaligned components, solder bridges, or damaged traces. Operators typically rely on either a magnifying glass or a calibrated microscope to assess whether the PCB is faulty or meets the required standards. This assessment helps them determine if any corrective actions are needed (Anoop, Sarath, & Kumar, 2015). As a result, this method can be used at a low cost, is simple to implement, can quickly identify certain visual defects, and it does not require specialized equipment. However, it is subjective and dependent on the inspector's expertise, it is limited in detecting internal or hidden defects, and it is inefficient for large-scale production or complex PCBs. Furthermore, as PCB components and solder joints become smaller, the chances of flaws increase,

making it possible for even experienced inspectors to make mistakes.

2.2.2 Traditional Image Processing-Based Inspection

Traditional image processing-based inspection is almost the same as Automated Optical Inspection (AOI), which utilizes cameras and image processing algorithms to detect defects in PCBs. Specifically, the system utilizes pre-established rules and thresholds to compare the characteristics and make decisions. It offers faster and more accurate inspection than visual methods, detecting issues like missing components, wrong placements, soldering defects, and polarity errors. That is why many researchers have introduced machine vision technique using traditional rule-based image processing method to detect PCB defects. Some of these studies concentrate on **Surface Defect Detection**. For example, (Chauhan & Bhardwaj, 2011) implemented a subtraction algorithm that compares a reference image to identify defective regions, including over-etchings, under-etchings, and holes. Likewise, (Ma, 2017) conducted a comparison between a target image and an aggregated set of images serving as the reference image. Furthermore, to maintain the reference image's validity, a dynamic updation strategy was incorporated.

On the other hand, the focus of other researchers is on **Solder Joints or Components Defect Detection**. For instance, (Hassanin, Abd El-Samie, & El Banby, 2019) presents an automatic inspection approach for PCBs that accurately determines fault location and identifies fault types using digital image processing techniques, including speeded-up robust feature extraction (SURF) for registration, feature point detection, and component matching. It enables the construction of a comprehensive dictionary for diverse PCB components and can localize and identify missing components regardless of PCB position or rotation.

Although traditional image processing-based approaches serve some advantages over visual inspection, it has some disadvantages, such as expensive equipment and setup costs, limitations in detecting non-camera-visible defects, and the need for additional inspections in complex or multilayer PCBs. Adding to the previous general cons, traditional machine vision-based methods for PCB defect detection face challenges related to the acquisition of suitable reference images, sensitivity to environmental changes, limited real-time capabilities, and the need for precise calibration. These methods typically rely on image subtraction or matching, necessitating a perfect reference

image that is difficult to obtain, time-consuming, and costly to prepare for different PCBs. The detection performance is highly reliant on the quality of the reference image, making them sensitive to variations in illumination, noise, and small shadows, leading to false alarms. Moreover, they lack generalizability to different samples and struggle to accurately detect components with similar colors to the background. Achieving real-time detection, especially with high-resolution inspection images in the real PCB industry, requires additional image processing procedures beyond simple subtraction, resulting in decreased inference speed. Furthermore, achieving pixel-level alignment between the reference and inspection images demands precise calibration equipment and trained personnel. Although some novel approaches have been introduced to address image shifts, they complicate the detection algorithm, increase computation costs, and involve more parameter choices (Ling & Isa, 2023).

2.2.3 X-ray Inspection

X-ray inspection in PCBs involves using X-ray technology to take detailed pictures of the inside of the board. The PCB is placed on a platform, and an X-ray machine emits X-rays that pass through the board. A detector captures these X-rays, creating images that can be analyzed for defects. In the next step, trained inspectors or automated systems examine these X-ray images to identify any faults and ensure the PCB's quality (Neubauer & Hanke, 1993). This technique is valuable for inspecting PCBs, especially for hidden defects in complex or multilayer boards. It is non-destructive and can detect issues like soldering shorts, voids, and insufficient solder. On the other hand, it requires expensive equipment, specialized expertise, and has a slower inspection process. Additionally, it has limited capability in providing detailed component functionality information.

2.2.4 Functional Testing

This technique involves testing the functionality of the PCB by applying signals and checking the responses. It can help identify faults such as open circuits, short circuits, or incorrect electrical behavior. By implementing functional testing, the overall functionality of the PCB is validated, and some faults that may not be apparent through visual inspection can be identified. However, it

requires specific test procedures and setups for different PCB designs, which is time-consuming, especially for complex boards. Furthermore, this method is limited in detecting subtle or intermittent faults (Gebus, Lorillard, & Juuso, 2004).

2.2.5 In-Circuit Testing (ICT)

ICT is a method used to check the electrical performance of PCBs. It involves placing the PCB on a fixture, running a test program to send electrical signals, and measuring the responses to ensure they match the expected values. Any deviations indicate potential issues with components or connections. This technique helps to ensure the PCB's quality and reliability before it is used in products. It detects defects like incorrect component values, faulty components, or improper soldering (Albee, 2013). Although this method can accurately identify faults in individual components and connections, it includes high setup and fixture costs and limited coverage for complex or densely defected PCBs. Thus, it could be more efficient for low-volume production or prototypes.

2.2.6 Flying Probe Testing

In this method, various places on the PCB are accessed, and electrical tests are run using robotic test probes. Although it can be slower than ICT, it is excellent for prototypes and low-volume production. Flying probe testing does not require custom fixtures; as a result, the setup costs are low. Also, it provides electrical testing of PCBs without physical contact. Conversely, this technique is slower compared to other testing methods and has limited access to specific points on the PCB (Jurj, Rotar, Opritoiu, & Vladutiu, 2020).

2.2.7 Environmental Stress Screening (ESS)

This technique is based on monitoring the PCBs for a specific duration before being assembled and subjects components to various conditions, such as temperature, humidity, vibration, thermal cycling, (Pohl & Dietrich, 1995). Although ESS helps identify hidden defects that could lead to failures in the field and simulates real-world operating conditions, it requires specialized equipment and controlled environments, adds additional time and cost to the manufacturing process, and cannot detect all types of defects, particularly design-related faults.

After examining the various methods discussed earlier, it is evident that existing traditional approaches have limitations. They either require operator involvement, resulting in slower processes and increased potential for errors or rely on rule-based algorithms with predefined rules that may not be suitable for mass production. Therefore, there is a critical need for a defect detection method that combines computer vision-based techniques with machine learning. This integration reduces the reliance on manual intervention, leading to faster inspections and improved efficiency and accuracy in automated defect detection. The following section will explore advanced fault detection methods and their advantages.

2.3 Advanced Fault Detection Methods

This section discusses emerging technologies and methodologies for fault detection in PCBs, such as machine learning methods, infrared thermography, and acoustic imaging. Furthermore, it explains how these advanced techniques improve fault detection accuracy and efficiency and emphasizes any limitations or challenges associated with these methods.

2.3.1 Non-optical Inspection Techniques

- **Infrared Thermography**

  Infrared thermal imaging is a technique that utilizes specialized cameras capable of detecting and capturing thermal radiation emitted by objects. It operates in the infrared spectrum, which is beyond the range of human vision. By detecting and visualizing temperature variations, infrared thermal imaging enables the identification and analysis of heat patterns on the surface of objects. This method for PCB fault detection primarily consists of three key steps: heat source identification, feature extraction, and thermal pattern recognition (Jiuqing & Xingshan, 2002). The first step involves locating areas on the PCB with abnormal heat patterns by comparing thermal distribution across the surface. Raised temperatures indicate potential faults or anomalies. In the next step, feature extraction, relevant thermal features

are captured from the identified heat sources, such as temperature gradients, intensity variations, or specific thermal signatures associated with different fault types. This quantifies and describes the detected anomalies' thermal characteristics. The final step is to use pattern recognition algorithms or machine learning techniques to classify and categorize the extracted features into specific fault types. Thermal patterns are matched with known fault signatures or differentiated based on thermal characteristics. Leveraging the unique capabilities of infrared thermal imaging, this non-destructive and non-contact PCB fault detection method, provides valuable insights (Sarawade & Charniya, 2018). It helps identify issues like excessive heat generation, poor electrical connections, faulty components, or thermal management problems. This technique has been implemented in various research. For instance, one study, (Dong & Chen, 2019), compares various registration algorithms, including mutual information-based and scale-invariant feature transform (SIFT) feature-based methods, to determine the optimal approach for PCB infrared thermal defect detection. The research findings indicate that the mutual information-based registration method achieves higher accuracy but requires more time to complete. Conversely, the SIFT feature-based registration method is less accurate but demonstrates greater efficiency. Another article explores how the temperature of a PCB relates to the components placed on it. It investigates different methods to accurately identify the edges of components in infrared images, providing a foundation for recognizing important features in these images (Z. Wang et al., 2022).

- **Acoustic Emission Analysis**

  Acoustic Emission Analysis in PCB defect detection is a technique that involves monitoring and analyzing sound waves generated during the operation or testing of a PCB. Acoustic emissions refer to the release of transient stress waves caused by the rapid release of energy within the material. In the context of PCB defect detection, acoustic emission analysis is used to detect and identify potential faults or anomalies. When defects occur, such as delamination, cracks, or component failures, they can produce distinctive acoustic emissions. These emissions are captured using specialized sensors and then analyzed to identify the specific fault type, location, and severity. Acoustic emission analysis plays a vital role in evaluating the

health and integrity of PCBs, providing valuable insights. This method complements other inspection techniques, offering an additional layer of information for comprehensive defect detection or monitoring both mechanical characteristics and acoustic emission parameters simultaneously in order to find their correlation (Kovtun, Boiko, & Petrashchuk, 2017). This non-destructive and real-time approach identifies and assesses faults, enhancing the reliability and performance of PCBs. Specifically, these non-destructive tests can be conducted on various PCB components, including solder joints, to evaluate their strength (Kovtun, Boiko, & Petrashchuk, 2018, 2019). Furthermore, these tests can also detect potential defects in specific components like capacitors (Krieger et al., 2006). Therefore, the utilization of acoustic emission techniques, in conjunction with mechanical testing, and the identification of correlations between acoustic emission characteristics and mechanical properties of solder joints, such as ultimate strength, provides a justification for employing it in the development of diagnostic methods for assessing the strength of PCB solder joints (Salahouelhadj et al., 2014).

- **Time Domain Reflectometry (TDR)**

  Time Domain Reflectometry (TDR) is a non-destructive method used to find defects in PCBs. It works by sending an electrical pulse through the PCB and analyzing the reflections caused by any changes or defects in the board. TDR can identify issues like open circuits, short circuits, or changes in impedance by studying the time and strength of these reflections. It is commonly used for finding faults, ensuring quality, and solving problems during PCB manufacturing and testing processes (Choi, Kim, Kim, & Kim, 2018). By combining TDR techniques with a ”return-path,” defects like right angle bends can be better detected, with a resolution of around 5-8 mm (Chun, Ang, Chai, & Tay, 2003). However, as devices get smaller, detecting flaws becomes harder due to limitations on reducing the pulse duration, which limits the achievable resolution.

2.3.2 Data-Driven Fault Detection and Maintenance

- **Machine Learning and Deep Learning Models**

  In recent years, various types of ML methods have been employed for defect detection in

electronic devices and PCBs. For example, the authors in (Vafeiadis et al., 2018) proposed a framework for evaluating various ML classifiers, including Support Vector Machine (SVM), to identify PCB defects. The SVM classifier proved to have the highest accuracy based on the results. Another approach for defect detection presented by (Yuk, Park, Park, & Baek, 2018) has utilized Speeded-Up Robust Features (SURF) and Random Forest. The methodology entailed extracting features from PCB images via SURF and the subsequent implementation of Random Forest to learn the fault pattern and generate probability estimations. In (Chen, Zhang, & Wu, 2021), A data-driven method using SVMs has been proposed to detect wire bonding defects in IC chips. This method was found to be more sensitive, accurate, and faster than other methods, such as Vision Detection System (VDS) and Convolutional Neural Networks (CNN). One of the most powerful algorithms for object detection is You Look Only Once (YOLO) (Enshaei, Ahmad, & Naderkhani, 2020), which is used for the localization part of the soldering joint. For example, the authors in (Dai et al., 2020) proposed a semi-supervised defect detection method based on the SVM classifier and K-means clustering. Some other data-driven models based on YOLO networks, such as YOLOV3–Mobilenet (Huang, Gu, Sun, Hou, & Uddin, 2019) and YOLO-v5 (Adibhatla et al., 2021; Parlak & Emel, 2023), have been presented for identifying electronic components and defect detection in circuit boards. Utilizing deep learning-based methods in fault diagnosis technology results in improved accuracy, speed, and robustness. Various forms of Artificial Neural Network (ANN), including Feedforward Neural Network (FNN), Recurrent Neural Network (RNN), Modular Neural Network (MNN), and CNN, are some of the deep learning methods used for extracting features and detecting defects from large-sized digital circuits (Gaber, Hussein, & Moness, 2021). CNNs have garnered significant attention from researchers for their application in the realm of defect detection. The implementation of CNNs has demonstrated remarkable outcomes in the field. For instance, in (Lin, Wang, & Lin, 2019), the authors proposed a CNN model for recognizing the IC component defects. A new method for inspecting PCB defects has been introduced in a research paper labeled (Jin et al., 2021). This method employs a CNN model, specifically utilizing EfficientNet-B1 to extract features and BiFPN to perform feature fusion. In order to suggest a suitable solution for real-time inspection of

components in the Surface Mount Technology (SMT), a simplified CNN-based defect detection model called PCBNet is proposed, which has the capability to accurately identify and classify the type and defects of electronic components with minimal processing complexity (H. Wu, Lei, & Peng, 2022). Another advanced deep learning approach based on a skip-connected convolutional autoencoder is used for PBC defect detection with a relatively low false pass rate below 2% (Kim, Ko, Choi, & Kim, 2021). To tackle the repairing problem in PBC production lines, the authors in (Khalilian, Hallaj, Balouchestani, Karshenas, & Mohammadi, 2020) presented a method based on denoising convolutional autoencoders for not only detecting and localizing defects but also for repairing them.

- **Data Analytics and Predictive Maintenance**

  Advanced fault detection methods are highly effective in identifying flaws in PCBs. However, waiting for defects to occur before inspecting them can result in significant costs and losses. Predictive maintenance fault detection offers a beneficial solution by enabling proactive monitoring and early detection of PCB defects, preventing potential issues and costly downtime. This method is particularly practical for critical systems in aerospace, defense, medical devices, and industrial automation, where the reliability and uptime of electronics are crucial. Additionally, it is valuable for complex PCB designs with high component density, requiring advanced technology for effective fault diagnosis and repair. Furthermore, products with long lifecycles, such as industrial machinery or infrastructure systems, can benefit from continuous monitoring and proactive defect detection to avoid sudden failures. Thus, this method serves as a comprehensive inspection approach for smart manufacturing in the PCB industry (Mourtzis, Angelopoulos, & Panopoulos, 2020).

  One specific application of this approach involves the integration of cyber-physical-social systems (CPSS) with artificial systems, computational experiments, and parallel execution to enhance quality assurance and improvement. The method encompasses descriptive, predictive, and prescriptive intelligence, providing workers with a learning platform, defect monitoring, and guidance for minimizing defects. Leveraging a Transformer-based model for

knowledge reasoning and human-computer interaction, this method demonstrates great potential in addressing quality inspection challenges during the smart manufacturing era of the PCB industry (Y. Wang, Wang, Cao, Li, & Kwan, 2022). Furthermore, this method can be used in digital production control for electrochemical deposition processes, particularly in the etching section of PCB production (Vasilyev, Medvedev, Barakovsky, & Korobkov, 2021)

Chapter 3

Application of Machine Learning for Anomaly Detection of the Solder Paste of PCBs With An Imbalance Dateset

In the introduction (Chapter 1), we discussed the role of inspection stations in ensuring the quality of PCBs during manufacturing. To elaborate, each change made to a PCB is followed by an inspection step to prevent defective boards from moving to the next step in the process. One critical inspection point occurs right after the solder paste stage. Using low-quality solder paste can lead to defects in a PCB. Therefore, any boards with suspect soldering are sent for further review at the AOI stage, which comes after component placement.

This chapter aims to develop a predictive model for determining which PCBs should undergo additional examination at the AOI stage. Utilizing ML and DL algorithms, we intend to create an intermediary model to flag which components should be sent to AOI. This model will be based on the physical characteristics of the solder paste, as recorded in the SPI dataset. In summary, our objective is to predict whether a component will be flagged as defective by the inspection system using SPI dataset.

The structure of this chapter is as follows: Section 3.1 begins by outlining the problem the research seeks to address. This is followed by Section 3.2, which provides an overview of the datasets

employed in this chapter. In Section 3.3, we discuss the various steps taken in data preparation, such as data aggregation, cleaning, encoding, feature selection, and handling imbalanced data. Section 3.4 subsequently introduces the ML and DL models that have been utilized for the task of classification and defect detection, along with a detailed discussion of their implementation. Evaluation of the model's effectiveness is presented in Section 3.5. The chapter concludes with Section 3.6, offering a brief summary and concluding observations.

3.1 Problem Statement

As previously discussed, the PCB production process begins with printing the Panel Identification Number (PanelID) on each board. Following this, a carefully measured amount of solder paste is applied to designated areas, known as solder pads, as depicted in Fig. 3.1. This step is highly sensitive and requires meticulous accuracy. Any errors, such as applying too much or too little solder paste or misalignment of the paste, can compromise the functionality of the entire PCB as a final product. During this solder pasting phase, sensors collect data on various physical characteristics of the applied paste. This information is then organized into tabular data, which displays both the quantity and precise placement of the solder paste on the pads. If the characteristics of the solder paste fall outside the pre-established parameters for each specific solder pad, the associated component may not attach securely or operate correctly. Consequently, early identification of these problematic PCBs is essential to separate them and send them for additional scrutiny at the AOI.

Having established the critical role of solder pasting and its impact on PCB quality, the next logical question is how to separate the boards that may require further attention effectively. That brings us to the focus of this chapter. In line with this objective, the chapter explores whether ML and DL models can effectively determine which circuit boards should advance from the component placement to the AOI stage, based solely on the SPI data, without considering the AOI defect labels. In other words, this chapter aims to predict the probability of the presence of the PCB components in the AOI stage by analyzing the physical characteristics provided in the SPI.

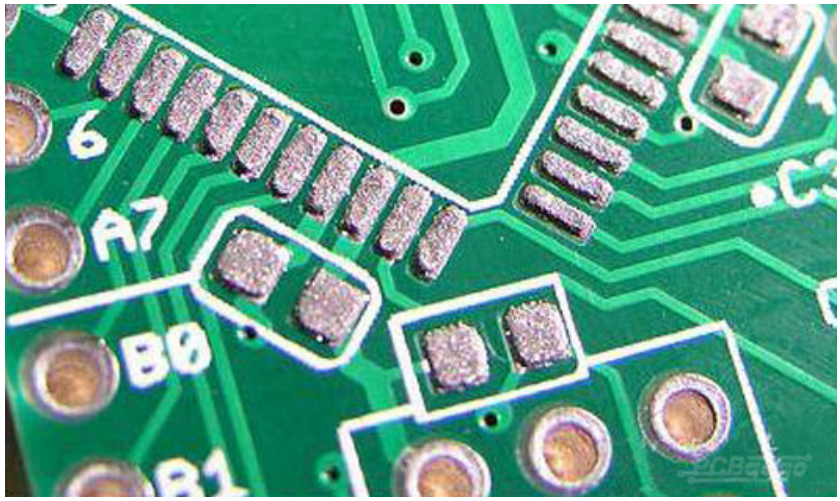


Figure 3.1: Solder Paste Application on Solder Pads (PCBGOGO, 2019)

3.2 Dataset Overview

As previously mentioned, the PCB production line under our analysis comprises five stages. However, our data collection for defect detection focuses specifically on the SPI and AOI stages. The SPI dataset includes information about the solder paste applied by the machine, with the aim of identifying potential issues such as excess or missing paste and short circuits. This dataset encompasses several key identifiers and quantifies specific attributes of the applied solder paste. Meanwhile, the AOI dataset includes identifiers similar to the SPI dataset and provides three types of associated defect labels. Subsequent sections will explain both datasets in greater detail, introducing their respective features.

3.2.1 SPI Dataset

The SPI dataset, with 21 features, serves as a comprehensive source of information for assessing the quality of solder paste application by the printing machine. It aims to detect a variety of potential issues. To this end, the dataset includes identifiers such as PanelID, FigureID, ComponentID, and PadID. The combination of PanelID and FigureID forms the BoardID, representing the PCB. The dataset also includes specific characteristics of the solder paste like Volume, Area, Height, and Shape. A thorough breakdown of these features will be presented in Table 3.1.

Table 3.1: Detailed Feature Overview of the SPI Dataset

	Feature	Unit	Description	Type
1	PanelID	-	Denotes the specific panel.	Categorical
2	FigureID	-	Denotes the specific Figure.	Categorical
3	ComponentID	-	Denotes the specific Component.	Categorical
4	PinNumber	-	Component's associated pin number.	Numerical
5	PadID	-	Unique ID for pin's supporting pad.	Categorical
6	Date	MM/DD/YYYY	Shows the SPI operation date.	Numerical
7	Time	HH:MM:SS	Time of SPI operation in seconds.	Numerical
8	PosX	mm	X coordinate of pin from bottom left.	Numerical
9	PosY	mm	Y coordinate of pin from bottom left.	Numerical
10	PadType	-	Specifies the type of pad.	Categorical
11	Volume	%	Percentage of the paste volume.	Numerical
12	Height	um	Height of the paste in micrometers.	Numerical
13	Area	%	Percentage of the paste area.	Numerical
14	OffsetX	%	X-axis offset percentage.	Numerical
15	OffsetY	%	Y-axis offset percentage.	Numerical
16	SizeX	mm	Size of the paste in the X direction.	Numerical
17	SizeY	mm	Size of the paste in the Y direction.	Numerical
18	Volume	um3	Paste volume in cubic micrometers.	Numerical
19	Area	um2	Paste area in square micrometers.	Numerical
20	Shape	um	Shape of the paste.	Numerical
21	Result	-	Outcome of SPI inspection.	Categorical

3.2.2 AOI Dataset

The AOI dataset is another valuable source of information that complements the SPI dataset. Similar to the SPI dataset, it includes PanelID, FigureID, ComponentID, and PadID identifiers. These common identifiers facilitate the integration of the two datasets for more comprehensive analysis. In addition to these shared identifiers, the AOI dataset contains unique defect labels. These labels are assigned to the PCBs during three distinct phases of the manufacturing process. These defect labels serve as crucial indicators, offering insights into potential issues and imperfections that may arise at different stages of production. The features will be fully explained in the Table 3.2.

Table 3.2: Detailed Feature Overview of the AOI Dataset

	Feature	**Description**	**Categories**
1	PanelID	Denotes the specific panel.	-
2	FigureID	Denotes the specific Figure.	-
3	ComponentID	Denotes the specific Component.	-
4	PinNumber	Component's associated pin number.	-
5	MachineID	Denotes the machine performing the AOI operation.	1) A 2) B
6	AOILabel	The label applied by the AOI machine based on the type of the defect.	1) Broken 2) Coplanarity 3) Jumper 4) LeanSoldering 5) Misaligned 6) Soldered 7) Translated 8) UnSoldered
7	OperatorLabel	The label applied by the human operator after visual inspection.	1) Good 2) Bad
8	RepairLabel	The label applied by the repairment operator after an inspection with a microscope.	1) Not Available 2) NotYetClassified 3) NotPossibleToRepair 4) FalseScrap

In Table 3.2, it is evident that machines A and B perform the automated optical inspection of PCBs. Each PCB is tagged with one of eight predefined defect labels during this inspection, collectively termed as **AOILabel**. These labels serve to flag potential issues with component pins on the PCBs. It is important to note that these machine-assigned labels are not definitive; due to a considerable margin of error in automated inspection, each PCB undergoes further examination. Following the AOI process, an expert operator re-evaluates the component pins and assigns a "Good" or "Bad" status (**OperatorLabel**), determining whether the PCB should continue in the production line or be sent for repair. Subsequently, a repair specialist examines flagged pins and assigns a **RepairLabel** based on their repairability. Further details on these labels are elaborated below:

(1) **AOILabel:**

- **Broken**: The pin has incurred physical damage or has been completely broken off, rendering it ineffective and unreliable for establishing any form of electrical connections.
- **Coplanarity**: Pins are not aligned on the same plane, leading to issues in making consistent and reliable connections.
- **Jumper**: An unintended connection between two pins, often due to solder bridging, which can result in electrical shorts.
- **Lean Soldering**: Insufficient solder material or poorly executed soldering technique, leading to a weak mechanical or electrical connection.
- **Misaligned**: The pin is not aligned properly with the intended footprint or pad on the PCB, making it difficult or impossible to establish a reliable connection.
- **Soldered**: This label typically means the pin is correctly soldered, but if it is classified as a defect, it could imply poor soldering quality.
- **Translated**: The pin has shifted from its original position, either laterally or vertically, which can affect the integrity of the electrical connection.
- **UnSoldered**: The pin has not been soldered at all, leaving it without an electrical or mechanical connection to the board.

(2) **OperatorLabel:**

- **Good:** According to the operator's evaluation, the AOI machine incorrectly flagged the pin as defective, and the pin is in proper condition and requires no repair.
- **Bad:** Upon review, the operator verifies that the AOI machine's defect detection is accurate, confirming that the component is indeed flawed and should be forwarded to the repair technician for further assessment.

(3) **RepairLabel:**

- **Not Available (NA):** This label is designated for components that were incorrectly flagged as defective by the AOI machine but are subsequently verified as "Good" by the OperatorLabel. Consequently, these components bypass the repair examination stage and are automatically assigned the "NA" label.
- **NotYetClassified:** This label is assigned to components for which repair information is currently unavailable, indicating that they have not yet been categorized in terms of repairability.
- **NotPossibleToRepair:** The label indicates that the damaged pin cannot be fixed, recommending that the entire panel be discarded.
- **FalseScrap:** This label is applied when the operator initially identifies a defect, but later assessments reveal that the component does not require any repair operation.

3.2.3 Descriptive Analysis of The Datasets

1. Statistical Overview of Feature Variables Table **??** offers an overview of the numerical characteristics of the SPI dataset, helping us understand the nature of the data we are working on. The table reveals that some features, such as Area(%), have values closely clustered around their mean. In contrast, other features like Volume(%) exhibit a high degree of variability. Notably, OffsetX(%) and OffsetY(%) appear to have outliers, especially in the negative range. Furthermore, the data for Volume(um3) and Area(um2) are skewed, with maximum values significantly higher than their respective means. Overall, the dataset presents a mix of features, some of which may need further analysis to address outliers.

Table 3.3: Descriptive Statistics of Numerical Features

	Volume(%)	Height(um)	Area(%)	OffsetX(%)	OffsetY(%)	SizeX
mean	92.71	112.51	99.25	0.92	0.66	0.87
std	11.38	9.59	7.70	2.76	5.65	0.82
min	0.00	35.00	0.00	-29.25	-55.08	0.22
median	92.38	112.64	99.42	0.61	0.42	0.50
max	472.64	346.25	219.09	26.04	71.77	5.10
	SizeY	**Volume(um3)**	**Area(um2)**	**Shape(um)**	**PosX(mm)**	**PosY(mm)**
mean	0.77	113102900	1074841	46.00	121.73	72.24
std	0.75	371857400	3680537	15.59	66.03	35.17
min	0.22	0	0	-35.00	9.10	11.80
median	0.50	27741470	238731	44.70	125.60	60.70
max	5.10	8078115000	27900110	360.70	236.10	125.80

2. Feature Correlation: In our study's exploration phase, a feature correlation heatmap was employed to examine the relationships among different features, depicted in Fig. 3.2. Remarkably, the features "sizeX," "sizeY," "Volume," and "Area" displayed exceptionally high correlation coefficients, which indicates a robust linear relationship among them. This phenomenon can be best understood by considering the inherent geometrical properties associated with PCB components. In essence, the size in the X-dimension ("sizeX") and the size in the Y-dimension ("sizeY") are fundamental parameters that dictate the layout and spatial requirements of each component. The "Volume" can be viewed as an encapsulation of these two parameters, often calculated as the product of "sizeX," "sizeY," and the height of the component. Thus, if either "sizeX" or "sizeY" changes, the "Volume" changes proportionally, establishing a nearly perfect correlation.

Similarly, the "Area" of a component, which is usually derived by multiplying "sizeX" by "sizeY," naturally exhibits strong dependencies on these size dimensions. Given that "Volume" is essentially an extension of the "Area" into the third dimension, it stands to reason that "Volume" and "Area" would also be highly correlated.

The strong connection between these features is not just a coincidence in the data; it shows how closely related the physical aspects of PCB components are. Understanding this correlation is pivotal for optimization algorithms and predictive models. Considering these correlations could help create simpler predictive models that save computational power. Therefore, this super-correlation serves as a crucial insight into the underlying structure of the dataset and the physical world it

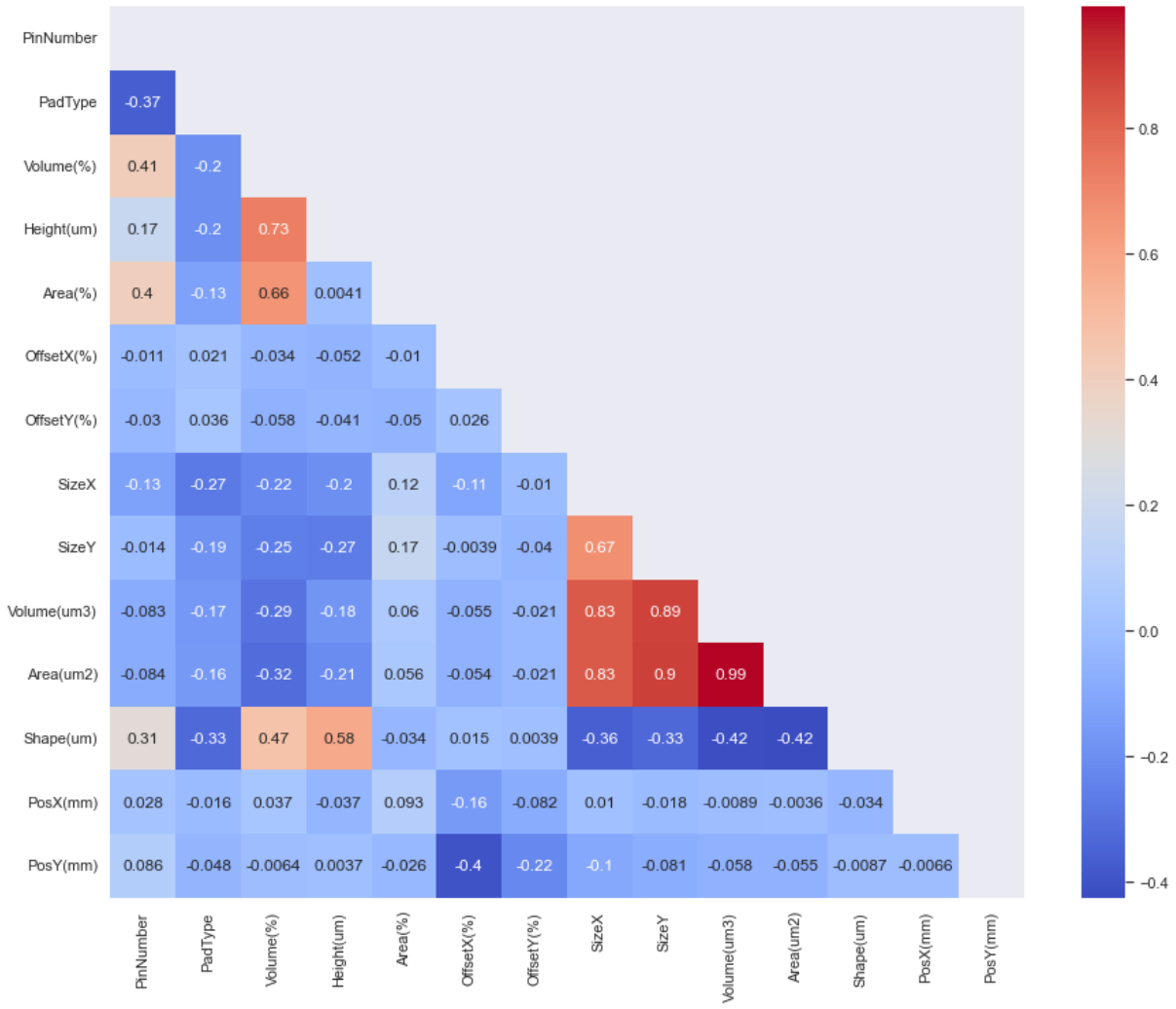


Figure 3.2: Heatmap of Feature Correlations

represents. It also guides feature selection, urging us to exercise caution when employing these features collectively in machine learning models or optimization algorithms.

3.3 Data Preparation

This section outlines the essential procedures required to prepare our datasets for subsequent ML and DL analyses. We begin by aggregating SPI and AOI datasets, into unified training and testing sets. Following this, we apply various pre-processing techniques—including data cleaning, data encoding, and feature selection—to refine the data and address the data imbalance problem. These

preparations lay the groundwork for the ML and DL models to perform accurate and efficient analyses.

3.3.1 Data Aggregation

Since the SPI dataset contains a massive amount of data, it is split into four different smaller CSV files for the train set and 2 for the test set. Therefore, the first step for making the proper dataset is to combine these files and build the main train and test datasets. The other dataset used in this work is AOI, which is much more compact. The Available AOI dataset contains one CSV file for the train set and one for the test set. After combining the datasets, the provided SPI dataset for this study contains 21 features and is divided into 6,000,000 and 3,000,000 data points as train and test sets, respectively. Furthermore, the AOI dataset includes eight features and is separated into around 31,000 rows in the training set and 13,600 rows in the testing set.

As previously stated, the goal of this chapter is to track the data presented in AOI, so we needed to merge the two SPI and AOI datasets by using the common features between the two datasets. In this regard, we concatenated the identifiers (PanelID, FigureID, ComponentID, and PinNumber) to create a unique ID in both datasets and then merged (joined) them on the newly added ID. After combining, a new column named ”Tag” was added to the SPI and filled based on the presence of the PCB component in the AOI. The rows in the AOI dataset get labeled 1, and the missing ones get labeled 0. Notably, this work's goal is to predict the ”Tag” column using various ML and DL models.

3.3.2 Data Pre-processing

Data pre-processing is a crucial subsequent step in preparing the data, as it converts unstructured data into a form readily processed by ML and DL models. The quality of the refined data, which is the outcome of this step, will greatly affect the models' accuracy and overall performance. Neglecting this preparatory stage can result in erroneous or skewed outcomes, undermining the algorithm's overall efficacy. In our project, given the extensive datasets we worked with, several measures were employed to prepare and enhance the data quality, including addressing data imbalance and reducing dimensions. The ensuing sections provide a detailed account of these steps.

1. Data Cleaning: The first task undertaken as pre-processing is data cleaning to ensure the quality of the dataset used for model training. This phase involved the removal of irrelevant and duplicated data, addressing any missing values, and reformatting the data into a structure conducive to analysis. For this study, type adjustments and name corrections were carried out in compliance with UTF-8 encoding. Some missing values were also identified in the ”PinNumber” feature within the AOI dataset, which needed to be managed. Given this study's aim, we preferred eliminating rows with missing values rather than imputing them to avoid introducing bias into our analysis.

2. Data Transformation: In the realm of data pre-processing, converting data into a format readily interpretable by ML models is a crucial step. Categorical features, in particular, pose a challenge for models as they are not represented in a numerical format. One-hot encoding is a widely employed technique for converting categorical variables into numerical data in the context of ML algorithms. The method involves the creation of a new binary column for each category within the categorical variable, thereby avoiding the risk of misinterpretation of categorical data as numerical data with an inherent order or hierarchy. In our study, only one categorical feature, named ”ComponentID,” was transformed into a numerical representation by utilizing One-hot encoding. Therefore, implementing One-hot encoding, owing to the many unique values presented in ”ComponentID,” resulted in adding 128 columns to the dataset, making it excessively large for analysis. We employed feature selection to address this, as explained in the subsequent section.

Another issue that needs consideration is whether to keep or remove identifier features such as PanelID, PadID, and FigureID in a PCB dataset. On the one hand, retaining these identifiers can provide valuable insights for group-specific trends or tracking quality over time. They can also be crucial if we want to involve multiple measurements for each identifier, as they help account for data clustering. On the other hand, including these identifiers in predictive models can lead to problems such as data leakage or overfitting, as the model may unfairly use these labels to make predictions, thereby not generalizing well to new, unseen data. If the identifiers have no predictive power or relevance to the target variable, their presence may add noise to the analysis. Therefore, the decision to keep or remove these identifiers should be made carefully, considering both the aim of the analysis and the characteristics of the data. In our study, the identifier features come

with multiple unique values. Encoding these values would expand the dimensionality of the data, complicating the analysis of an already large dataset. Given these challenges, along with the fact that data clustering and pattern identification are not objectives of this study, we have chosen to remove these identifier columns to simplify the interpretation of our results.

3. Feature Selection: Feature selection is essential to data preparation, especially important for handling high-dimensional, complex datasets in machine learning. It serves two primary purposes: it simplifies models for better comprehension and helps clean the data (Li et al., 2017). This becomes particularly relevant for our project, where the dataset has multiple correlated features, as highlighted in Section 3.2.3. Additionally, our dataset suffers from a significant class imbalance, a topic we explore in a later subsection. While feature selection is not a direct solution for class imbalance, it indirectly aids in addressing this issue (Tiwari, 2014).

First, feature selection identifies and keeps the most informative features, discarding irrelevant or redundant ones. This can be particularly helpful for imbalanced datasets, making them more focused and manageable. Second, using feature selection helps to reduce the bias in models that might otherwise lean toward the majority class and become less influenced by noise. This allows for better learning from the minority class, thereby balancing the model. Lastly, by focusing on features that better represent the minority class, feature selection improves the model's generalization and prediction accuracy for that class.

In pursuing this crucial objective of refining the data, various techniques exist for feature selection, such as Recursive Feature Elimination (RFE), Least Absolute Shrinkage and Selection Operator (LASSO), Chi-Squared test, mutual information, and correlation-based methods (Effrosynidis & Arampatzis, 2021). Since the RFE method considers the interactions between features and is suitable for complex datasets, this method is decided to be utilized with different techniques on our dataset. RFE utilizes a greedy search algorithm to find the best subset of features for modeling (Kumari et al., 2023). The overview of the RFE Process is illustrated in Fig.3.3 and is further elaborated in the following.

(1) Utilize the selected RFE with ML algorithm to prioritize the significance of each feature.

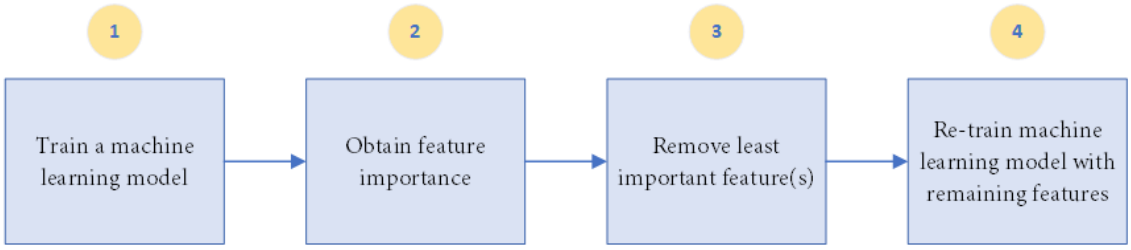


Figure 3.3: Overview of the RFE Process

(2) Remove the feature with the lowest importance.

(3) Construct a model utilizing the features that remain.

(4) Continue repeating steps 1-3 until you achieve the target number of features.

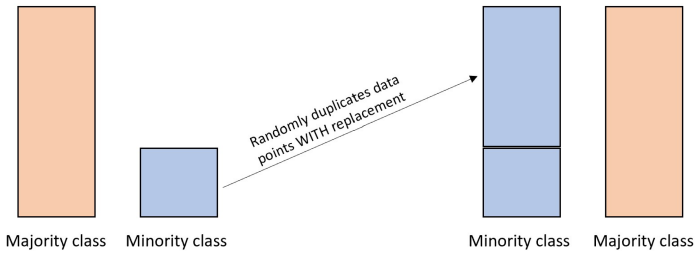
For achieving the best performance with RFE, three fundamental guidelines are recommended. First, experiment to find the optimal number of features for the model. Second, select an effective machine learning algorithm for ranking features. Third, set an appropriate number of cross-validation folds to prevent overfitting. Following these guidelines and searching over different hyperparameters for the RFE, we arrived at using a Decision Tree Classifier as an estimator, setting the desired number of features to 100, and choosing the cross-validation of 5.

4. Handling Imbalance Data: Before passing the data to ML and DL models, the last step is checking for data imbalance and handling it properly. The problem of class imbalance frequently arises in classification tasks, where the distribution of samples among various classes is not even (Bi & Zhang, 2018). In quality control fields, data imbalance can happen due to the oversaturation of common conditions or outcomes, while rare defects or failures may be underrepresented. This discrepancy can stem from the natural occurrence of these events or biases in data collection and monitoring processes. As a result, the imbalance may hinder the ability of quality control systems to detect and address rare but potentially critical issues accurately.

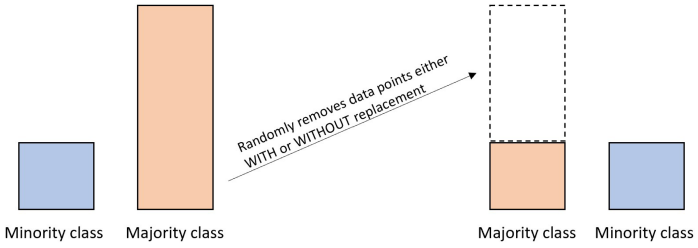
Various methods have been developed to address the class imbalance problem in machine learning. These methods either aim to reduce the algorithm's favoritism towards the majority class or make it more sensitive to the minority class. Among all, data-level methods often stand out for

their practical advantages. Data-level approaches are commonly employed to address class imbalance issues due to their simplicity, compatibility with various learning algorithms, and efficiency. These approaches are often the preferred solution in research, as they offer proven effectiveness without adding computational complexity (Liu et al., 2022). Owing to the advantages of data-level approaches, we chose to employ this technique. Moreover, because our dataset is significantly large and challenging to manage, opting for complex methods would unnecessarily complicate the computational process.

Oversampling and undersampling are two common data-level techniques used to address class imbalance in a dataset, and they exemplify the advantages mentioned earlier. Oversampling achieves this by adding more instances of the minority class, either through duplication or interpolation, thus preserving all original information. In contrast, undersampling balances the classes by removing instances from the majority class, which can make the dataset more manageable but may result in the loss of valuable information (Mohammed, Rawashdeh, & Abdullah, 2020). The processes for these methods are illustrated in Fig. 3.4.



(a) Random Oversampling



(b) Random Undersampling

Figure 3.4: Resampling Method Processes (Shamsudin et al., 2020)

Extending the analysis from the feature selection section, our dataset presents a notable imbalance that necessitates specific interventions for improved outcomes. Specifically, 99.6% of the data entries are labeled as 0, indicative of an absence of defects. To ameliorate this issue, we chose to

employ oversampling rather than undersampling. This decision was informed by the limitations inherent to undersampling in contexts of pronounced class imbalance, such as potential loss of crucial information and a significant reduction in dataset size. Given that the minority class accounts for a mere 0.4% of the total data, the use of undersampling would have adversely affected both the dataset size and the performance of our models. By leveraging oversampling techniques, we achieved a balanced dataset, elevating the representation of the minority class to 50% of the total data.

3.4 Proposed Models

After preparing the data, the next step is to pick the suitable models and train them using the dataset. In our research, we implemented several machine learning models and a single deep learning model to assess which ones are most effective for our large, imbalanced dataset. Subsequent sections will offer detailed explanations of the models we employed. We will also include a final subsection outlining the tools and libraries utilized during the model training process.

3.4.1 Decision Tree

This model is a well-known and easy-to-understand machine learning technique commonly applied in classification and regression problems. It operates by continuously dividing the dataset into smaller groups based on the attributes of the input features, eventually forming a tree-like diagram to illustrate the decision pathway. Every internal node in this tree signifies a decision criterion related to a feature, while each leaf node represents either a class label or a forecasted outcome (Quinlan, 1987).

3.4.2 Random Forest (RF)

This model is a robust machine learning technique that excels with large, structured datasets frequently found in sectors like finance, marketing, and manufacturing, where data-driven decision-making is crucial. This algorithm generates a collection of decision trees and combines their output using a vote-based system for the ultimate prediction. One of Random Forest's advantages is its

ability to manage data with many dimensions and recognize intricate nonlinear associations among variables. Additionally, it has a reduced tendency to overfit and can assist in identifying important features and ranking them (Breiman, 2001).

3.4.3 Extra Tree Classifier

This model, also called Extreme Randomized Trees, shares similarities with the Random Forest algorithm. Similar to its counterpart, the Extra Trees classifier creates a collection of decision trees and combines their forecasts for the final prediction. What sets it apart is the extra layer of randomness it adds by selecting feature thresholds at random rather than hunting for the best one. This added randomness avoids overfitting and enhances the model's ability to generalize (Geurts, Ernst, & Wehenkel, 2006).

3.4.4 Light Gradient Boosting (LightGBM)

LightGBM is a gradient-boosting model that has become increasingly popular lately. It is designed to handle large and high-dimensional datasets and offers quick training and prediction times while using minimal memory. The model utilizes histogram-based techniques to group data into bins, enabling efficient calculation of gradients, which is a key factor in its high performance (Ke et al., 2017).

3.4.5 Support Vector Machine (SVM)

SVM is a widely-used machine learning method employed in diverse areas. The core concept of SVM is to find the best hyperplane that divides the dataset into separate classes. A group of support vectors determines this hyperplane and the best one is chosen based on maximizing the margin—the distance between the hyperplane and the closest data points from each class. SVMs can manage data with many dimensions and identify intricate nonlinear relationships between variables via kernel functions. They have demonstrated strong generalization capabilities and have found successful applications in various real-world scenarios (Hearst, Dumais, Osuna, Platt, & Scholkopf, 1998).

3.4.6 One-dimensional Convolutional Neural Networks (1D-CNNs)

This type of neural network is commonly applied in tasks involving image and video analysis. A 1D-CNN is a type of neural network designed to handle one-dimensional data like time series or text. Unlike 2D-CNNs, which work with images, a 1D-CNN focuses on identifying patterns along a single dimension CNNs utilize convolutional layers to recognize spatial features in the input automatically and pooling layers to reduce the dimensions of the output from the convolutional layers. Convolutional layers slide a filter along the data sequence to produce a feature map, highlighting regions where particular features appear. This makes 1D-CNNs excellent for tasks that require understanding the local patterns or temporal dependencies in the sequence, such as signal or text classification and anomaly detection. (LeCun et al., 1995).

3.4.7 Tools And Libraries

In this study, we employed two primary Python libraries to implement the models discussed. For the ML models, we used **PyCaret** (*PyCaret — pycaret 2.3.5 documentation*, n.d.), a machine learning library that has rapidly gained traction in the data science and machine learning communities. This low-code, open-source framework streamlines the complete machine learning workflow, from data preprocessing and model choice to deployment and interpretation, facilitating the development of reliable and precise predictive models. This library simplifies critical steps in data preparation, such as data standardization and splitting. These features are especially beneficial given the large size of our dataset. The lengthy training and hyperparameter tuning processes for each model can be exhausting; however, utilizing PyCaret significantly reduces the computational time and effort required, making model implementation more manageable.

Moreover, for implementing the CNN model, we opted for the **PyTorch** library (*PyTorch*, n.d.). Widely recognized as a leading open-source library, PyTorch is commonly employed for DL applications, especially CNNs.

3.5 Evaluation and Results

In this part, we initially focus on crucial metrics essential for properly evaluating and optimizing our ML and DL models. Picking the right metrics is crucial because choosing incorrect ones can affect our understanding of the model's performance. Following this background, we will closely analyze the results from the proposed models, evaluating their effectiveness to draw solid conclusions.

3.5.1 Metrics

In this chapter, we have utilized precision, recall, and F1-score as metrics to compare the implemented models that are calculated as follows (Erickson & Kitamura, 2021):

$$Recall = \frac{TP}{TP + FN}, \tag{1}$$

$$Precision = \frac{TP}{TP + FP}, \tag{2}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}, \tag{3}$$

$$Specificity = \frac{TN}{TN + FP}. \tag{4}$$

where TP is True Positives, TN is True Negatives, FP is False Positives, and FN is False Negatives.

3.5.2 Results

While recognizing faultless components is undoubtedly beneficial, the primary objective of ML and DL models in the inspection process is to distinguish faulty components because these are the ones that have to be inspected in the AOI stage. This makes Recall, also known as Sensitivity, an essential evaluation metric that provides crucial information about the model's performance.

In our previous discussion, we noted that three key metrics were used to assess the performance

Table 3.4: Evaluation metrics of the proposed ML methods

Model	Dataset	Precision	Recall	F1-score
Random Forest Classifier	Train	0.997	0.999	0.998
	Test	0.630	0.398	**0.490**
Extra Tree Classifier	Train	0.998	0.999	0.999
	Test	0.665	0.355	0.463
Decision Tree Classifier	Train	0.992	0.999	0.996
	Test	0.298	0.322	0.310

of our various models. The results of the top-performing models are detailed in Table 3.4. According to these metrics from the test dataset, the Random Forest classifier outperforms other models with an F1-score of 0.49. This model underwent a five-fold cross-validation and was configured with parameters such as $criterion = gini$, $max_features = sqrt$, $min_samples_split = 2$, and $n_estimators = 100$. The relatively low performance of the models on the test dataset can primarily be attributed to an imbalance in the class distribution within the data. Besides ML algorithms, we also experimented with a DL model featuring 1D-CNN layers, complemented by various pooling and batch normalization layers. However, this CNN model did not yield significant improvements in detecting AOI defects. This limitation is likely since CNNs are mainly effective with data that exhibits spatial correlation among features, which our tabular dataset lacks. Therefore, CNN's inability to handle this data structure contributed to its poor performance in predicting AOI defects (Shwartz-Ziv & Armon, 2022). Consequently, the results from the 1D-CNN model were not included in the final evaluation table.

As mentioned before, oversampling was one of the key steps in our data preprocessing. The major risk associated with oversampling is the potential for overfitting. To mitigate this, we restrained how much we oversampled the training data while leaving the test data untouched. As a result, a noticeable difference emerged between the evaluation metrics of the training and testing stages. This can be explained by the differing balance ratios in the two datasets. Specifically, our

training set had a defect ratio of 25%, while the ratio in the test set was only 0.4%. One significant challenge we faced was the stark data imbalance; AOI defects made up just 0.4% of the entire dataset. It is worth noting that we chose to limit the extent of oversampling to prevent generating an excessive amount of duplicated or synthetic data, as this could lead to skewed results. Additionally, we were constrained by the computational power available, making it impractical to conduct a comprehensive analysis on an overly large dataset. We contend that the modest performance on the test dataset can be attributed to this issue. Therefore, our ML models exhibited a slight tendency to label components as being without defects, even when such defects existed. This is a common challenge when working with datasets that have an imbalance.

3.5.3 Comparing Our Methods to Existing Solutions

Although our predictive models may not efficiently identify defects in the AOI labels, they perform better than similar models in this specific context, and the defect detection is improved, as shown in 3.5. Other studies have achieved the highest F1-score of 0.42 and 0.44 by utilizing Artificial Neural Networks (ANN) and Random Forests, respectively (Taco et al., 2022; Tang et al., 2022). Additionally, the models in these studies not only had lower performance in the test set but also did not work well in the training set due to the neglect of the profound impact of data imbalance on the performance of ML models.

To improve the performance of our models, we shall employ algorithm-level methods for handling imbalanced data (Liu et al., 2022) and develop hybrid models. These techniques could significantly enhance the models' efficiency, potentially providing more accurate and efficient predictions of AOI defect labels.

3.6 Summary of The Chapter

This task contributes to the growing body of research in defect detection, specifically by examining the efficacy of machine learning methods on tabular datasets and emphasizing the importance of addressing data imbalance. In this task, we presented a methodology for predicting the AOI defects by merging two large datasets using a unique ID and implementing various machine learning and

Table 3.5: F1-score Comparison for AOI Label Prediction

Comparison Cases	**Train**	**Test**
Our Model	0.99	0.49
Case 1 (Gaffet, Roa, Ribot, Chanthery, & Merle, 2022)	0.43	0.41
Case 2 (Taco et al., 2022)	0.43	0.42
Case 3 (Tang et al., 2022)	0.43	0.44
Case 4 (Schmidt, Dingeldein, Hünemohr, Simon, & Weigert, 2022)	0.39	0.41

data processing techniques. The merging of datasets was done by concatenating the identifiers of the two datasets and then filling a new column based on the presence of the PCB component in the AOI. We employed data pre-processing steps such as data cleaning, one-hot encoding, feature selection, and oversampling to prepare the data for analysis. We balanced the highly imbalanced dataset using oversampling, as the minority class included only 0.4% of the whole data, and we eliminated the least essential features using Recursive Feature Elimination. Finally, we utilized machine learning and deep learning models to compare and evaluate their performance. Overall, our results showed that the Random Forest and Extra Tree classifiers outperformed other machine learning and deep learning techniques in terms of their performance.

Furthermore, algorithm-level methods for handling imbalanced data will be employed to enhance the performance of the models, and hybrid models will be developed. These techniques can significantly improve the models' efficiency, resulting in more accurate and efficient predictions of AOI defect labels.

Chapter 4

Predict the Human Inspection Label By Utilizing TabNet Model

In the previous chapter, we focused on using ML methods during the early stage of PCB manufacturing – specifically, during solder pasting – to identify flaws and prevent subsequent defects proactively. However, it is important to acknowledge that defects can arise at various stages of PCB production, each process carrying its own inherent risks. To maintain impeccable PCB quality and minimize potential issues throughout the entire manufacturing process, it is imperative to implement an advanced system capable of promptly recognizing defects at each stage. Such a system should effectively intercept defective PCBs and ensure they meet the highest quality standards before proceeding, thereby preventing significant costs for the manufacturer.

This chapter looks into investigating the quality of PCBs after component placement. Our focus shifts to employing diverse ML and DL methods for defect detection during the AOI stage of PCB production. Through comprehensive analysis and experimentation, we aim to enhance defect detection capabilities, enabling the timely identification and resolution of potential issues. By employing state-of-the-art ML and DL techniques, we endeavor to elevate PCB quality and ensure that only flawless PCBs advance to subsequent stages of production. This, in turn, guarantees increased operational efficiency, reduced manufacturing costs, and heightened customer satisfaction.

The subsequent sections of this chapter are structured as follows: Section 4.1 addresses the

problem statement, while Section 4.2 provides an overview of the dataset employed in developing the proposed ML models and outlines the steps taken for its preparation. Additionally, Section 4.3 introduces the ML and DL models proposed for defect detection, along with a discussion of the implementation process and the resulting outcomes. Finally, Section 4.5 presents the concluding remarks for this chapter.

4.1 Problem Statement

In the previous chapter, we discussed that after the AOI stage at the end of the production line (shown in Figure 1.1), operators manually examine PCB components to validate the labels assigned by the AOI stage and add a new label named ”OperatorLabel.” This examination is necessary since correct labeling is crucial to avoid costly consequences for the manufacturer. For instance, if the AOI inspection mistakenly marks a good PCB as faulty, it might move to another inspection step. This can lead to unnecessary expenses and time spent fixing or remanufacturing the entire board. On the other hand, if the PCB is truly flawed, but the automatic system labels it as non-defective, the flawed PCB will continue along the production line; as it progresses closer to becoming a final product, more cost and resources are invested. Therefore, discovering the problems later ends up a waste of more money and material. Plus, because the PCBs will be used in electronic devices, any undetected defects could be risky for workers and customers.

To reduce the mentioned risks, in PCB production lines with error probabilities, a designated operator is responsible for reviewing AOI labels, controlling the PCBs, and making the final decision regarding the PCBs’ condition. Hence, owing to the importance of this step, we aim to automate this step by predicting the label generated by the human operator, named ”OperatorLabel” in our dataset, as the target variable for this chapter. This variable has two labels: ”Good” and ”Bad”. The ”Good” label indicates that the AOI mistakenly detected a defect, and the component should not have proceeded to the inspection stage. On the other hand, the ”Bad” label confirms a genuine defect identified by AOI.

Although utilizing human inspection brings unique qualities to defect detection processes and operators can distinguish between false alarms and actual defects, using ML and DL models for

defect detection offers advantages over relying solely on human operators. ML and DL models can analyze vast amounts of data rapidly and consistently, reducing the chances of human error. They have the ability to learn from patterns and anomalies, potentially uncovering defects that operators might miss, all while maintaining continuous monitoring. This combination of speed, consistency, and capacity to spot minor variations makes ML and DL models a valuable tool for efficient and accurate defect detection. That is why we aim to create ML and DL models that can replace the operator and predict the OperatorLabels to enhance the quality of the defect detection process. These models will help speed up the detection process and provide valuable insights for ensuring PCB quality. Combining human expertise with automated inspection methods can close the gap between manual assessments and AOI systems. This integration will enhance overall manufacturing efficiency and help ensure a more reliable PCB production process.

Consequently, the goal of this chapter is to compare the proficiency of ML and DL models and human operators in detecting flaws in PCB boards. This aims to assess the feasibility of replacing human operators with automatic machine decision-makers, leading to a more cost-effective and time-saving quality control process.

4.2 Dataset Description and Data Preparation

The data analyzed in this chapter consists of a combination of SPI and AOI datasets. The following section will detail essential procedures that are integral to the data preparation, aiming to create a uniform dataset for analysis. This preparation includes several key processes, such as label generation, data conversion, cleaning, scaling, and feature reduction. By executing these steps, the data is prepared, which is vital for minimizing errors in the final prediction.

4.2.1 Label Generating

Based on what is discussed in the Problem Statement section (4.1), our objective in this section is to predict the OperatorLabel by utilizing both the SPI and AOI datasets. The AOI dataset solely contains labels without any additional features, necessitating merging the two datasets. To achieve this goal, the unique ID, mentioned in section 3.3, must be generated by concatenating the four

common features found in both datasets: PanelID, FigureID, ComponentID, and PinNumber. Upon merging the datasets, certain columns such as Date, Time, and ConcatenatedID, which hold no meaningful information, will be removed. This process results in a tabular dataset comprising 23 features and 22,450 rows.

4.2.2 Data Cleaning

In preparing the data, the next essential step involves addressing missing values and duplicate entries. Fortunately, the data used in this chapter is free from any missing values, eliminating the need to implement techniques to fill these gaps. Additionally, 43 rows of data was identified as duplicated and removed from the dataset. After that, the index was reset to maintain a smooth and ordered data sequence. This action ensures that our data is consistently organized and avoids potential problems when performing various operations.

Another aspect that requires attention is preventing data leakage and attaining unbiased performance assessments. This leads us to employ distinct train and test datasets with identical features. Failure to do so can lead to improper functioning of the models. Consequently, it is crucial to examine all the feature values carefully. Notably, the ”Result” and ”ComponentID” columns exhibit some differences between the train and test sets. In order to address this disparity, specific adjustments have been made to achieve alignment between these columns to make the train and test sets consistent for feeding in Machine Learning models. As a result, in the train set, the following entries have been removed from the Result column: E.Bridging, W.Position, W.Excessive, and E.HeightU. Additionally, the entries C12, C3, D8, DZ1, L3, R41, TR2, TRB2, and TRB7 have been eliminated from the ComponentID column. Similarly, the test set necessitates the removal of C32, C4, and R8 from the ComponentID column. All columns in the train and test sets will be harmonized by implementing these modifications. It is important to note that the amount of training data removed for achieving the mentioned harmonization is only 0.3 percent, while for the test data, it is 0.08 percent. These percentages are minimal and can be considered negligible.

4.2.3 Data Encoding

After completing the preceding steps, the next essential task is to encode the data. This is necessary because the majority of machine learning algorithms demand numeric data for processing. The ”Good” and ”Bad” labels in the ”OperatorLabel” are encoded as 1 and 0, respectively, using LabelEncoder, allowing seamless integration into the machine learning models for accurate predictions. Additionally, OneHot encoding is used to convert other non-numerical features, such as ”ComponentID,” ”Result,” ”MachineID,” and ”AOILabel,” from categorical to numerical, which transforms the size of the features from 20 to 120. Moreover, the prepared data requires an additional adjustment in the form of a data-type conversion. This is essential as ML models can only handle integer or float data types, not ”object” data. Despite the previous step’s encoding, the features mentioned retain their ”object” data type and need to be changed to float. Consequently, the data type of ”ComponentID,” ”Result,” ”MachineID,” and ”AOILabel” is altered to float for seamless compatibility with the ML models.

4.2.4 Feature Scaling

Data scaling and standardization are crucial pre-processing steps in machine learning. They ensure that the dataset’s features are on a similar scale, which can significantly improve the performance of many ML and DL algorithms. Given that the features in the data utilized in this chapter exhibit varying scales and fluctuate across a broad range, data standardization has been adopted at this stage. While different data scaling methods such as Quantile Transformer, Max Abs Scaler, and Min-Max Scaler can be applied, choosing the appropriate data scaling method depends on the underlying distribution of the data, the presence of outliers, and the specific requirements of the model being used. Experimentation with different scaling methods, analysis of the data’s statistical properties, and consideration of the algorithm’s assumptions guide the selection of the most suitable scaling technique to ensure optimal model performance. To illustrate this process in practice, three well-known and widely used scaler methods have been implemented on the data. These methods, each briefly summarized in the following, were applied to the data and subsequently compared to one another to evaluate their respective performances.

(1) **Standard Scaler**: The Standard Scaler, also known as Z-score normalization, is a pre-processing technique used to standardize the features of a dataset. It transforms the data into a distribution with a mean of 0 and a standard deviation of 1 (Thara, PremaSudha, & Xiong, 2019). This method is widely used in machine learning algorithms sensitive to the scale of input features, such as SVM and k-nearest Neighbors (k-NN). By transforming features to have the same scale, the Standard Scaler ensures that all features contribute equally to the computation of distances in these algorithms, which can lead to better performance and stability in the learning process. However, this approach is easily affected by outliers because it relies on the mean and standard deviation of the data, which outliers can heavily influence. This causes the standard data to be scaled into a small range while the outliers dominate the scale, thus affecting the transformed data. The formula to scale with this method is shown below:

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma} \tag{5}$$

where x is the original value, μ is the mean and σ is the standard deviation.

(2) **Min-Max Scaler**: This method scales the data by shifting and rescaling it, usually to the range [0, 1] or sometimes [-1, 1]. The application of Min-Max scaling is particularly beneficial in algorithms sensitive to the input features' scale, such as gradient descent-based methods and distance-based algorithms like k-NN (Powers, 2020). By adjusting all the features to a uniform scale, the method ensures that no individual feature dominates the others merely because of its size. Consequently, this helps to refine the learning process, often leading to an enhancement in the model's performance. On the other hand, the method is particularly sensitive to outliers, as extreme values can shift the min and max, affecting the scaling. The formula for Min-Max scaling is:

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{6}$$

where X_{min} and X_{max} are the minimum and maximum values in the feature column.

(3) **Robust Scaler**: Robust Scaler is a pre-processing technique that scales features using the median and interquartile range, making it resistant to outliers. It is particularly beneficial when the data contains extreme values that might skew the scaling process (*Comparing different preprocessing techniques available in scikit-learn*, 2007). The scaled values with this method are computed as follows:

$$X_{\text{scaled}} = \frac{X - \text{median}}{\text{IQR}} \tag{7}$$

where IQR is the Interquartile Range which is a measure of statistical dispersion and is calculated as the difference between the third quartile ($Q3$) and the first quartile ($Q1$) in a dataset.

$$\text{IQR} = Q3 - Q1 \tag{8}$$

Using less sensitive statistics to outliers, Robust Scaler ensures a more accurate representation of the data's underlying distribution. This approach is commonly used in machine learning algorithms where the presence of outliers can adversely affect performance, providing a way to standardize features without being unduly influenced by anomalous observations.

Having explored the various data scaling techniques, it is imperative to identify which method best fits our particular dataset. The choice among Standard Scaler, Min-Max Scaler, and Robust Scaler should be guided by an in-depth understanding of the data's characteristics, the algorithm's requirements, and the specific goals of the analysis. Robust Scaler differs from Min-Max and Standard Scalers in its insensitivity to outliers and its lack of assumptions about a specific data distribution. The Min-Max Scaler confines values within a defined range, which might suppress outlier effects, but this may not be appropriate if those outliers are vital to the analysis. Conversely, the Standard Scaler is prone to misrepresentation by extreme values. Robust Scaler effectively addresses these challenges, making it an adaptable choice when managing anomalous observations or when there are no particular data distribution assumptions. As such, for datasets like ours, where significant outliers are present, the Robust Scaler is often the more suitable option, while Standard

and Min-Max Scalers might be better suited to other scenarios. In our study, we applied all three mentioned scalers to our dataset, and as expected, the Robust Scaler emerged as the best fit, particularly due to significant outliers in our data. This method handled these anomalies effectively, enhancing the overall performance of our ML model. The results confirmed the importance of selecting the appropriate scaling technique based on the specific characteristics of the data.

4.2.5 Feature Selection

As previously stated, our dataset expanded significantly by applying one-hot encoding to the categorical features. Consequently, feature selection becomes crucial in dealing with such large datasets in the machine learning process. This involves selecting the most pertinent features to build effective predictive models. It improves accuracy by eliminating redundant or irrelevant attributes, allowing the model to concentrate on the most critical aspects of the data. This step also minimizes overfitting by simplifying the model, accelerates training time, and enhances interpretability (Remeseiro & Bolon-Canedo, 2019). It also helps reduce computational costs, particularly in large-scale applications like our work.

The RFE method (See Section 3.3.2) is chosen for our dataset for its skill in managing complex feature interactions. For optimal results, it is essential to balance the model's complexity through feature experimentation, select the suitable ML algorithm for feature ranking, and set an appropriate number of cross-validation folds to avoid overfitting. These combined approaches make for a compelling feature selection strategy.

Building upon these principles, we conducted experiments with the RFE method, choosing various quantities of features, including sets of 100, 90, 80, 70, and 60. We also implemented varied machine learning algorithms such as Random Forest, Logistic Regression, Decision Tree, Gradient Boosting, and Perceptron on the data to compare the results. We attempted a two-stage feature selection process to enhance the outcomes by combining two different ML algorithms. This approach leverages the strengths of both methods to refine the selection process, leading to improved performance and robustness. For instance, Logistic Regression and Gradient Boosting can be combined in a two-stage method. In the first stage, Logistic Regression is used to assess feature importance,

which is particularly useful in binary classification. It provides foundational insight into the feature-to-target relationship and helps to eliminate less significant features. Following this reduction, the second stage employs Gradient Boosting, known for its performance and ability to understand complex non-linear relationships. Gradient Boosting iteratively refines feature importance rankings by learning from previous models' errors. Consequently, the superiority of this two-stage approach over other methods is evident in its precision, flexibility, robustness, and efficiency. By merging both linear and non-linear perspectives, it leads to more accurate feature selection. Its adaptability makes it suitable for various data types, and its sturdiness helps prevent overfitting by enabling the models to compensate for each other's weaknesses (Pahwa & Kumar, 2017). Furthermore, the quick filtering of irrelevant features by Logistic Regression enhances the efficiency of the Gradient Boosting stage.

Besides implementing a two-stage feature selection of Logistic Regression and Gradient Boosting, we have conducted systematic experiments with several of these two-stage combinations, thoroughly analyzing their behavior and evaluating their performance against various criteria. The detailed findings, including insights into the advantages and limitations of each combination, are provided in "Experimental Results", section 4.4.

4.2.6 Handling Imbalance Data

As noted earlier in Chapter 3, the next step before feeding the data to ML models is to check for data imbalance and handle it appropriately. Due to our dataset's highly imbalanced nature, addressing this issue is a crucial step, especially since it can significantly affect the reliability of ML predictions. The "OperatorLabel" feature exhibits a substantial imbalance in the prepared dataset, with a ratio of 95 to 5. This indicates that 95 percent of the data is labeled as "Good" from the operator's perspective, while only 5 percent of the PCB boards were truly defective and required inspection. Such imbalance may lead to inaccuracies in our model, necessitating effective management.

Several techniques exist for addressing imbalanced data within the target column, and one prominent method, oversampling, was explored in the previous chapter (Chapter 3). While oversampling can effectively deal with data imbalance in specific scenarios, it may fall short when faced with a significant imbalance ratio. Recognizing this limitation, in this chapter, we have ventured

beyond oversampling to explore other methods and combinations thereof to tackle the imbalance problem comprehensively.

Building on that, this chapter also investigates a wider range of techniques beyond oversampling. These techniques have been crafted to either minimize an ML model's bias towards the majority class in a dataset or to fine-tune the model to be more attentive to the minority class, which are categorized into three main types, shown in figure 4.1: data-level methods focusing on resampling the data, algorithm-level methods that alter the classification algorithms, and hybrid methods that merge the strengths of both (Wongvorachan, He, & Bulut, 2023). Our research primarily focuses on evaluating data-level approaches, which aim to create a balanced dataset with equal class distribution to lessen the bias in ML algorithms. These data-level methods feature three key resampling tactics: oversampling, undersampling, and combined sampling techniques. While our primary emphasis remains on data-level approaches, we have also explored algorithm-level and hybrid methods to some extent for a more comprehensive understanding of imbalance solutions. In the following sections, various approaches implemented on our dataset are explained in detail.

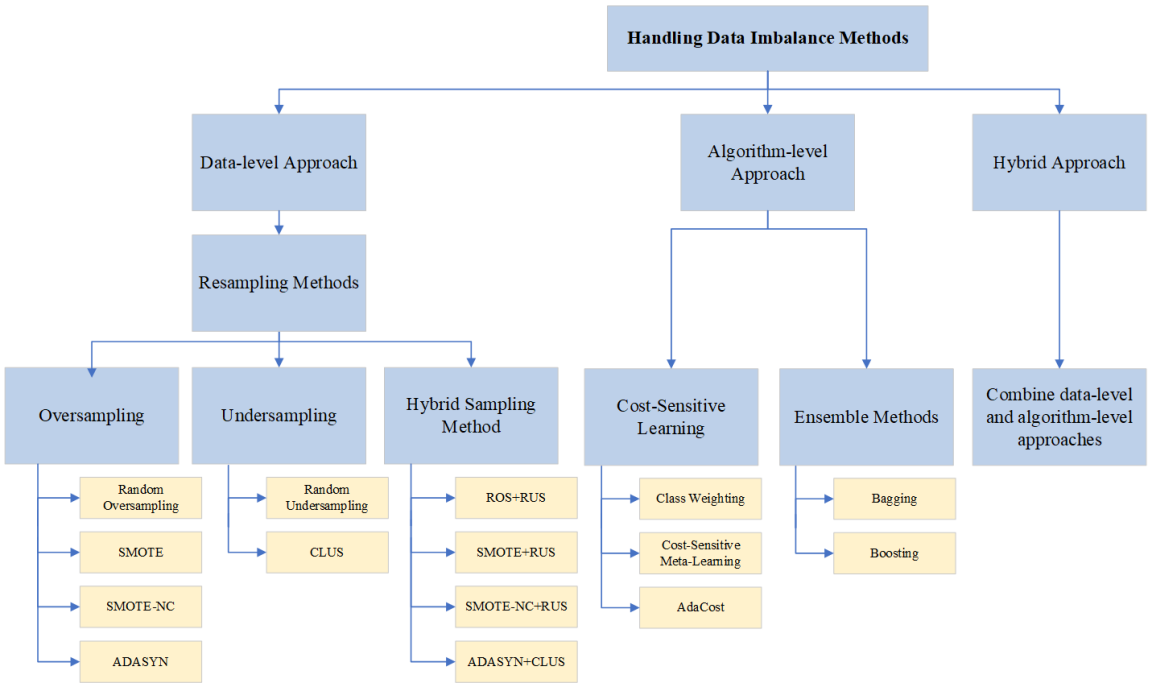


Figure 4.1: Data Imbalance Handling Methods

(1) **Resampling Methods:** These methods for handling imbalanced data include oversampling

techniques like Random Oversampling (ROS), SMOTE, and ADASYN, which increase minority class instances, and undersampling techniques like Random Undersampling (RUS), NearMiss, and Tomek Links, which reduce majority class instances. These methods can be tailored to the dataset's specific needs, either individually or in combination, to ensure a balanced class distribution and improve model performance. The methods employed in our study are detailed in the subsequent section.

- **SMOTE:** Synthetic Minority Oversampling Technique (SMOTE) works by creating synthetic samples in the feature space of the minority class instead of replicating the instances. It does this by selecting existing instances and generating new instances that are similar but slightly altered. Specifically, given a sample x_i from the minority class, the algorithm first selects one of its k nearest neighbors x_{zi}, also from the minority class (Fernández, Garcia, Herrera, & Chawla, 2018). A synthetic sample s is then generated using the following equation:

$$s = x_i + \lambda \cdot (x_{zi} - x_i)$$

  where λ is a random number between 0 and 1. This process creates a new instance that lies between the original instance and one of its neighbors, allowing for more diverse representation of the minority class (Fernández et al., 2018).

- **SMOTE-NC:** SMOTE for Nominal and Continuous features (SMOTE-NC) is a variation of SMOTE specifically designed for datasets with a mix of categorical and continuous features (Mukherjee & Khushi, 2021). It applies the SMOTE algorithm separately to the nominal and continuous parts of the feature space, ensuring that the synthetic instances are meaningful given the different nature of these types of features.

- **Combining ROS and RUS:** Combining ROS and RUS is a strategic approach for handling imbalanced data in classification tasks (Shamsudin et al., 2020). As mentioned before, Oversampling involves increasing the number of instances in the minority class, either by duplicating existing samples to make it more representative. Undersampling, on the other hand, reduces the number of instances in the majority class, usually by randomly removing samples. By simultaneously increasing the minority class size and decreasing the majority

class size, this combined approach aims to create a more balanced class distribution, thereby enhancing the model's ability to learn from both classes without being biased towards the majority class.

(2) **Class Weighting:** This method is another popular technique for handling imbalanced data, especially in the context of classification problems. In a dataset with imbalanced classes, traditional algorithms may lean towards predicting the majority class since the higher occurrence biases them. This leads to poor performance in correctly classifying the minority class. This method aims to rectify this by assigning different weights to classes in the loss function. The weights are inversely proportional to the class frequencies. For example, the minority class is assigned a higher weight, while the majority class is assigned a lower weight. This weighting scheme aims to make the algorithm more sensitive to the minority class. When a misclassification occurs in the minority class, it results in a higher loss compared to a misclassification in the majority class. Thus, during the training process, the model is penalized more for making mistakes in the minority class, which encourages it to pay more attention to getting these instances right. Class Weighting can be applied in various ML algorithms, and it provides a straightforward way to give more importance to the minority class without needing to alter the original dataset. This approach is beneficial in scenarios where oversampling the minority class or undersampling the majority class may not be suitable or desirable.

(3) **Ensemble Methods** Ensemble methods are powerful techniques to address imbalanced data, especially in classification tasks where one class significantly outweighs the others. In essence, ensemble methods combine multiple models to create a more robust classifier. For handling imbalanced data, techniques like bagging and boosting are often used. Bagging involves creating subsets of the original dataset, with a balanced proportion of minority and majority classes, and training multiple models on these subsets. Boosting, on the other hand, focuses on training subsequent models in the instances where previous models performed poorly, thereby giving more weight to the minority class. When predictions are made, these individual models "vote" on the most likely class, yielding a prediction that considers the information from various perspectives. This often leads to a more balanced and accurate classification of

the minority class.

After exploring the approaches outlined earlier, we have summarized a comparison of these methods and their various combinations in Table 4.1. The evaluated methods include ROS, RUS, SMOTE-NC, as well as combinations like ROS with RUS, SMOTE-NC with RUS, and an ensemble of SMOTE-NC and RUS incorporating Class Weighting. This table provides a side-by-side look at each approach's operating principles, advantages, and disadvantages.

Table 4.1: Comparison of Handling Imbalanced Data Methods

Approach	Operating Principle	Advantages	Disadvantages
ROS	Randomly duplicates minority samples with replacement.	- Provides more replicated minority samples.	- Increases the likelihood of overfitting by introducing replicated samples. - Increases training time of the predictive model.
RUS	Randomly removes majority samples with or without replacement.	- Reduces excessive majority samples. - Decreases training time of the predictive model.	- Potential to exclude helpful information.
SMOTENC	Creates synthetic minority samples by interpolating between existing instances and nearest neighbors, handling nominal and continuous features.	- Adapts to both numerical and categorical features in the minority class. - Provides more diverse training data without simple duplication, promoting model robustness.	- Introduce noise by creating synthetic samples, leading to over-generalization.
ROS+RUS	Randomly duplicates minority samples with replacement, then randomly removes majority samples to balance the class distribution.	- Blends oversampling of minority with undersampling of majority for balance. - Avoids majority class bias, enhancing performance. - Provides flexibility in adjusting class distribution.	- Potential overfitting through replication of minority samples. - Possible loss of essential information by undersampling majority class. - Demands careful tuning to balance sampling, increasing preparation complexity.
SMOTENC+RUS	Creates minority samples using nearest neighbors, and a specific minority sample ratio, while also allowing for random removal of majority samples.	- Provides non-replicated minority samples - Can handle categorical variables - Less overfitting chance than ROS - Decreases training time of the predictive model	- Suboptimal performance with high-dimensional datasets - Potential to create noise samples - Negligence of local K-neighbor parameter - Potential to exclude helpful information
SMOTENC+RUS +Class Weighting	Synthesizing minority samples, reducing majority samples, and adjusting the model's focus on classes, aiming to balance the class distribution in the dataset.	- Enhanced focus on minority class - Applicable to various kinds of data (nominal and continuous) - Offers a comprehensive solution for various imbalanced scenarios. - Customizable and potentially boosted performance	- Increased complexity in model tuning - Risk of bias towardsminority class - Sensitivity to parameter choices - Potential for overfitting - Incompatibility with certain algorithms

4.2.7 Data Splitting

Data splitting is a vital step in data pre-processing, mainly to avoid overfitting, evaluate model performance, assist in model selection, ensure fair assessment, reflect real-world performance, and prevent data leakage. It involves dividing the data into training, validation, and test sets, enabling an unbiased evaluation of the model and helping to choose the best-performing model that generalizes well to unseen data. Moreover, to split a dataset into training and test sets using the popular function **train_test_split** from **scikit-learn** in Python, the default behavior is to shuffle the dataset randomly before splitting. This means that the samples are randomly drawn without replacement, ensuring that the training and test sets are random subsets of the original data (Reitermanova et al., 2010). When we want to determine how to split data, a specified ratio must be chosen. Popular options include 80:20, where 80% of the data goes towards training and the remaining 20% for testing. Other common proportions like 70:30, 60:40, or even 50:50 are also utilized. The ideal ratio for any particular dataset is not well-defined, however, the 80:20 split is more common and is justified by the Pareto principle (V. R. Joseph, 2022). Although an 80:20 ratio is often used for splitting data, in this study, two distinct datasets are already available. Therefore, the entire training dataset is used for model learning and hyperparameter tuning. The separate test dataset is reserved exclusively for evaluating the performance of the models.

4.3 Model Implementation

Similar to the previous chapter, once the data has been prepared and preprocessed, the next step involves selecting the models that are most likely to yield the best results. In the context of binary classification with big and imbalanced datasets, tree-based ML models like Random Forest, Extra Tree Classifiers, and Gradient Boosting Machines (such as XGBoost) have proven to be effective owing to their ability to capture complex non-linear relationships in the data, their interpretability, and their flexibility in handling different types of input variables. Random Forest is robust against imbalance, while Gradient boosting can be fine-tuned for imbalanced classes through weighted adjustments, and Extra Tree Classifier is ideal using ensemble learning and randomization to handle class imbalances effectively. These tree-based models inherently handle imbalanced data well due

to their hierarchical structure. They make decisions based on rules, splitting the data across various features, which can be tuned to better capture the minority class. They also benefit from ensemble techniques, combining multiple weak learners to create a more robust model.

On the deep learning front, specialized neural networks designed with weighted loss functions and appropriate activation functions like Sigmoid have shown promising results. These deep learning models can be adapted to address the class imbalance by carefully calibrating class weights and utilizing relevant loss functions. Building on these established ML and DL approaches, a recent innovation in the field is TabNet, which uniquely combines the characteristics of decision trees with the power of neural network architectures. This model uses decision-making paths similar to decision trees but leverages the expressive power of neural networks. TabNet can learn complex patterns from large datasets and is known for being interpretable, scalable, and able to handle imbalanced data. By utilizing attention mechanisms, TabNet can focus on essential features and ignore irrelevant ones, making it a practical option for binary classification with imbalanced datasets. As a result, the combination of these ML and DL techniques provides a flexible and powerful toolkit for tackling the unique challenges posed by large and imbalanced binary classification problems. In summary, both tree-based models and TabNet offer powerful solutions for big and imbalanced datasets, bringing together the strengths of traditional machine learning and modern deep learning techniques to address the challenges of binary classification.

While the previous chapter detailed the underlying principles and mechanics of various tree-based machine learning models, the focus of this section shifts to a more in-depth exploration of TabNet. In the following, we will begin by discussing the architecture and application of the TabNet model, and then present the classification results for predicting the OperatorLabel, allowing us to compare the proficiency of various models.

4.3.1 TabNet

TabNet is a deep learning model that falls under the category of tabular data models. It is designed specifically for handling tabular datasets, where the data is structured in rows and columns like spreadsheets. This model is based on the idea of the attention mechanism and employs a unique combination of attention and sparse feature selection to make accurate predictions on tabular data.

The TabNet model has gained popularity due to its ability to handle high-dimensional and sparse tabular data efficiently. It uses the concept of adaptive attention to focus on relevant features while ignoring irrelevant ones, which helps in improving the model's predictive performance (Arik & Pfister, 2021). Although TabNet has the potential to achieve impressive results, it does come with some challenges. It often requires a considerable amount of time to run, which may be linked to its complex nature or the data it handles (McDonnell, Murphy, Sheehan, Masello, & Castignani, 2023). Additionally, finding the right settings, known as hyperparameters, can be a time-consuming process. These factors combined mean that using TabNet effectively can require both significant computational effort and careful tuning.

The primary application of the TabNet model is in tabular data tasks such as classification and regression. It can be used in various industries and domains, including finance, healthcare (L. P. Joseph, Joseph, & Prasad, 2022), e-commerce, and customer analytics, where structured data is prevalent. TabNet is particularly useful when dealing with datasets with a large number of features, missing values, or imbalanced classes.

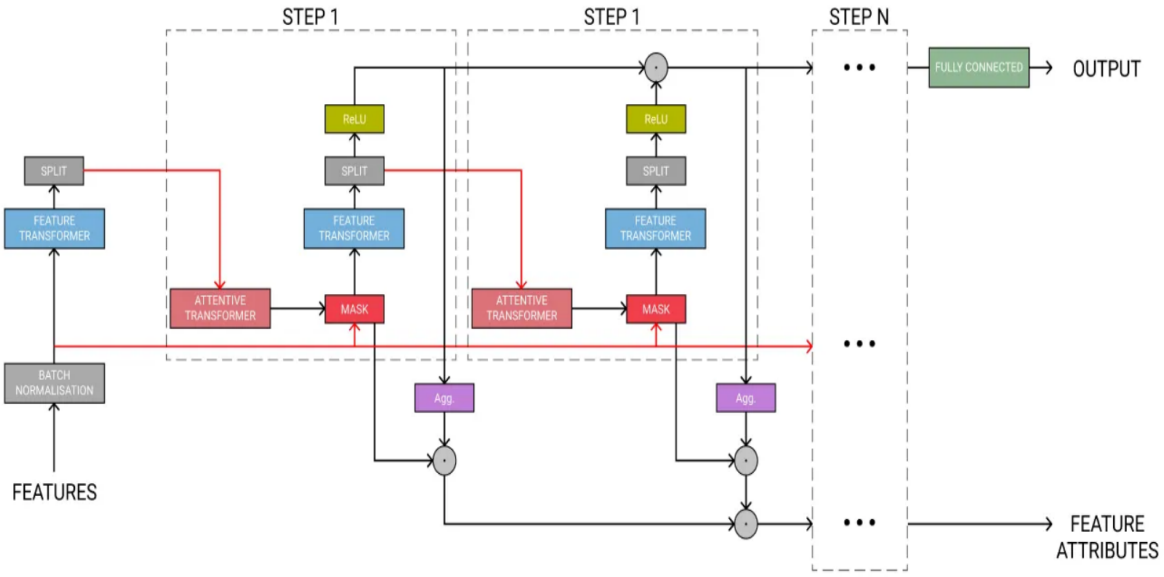


Figure 4.2: TabNet Model Architecture (Arik & Pfister, 2021)

TabNet's architecture is built on several key components, including an attention mechanism that

prioritizes essential features, a split decision process that mimics traditional decision trees, feature transformation techniques, and a specifically tailored loss function for training. These components work in unison to enable TabNet to capture complex nonlinear relationships in the data, making it both interpretable and scalable. In the following sections, we will break down these elements, providing a mathematical insight into how TabNet operates, and what makes it a valuable tool in the modern machine learning toolkit.

1. Attention Mechanism: The attention mechanism is central to TabNet's design. It determines how much focus the model should place on each feature during every decision step, ensuring that the most relevant features are prioritized. This is achieved through the calculation of attention weights w_i, which are computed using the exponential function. The higher the weight, the more attention the feature receives. The attention weights can be expressed as:

$$w_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}, \tag{9}$$

where w_i is the attention weight for the i-th feature, and s_i is a score calculated for that feature based on its relevance to the decision-making process.

2. Split Decision: The split decision determines how the data is partitioned at each step of the decision process, mimicking the decision-making paths of traditional decision trees. TabNet uses a neural network layer to learn these split decisions, represented by:

$$M = \text{Relu}(b + W \cdot \text{previous output}), \tag{10}$$

where M is the mask for the decision step, W is a weight matrix, b is a bias term, and Relu is the Rectified Linear Activation function that introduces nonlinearity. This helps the model to learn complex patterns and decisions.

3. Feature Transformation: Before feeding features into the attention mechanism, TabNet often applies transformations to make them more suitable for processing. This involves using a linear transformation followed by a non-linear activation function. This could be represented by:

$$F = \text{Relu}(W \cdot x + b), \tag{11}$$

where F is the transformed feature, W is a weight matrix, x is the input feature, and b is a bias term. his transformation enables TabNet to learn more complex relationships between features and outcomes.

4. Loss Function: TabNet is trained by minimizing a loss function, which quantifies how well the model's predictions align with the true labels. For binary classification, this might be the binary cross-entropy loss, given by:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)], \tag{12}$$

where N is the number of samples, y_i is the actual label, and $\hat{y}_i$ is the predicted label. The loss function drives the model to make predictions close to the true labels, adjusting its internal parameters accordingly.

4.4 Experimental Results

In this section, we first highlight the key metrics vital for correctly assessing and fine-tuning our ML and DL models. As stated before, the importance of choosing the correct metrics is substantial, as wrong ones can mislead the models' results interpretation. Given the importance of choosing the right metrics, we extend our focus to explore Class-level and Coefficient-based metrics in the following section. After setting this context, we will closely examine the outcomes from each proposed model, comparing their efficiency to make well-founded conclusions.

4.4.1 Metrics

Selecting the most suitable metrics for evaluating our model primarily depends on the specific characteristics of our data and the objectives we aim to achieve. Given that our main focus is on identifying defects to prevent further costs down the PCB production line, class-level metrics become particularly relevant for us, especially since the cost of false negatives is significant. Alternatively,

Coefficient-based metrics come into play when dealing with imbalanced datasets or when different errors have varying cost implications. In our case, misclassifying a healthy PCB component as defective can result in additional costs for the manufacturer, either through unnecessary further inspection or by discarding the board. Therefore, Coefficient-based metrics provide a well-rounded assessment by taking into account multiple facets of the model's performance. Further information on both types of metrics will be discussed in the following.

(1) **Class-level Metrics**

Class-level metrics serve as specialized evaluation tools in ML to measure how well a classification model performs for each unique class in the dataset. As we have discussed before, these metrics are particularly important when dealing with imbalanced datasets or when different classes have varying costs of misclassification (Picek, Heuser, Jovic, Bhasin, & Regazzoni, 2019). Key metrics such as Precision (see Section 2), Recall (see Section 1), F1-score (see Section 3), and Specificity (see Section 4) can be calculated for individual classes, offering a detailed understanding of the model's performance.

(2) **Coefficient-based Metrics**

Unlike class-level metrics, which offer a detailed evaluation for each unique class, coefficient-based metrics usually provide a single value that summarizes the quality of a model's predictions across all classes or data points. This summary value can be especially useful for comparing different models quickly or for scenarios where a comprehensive view of model performance is desired. Coefficient-based metrics are often employed when ease of interpretation is a priority, as they clarify complex data into a single, understandable figure. Among these metrics, the Matthews Correlation Coefficient or MCC stands out for its effectiveness in providing a balanced view of classification performance, especially in imbalanced datasets (Matthews, 1975). Details about this metric are elaborated in the subsequent section.

- **MCC** is a statistical measure used in binary classification to evaluate the quality of predictions. It takes into account true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), and it is generally considered a balanced measure since it can be

used even when the classes are of very different sizes (Chicco & Jurman, 2023). Additionally, MCC is a robust metric often used in imbalanced datasets where one class significantly outnumbers the other. It offers a more insightful assessment of the model's performance compared to simpler metrics like accuracy.This metric expressed as:

$$\text{MCC} = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

.

This metric returns a value between -1 and +1, where a score of +1 denotes flawless forecasting, a score of 0 implies a prediction equivalent to random guessing, and a score of -1 indicates a complete mismatch between the predicted and actual outcomes.

4.4.2 Results

As previously mentioned, the data has been prepared for analysis by the five ML and DL models discussed in Section 4.3. Moreover, it is noteworthy to mention we used various hyperparameter settings by grid search to train these models and employed 5-fold cross-validation for dependable and sturdy evaluation. While 10-fold cross-validation is an option, we opted for 5-fold due to the time-intensive nature of testing multiple hyperparameters with a 10-fold approach. The results for these models are provided separately in the following.

Table 4.2 displays the performance of the RF model in predicting the 'OperatorLabel' for both the training and test sets. The model was trained using various hyperparameters. The best performance was achieved with a class defect ratio of 0.9, a 'min_samples_split' parameter set to 7, a criterion of 'gini', and a 'max_depth' value of 6. In our evaluation, it becomes evident that the model struggles with defect detection when no data imbalance methods are applied; specifically, it fails to correctly predict true negatives. However, when data imbalance techniques such as SMOTE, ROS+RUS, and ROS are implemented, the RF model performs commendably. The F1-scores across these techniques are remarkably similar, indicating robust performance in identifying healthy components. Most notably, the model excels in detecting truly defective pieces when Random Oversampling is utilized, achieving a Specificity score of 0.96 on the training set and 0.65 on the test

set.

Table 4.2: Random Forest Performance with Handling Imbalance Techniques

Imbalance Techniques	**Feature Selection**		Data	**Metrics**				
	Model	Feature Number		Accuracy	Precision	Recall	F1-score	Specificity
None	None	-	Train	0.97	0.97	1.00	0.99	0.33
			Test	0.96	0.96	0.99	0.98	0.08
ROS	None	-	Train	0.88	0.86	0.93	0.90	0.84
			Test	0.92	0.98	0.93	**0.96**	**0.65**
SMOTE (KN = 30)	None	-	Train	0.93	0.91	0.96	0.94	0.90
			Test	0.94	0.97	0.96	0.97	0.44
SMOTE (KN = 5)	None	-	Train	0.93	0.91	0.96	0.94	0.90
			Test	0.94	0.97	0.96	0.97	0.45
ROS+RUS	None	-	Train	0.89	0.86	0.94	0.90	0.84
			Test	0.93	0.98	0.94	**0.96**	**0.63**
ROS+RUS	LR+GB	60	Train	0.89	0.86	0.95	0.90	0.83
			Test	0.93	0.98	0.95	**0.97**	**0.60**

Likewise, Table 4.3 exhibits the Decision Tree model's effectiveness in predicting the "OperatorLabel" across the training and test sets. The model was optimized using a feature set of 60, chosen through a combination of Logistic Regression and Gradient Boosting models in the feature selection procedure. The top performance was realized with a "min_samples_split" parameter of 30, a "gini" criterion, and a "max_depth" value of 10. Like the RF model, the Decision Tree model also benefits from data imbalance methods. Specifically, using ROS+RUS as the imbalance handling technique yields the best outcomes, achieving an F1-score and Specificity of 0.96 and 0.62 on the test data, respectively. This underscores ROS+RUS's effectiveness in data balancing, optimizing the Decision Tree model comparably. In contrast, although using ROS+RUS, along with feature selection methods, showed slightly better results and avoided overfitting, Decision Tree model along with none of the imbalance techniques managed to dramatically improve Specificity, which generally stayed below 0.7. In conclusion, while imbalance techniques can improve most performance metrics, challenges remain in increasing Specificity, warranting further investigation for optimizing classifier performance.

The Extra Tree Classifier is another predictive model we examined, with detailed results provided in Table 4.4. When no methods are used to handle imbalanced data, the model's Specificity score is notably low at 0.35, suggesting overfitting, even though other metrics seem strong. Data

Table 4.3: Decision Tree Performance with Handling Imbalance Techniques

Imbalance Techniques	Feature Selection		Hyperparameters			Data	Metrics					
	Model	Feature Number	min_samples_split	criterion	max_depth	Data	Accuracy	Precision	Recall	F1-score	Specificity	MCC
None	None	-	Default			Train	0.99	0.99	0.99	0.99	0.74	0.79
						Test	0.96	0.97	0.99	0.98	0.29	0.44
ROS	None	-	30	gini	10	Train	0.92	0.91	0.95	0.93	0.88	0.84
						Test	0.94	0.97	0.97	0.97	0.47	0.62
SMOTE (KN=5)	None	-	30	gini	10	Train	0.97	0.97	0.99	0.98	0.96	0.94
						Test	0.96	0.97	0.98	0.98	0.44	0.59
ROS+RUS	None	-	4	gini	5	Train	0.92	0.9	0.96	0.93	0.88	0.84
						Test	0.93	0.98	0.96	0.97	0.49	0.62
ROS+RUS	None	-	4	entropy	5	Train	0.88	0.84	0.96	0.90	0.80	0.77
						Test	0.93	0.97	0.96	0.96	0.44	0.59
ROS+RUS	None	-	4	gini	5	Train	0.88	0.87	0.93	0.90	0.82	0.75
						Test	0.92	0.98	0.94	**0.96**	**0.60**	**0.69**
ROS+RUS	DT	80	30	gini	10	Train	0.93	0.91	0.96	0.93	0.89	0.85
						Test	0.94	0.98	0.96	0.97	0.51	0.64
ROS+RUS	LR+GB	60	30	gini	10	Train	0.88	0.86	0.93	0.89	0.83	0.73
						Test	0.92	0.98	0.93	**0.96**	**0.62**	**0.69**

balancing techniques like ROS and SMOTE yield slightly lower performance metrics but are relatively similar to each other. A combination of ROS and RUS delivers a more balanced set of metrics, especially when paired with feature selection techniques like Logistic Regression or a hybrid of Logistic Regression and Principal Component Analysis (LR+PCT). For the highest Specificity rate of 0.82 on the test dataset, this combination is most effective when 60 features are selected using the LR+PCT method. Besides, the balanced performance across all metrics, ROS+RUS with Logistic Regression and 60 selected features provides the best overall results, achieving an F1-score of 0.94 and a Specificity score of 0.77. The best hyperparameters for this setup include a minimum sample split of 4, using ”entropy” as the criterion, a maximum depth of 6, and setting maximum features to 5; these parameters yielded the highest balanced metrics in our tests.

Lastly, in Table 4.5 the performance of TabNet model by implementation of different feature selection and data imbalance techniques is provided. The results clearly indicate that combining ROS with RUS yields relatively consistent and superior performance across metrics. Specifically, this combination outperforms when feature selection by Logistic Regression is implemented. Furthermore, the best identified hyperparameters for this result is ”n_d=8’, ”n_a=8”, and ”n_steps=7” with a ”batch_size=1024”, 60 features selected. In this scenario, the model achieved the highest specificity of 0.66 while maintaining a competitive F1-score of 0.96 on the test data.

Table 4.4: Extra Tree Classifier Performance with Handling Imbalance Techniques

Imbalance Techniques	**Feature Selection**		**Hyperparameters**					**Data**	**Metrics**					
	Model	Feature Number	min_samples_split	criterion	max_depth	num_features	max_features		Accuracy	Precision	Recall	F1-score	Specificity	MCC
None	None	-	Default					Train	0.98	0.99	0.99	0.99	0.83	0.87
								Test	0.96	0.97	0.99	0.98	0.35	0.52
ROS	None	-	4	entropy	6	90	5	Train	0.84	0.84	0.87	0.85	0.81	0.69
								Test	0.87	0.98	0.88	0.93	0.70	0.68
SMOTE (KN=5)	None	-	4	entropy	6	90	5	Train	0.84	0.84	0.86	0.85	0.82	0.69
								Test	0.87	0.98	0.87	0.93	0.69	0.68
SMOTE (KN= 40)	None	-	4	entropy	6	90	5	Train	0.85	0.84	0.89	0.87	0.81	0.70
								Test	0.89	0.98	0.90	0.94	0.67	0.71
ROS+RUS	None	-	4	gini	6	150	5	Train	0.85	0.83	0.91	0.87	0.80	0.72
								Test	0.89	0.98	0.90	0.94	0.67	0.71
ROS+RUS	RF	70	4	entropy	6	90	5	Train	0.85	0.83	0.90	0.86	0.80	0.71
								Test	0.91	0.98	0.92	0.95	0.66	0.71
ROS+RUS	LR	60	4	entropy	6	90	5	Train	0.87	0.84	0.93	0.88	0.80	0.74
								Test	0.88	0.99	0.89	**0.94**	**0.77**	**0.72**
ROS+RUS	LR+GB	60	4	entropy	6	90	5	Train	0.86	0.84	0.91	0.87	0.80	0.72
								Test	0.90	0.98	0.91	0.94	0.63	0.70
ROS+RUS	LR+PCT	60	4	entropy	6	30	5	Train	0.85	0.86	0.85	0.85	0.84	0.74
								Test	0.86	0.99	0.87	**0.92**	**0.82**	**0.79**

In a comparative analysis of the four ML and DL models—Random Forest, Decision Tree, Extra Tree Classifier, and TabNet—each demonstrated distinct strengths and weaknesses in this binary classification task. Utilizing data imbalance techniques universally improved model performance, albeit to varying degrees. Random Forest was notably proficient in identifying defects but faltered in the absence of data-balancing methods. The Decision Tree model displayed robust performance yet faced constraints in elevating its Specificity scores. Extra Tree Classifier outperformed the rest by achieving the highest Specificity rate of 0.82, particularly when optimal feature selection and data balancing techniques were employed. TabNet exhibited consistent, superior metrics and boasted a competitive F1 score. However, a common limitation across all models was the challenge of enhancing Specificity, indicating an avenue for future optimization. Among these, Extra Tree Classifier and TabNet emerged as the most balanced regarding F1-score and Specificity.

While TabNet is tailored for tabular data and exhibits a strong F1-score, it does not necessarily outperform the Extra Tree Classifier in terms of Specificity for various reasons. These could include more effective hyperparameter tuning and feature selection in the Extra Tree Classifier, as well as the particular success of data imbalance techniques with this model. Additionally, the Extra Tree Classifier's simpler structure may lend itself better to the specific dataset or classification task,

Table 4.5: TabNet Performance with Handling Imbalance Techniques

Imbalance Techniques	Feature Selection		Hyperparameters				Data	Metrics					
	Model	Feature Number	n_d	n_a	n_steps	batch_ size		Accuracy	Percision	Recall	F1-score	Specificity	MCC
None	None	-	Default				Train	0.98	0.98	0.99	0.99	0.54	0.72
							Test	0.96	0.96	0.99	0.98	0.16	0.37
ROS	None	-	Default				Train	0.95	0.94	0.97	0.95	0.92	0.90
							Test	0.94	0.97	0.96	0.97	0.48	0.62
SMOTE (KN=30)	None	-	Default				Train	0.93	0.91	0.97	0.94	0.88	0.86
							Test	0.94	0.98	0.96	0.97	0.55	**0.72**
ROS+RUS	None	-	16	16	4	1024	Train	0.93	0.91	0.96	0.94	0.90	0.86
							Test	0.92	0.97	0.95	0.96	0.49	0.62
ROS+RUS	LR	60	40	20	6	128	Train	0.93	0.92	0.96	0.94	0.91	0.87
							Test	0.93	0.98	0.95	**0.96**	**0.61**	**0.69**
ROS+RUS	LR	60	8	8	7	1024	Train	0.92	0.90	0.95	0.92	0.88	0.84
							Test	0.93	0.98	0.95	**0.96**	**0.66**	**0.73**
ROS+RUS	LR+RF	60	8	8	7	512	Train	0.91	0.90	0.94	0.92	0.88	0.83
							Test	0.92	0.98	0.93	0.95	0.56	0.67
ROS+RUS	PCT+LR	100/60	16	16	7	512	Train	0.93	0.91	0.96	0.94	0.90	0.86
							Test	0.94	0.97	0.96	0.97	0.48	0.62

whereas TabNet's complexity could require more fine-tuning to reach optimal performance. Both models are sensitive to the nature of the data and the metrics used for evaluation, and in this instance, Extra Tree Classifier appears to offer a better balance of F1-score and Specificity.

4.4.3 Evaluating Our Approaches Against Current Solutions

As we navigate the complexities of ML and DL approaches in PCB manufacturing, it becomes crucial to understand how our methods compare to existing research. This section serves as a comparative analysis, designed to show how our work fits into the bigger picture of similar research, and We examine key similarities and differences to highlight the unique value of our approach. Table 4.6 provides a comparative summary with four related studies. Even though the best training performance in other studies achieved an F1-score of 0.81, their low test set results indicate that their models are not properly trained. In contrast, our model is well-trained, as evidenced by an F1-score of 0.96 for the train set. Our method, which employs TabNet combined with a hybrid method strategy for handling data imbalance (ROS+RUS) and utilizes feature selection with an LR model, significantly outperforms existing methods. For instance, the highest F1-score reported in other studies is 0.67 for the test set, while we achieved an F1-score of 0.92, demonstrating a substantial advantage. Moreover, unlike other studies that have largely ignored the value of predicting True

Negatives, we emphasize this aspect. Our approach not only considers Specificity but also shows a Specificity of 0.82 when using the Extra Tree Classifier model in conjunction with the ROS + RUS hybrid method and LR+PCT for feature selection.

Table 4.6: F1-score Comparison for OperatorLabel Prediction

Comparison Cases	**Train**	**Test**
Our Method (TabNet Result)	0.96	0.92
Case 1 (Gaffet et al., 2022)	0.66	0.67
Case 2 (Taco et al., 2022)	0.78	0.62
Case 3 (Tang et al., 2022)	0.68	0.54
Case 4 (Schmidt et al., 2022)	0.81	0.38

4.5 Summary of The Chapter

This chapter shifts its focus to quality control in PCB manufacturing after component placement, specifically employing ML and DL methods for defect detection during the AOI stage. The chapter aims to enhance defect detection capabilities by comparing the proficiency of ML and DL models with human operators. While human operators add valuable insights to the defect identification process, ML and DL models offer the benefits of speed, consistency, and reduced human error. The chapter also outlines the data preparation steps and model implementations. Various ML and DL models, such as Random Forest, Decision Tree, Extra Tree Classifier, and TabNet are introduced and evaluated for their effectiveness in defect detection.

Furthermore, this chapter delves into the critical issue of imbalanced data, which can compromise the accuracy of ML models' predictions in the context of PCB manufacturing. In this regard, various techniques to address this imbalance, ranging from data-level approaches to algorithm-level methods, are explored. Upon evaluation, the Extra Tree Classifier and TabNet models emerge as leaders, demonstrating a well-rounded performance in key metrics like F1-score and Specificity.

Despite these advances, enhancing Specificity continues to be a universal challenge across all considered models. At last, the chapter wraps up by underscoring that while ML and DL models bring advantages like speed and efficiency, selecting the appropriate model and approach is crucial, given the unique challenges and objectives in PCB manufacturing.

Chapter 5

Categorizing Repair Labels in PCBs through Multi-Classification

In the previous chapter, we conducted an in-depth analysis of various ML and DL algorithms, focusing on their efficacy in predicting the health status of PCBs after the component replacement stage. This analysis served as a keystone for determining whether these PCBs are fit for progression to subsequent manufacturing stages. Therefore, based on the operator's evaluation, PCBs deemed in good condition advance to the next production phase, whereas those identified as faulty are either discarded or set aside for potential repair in future stages. Building upon this foundation, this chapter aims to take the next logical step in optimizing PCB manufacturing. We extend our earlier work on automated defect detection in PCBs to explore the feasibility of automatically generating repair labels using data analysis along with ML and DL models. As previously noted, PCBs marked as "Bad" at the "OperatorLabel" stage proceed to a subsequent inspection phase where their potential for repair is assessed. Currently, this evaluation is conducted manually by a trained operator, a method that could be more cost-effective and consistently accurate. Consequently, our objective in this chapter is to investigate the automation of this labeling process to determine if "RepairLabels" can be accurately assigned based on historical data without human intervention.

An automatic repair label generator using ML can significantly enhance the efficiency and quality of a PCB manufacturing line. By automating the label classification, the system not only speeds

up the production process but also ensures a consistent and precise evaluation of each board. This reduces labor costs and minimizes material waste by accurately identifying boards that are either good to go, need repair, or should be discarded. In addition, the data-driven nature of machine learning provides valuable insights for continuous improvement and ensures rigid compliance with manufacturing standards. Overall, the technology offers a cost-effective, accurate, and efficient solution for managing quality control in PCB manufacturing.

In the following sections of this chapter, the content is organized as follows: Section 5.1 introduces the problem under investigation and elaborates on the key topics that will be covered. Section 5.2 details the dataset used for this study, along with the steps undertaken for its preprocessing and preparation. Subsequently, Section 5.3 discusses the ML and DL models employed for the multi-class classification of ”RepairLabel.” Lastly, Section 5.4 provides the concluding remarks, summarizing the chapter's key findings and implications.

5.1 Problem Identification

Choosing to repair a PCB rather than dispose of it offers numerous advantages from both economic and environmental perspectives. Firstly, repairing a PCB can be a cost-effective solution, while reproducing a new PCB can be significantly more expensive, especially if the fault is minor and can be easily fixed. Moreover, repairing a PCB helps reduce electronic waste and its associated environmental impact. Discarding PCBs contributes to the growing problem of electronic waste, which often ends up in landfills and can release hazardous substances into the environment. As a result, the accurate assignment of the ”RepairLabel” not only maximizes resource utilization but also contributes to sustainable practices. As a result, the accurate assignment of correct repair labels not only maximizes resource utilization but also contributes to sustainable practices. However, accurately identifying which PCBs should be repaired is a challenging task that relies heavily on expert evaluation.

To address this challenge, we focus on predicting repair label in this chapter. As shown in Figure 1.1, after inspecting PCBs and assigning labels indicating their health status following the AOI stage, PCBs with faulty components are directed to a specialized section where an expert

operator examines them and determines whether they are repairable. Due to the complexity and delicacy of PCBs, this task is challenging and prone to errors, relying heavily on the operator's expertise and evaluation. To improve this process, our goal in this chapter is to introduce a model that can accurately predict a PCB's "RepairLabel." Utilizing ML and DL techniques, the model will employ SPI features and AOI labels as predictive variables, aiming to prevent the need for human oversight in this critical assessment.

In order to attain the objective of creating a model that can replace the human inspector, it must have the capability to predict the repair needs of the PCBs and their corresponding label classes. Distinguishing between these specific repair label categories adds complexity to the task, as the model must discern the differences that dictate whether a PCB is irreparable, yet to be classified, or falsely deemed as scrap. The dataset provided for this study includes four distinct categories under the "RepairLabel" feature: NA (Not Available), NotYetClassified, NotPossibleToRepair, and FalseScrap. These categories will be predicted using developed ML and DL models.

5.2 Data Preprocessing

The dataset used in this chapter is the same as the data aggregated and prepared in Chapter 4 (Section 4.2), but with some variations. Firstly, an additional feature called "OperatorLabel" has been included, which will be utilized for model training. This feature, which previously was a target for prediction in the past chapter, has now been added to the features group due to its impact on model decisions.

Secondly, the target column, "RepairLabel," comprises four classes, one of which is "Not Available." This class represents components labeled as "Good" in the prior stage, which are exempt from repair examination. Consequently, all rows with this label must be excluded from the dataset. This action significantly reduces data size, as it is the major class. After this reduction, the data stands at a total of 905 rows.

Thirdly, following the guidelines in Section 4.2.2, two distinct datasets are employed for training and testing. These datasets should have matching features after OneHotEncoding of categorical

features to ensure proper model functionality. It is essential to align all features meticulously. Notably, the ”AOILabel,” ”Result,” and ”ComponentID” columns exhibit differences between the train and test sets. To address this disparity, specific adjustments are necessary to align these columns and prepare both sets for input into ML and DL models. One approach could be removing unmatched entries in the train and test sets. Specifically, for the train set, the AOILabel and Result columns should undergo the removal of specific entries, including Broken, E.Bridging, E.Position, and W.Position. Moreover, the ComponentID column in the train set excludes entries such as C12, C16, C23, C24, C3, CN1, L3, R13, R15, R40, R41, R42, R43, R46, R52, RA1, RA3, TR1, TR2, TR4, RA1, TRB7, and U1. Similarly, the test set requires the exclusion of C19, C20, C28, C29, C32, C35, C4, C8, DZ2, R16, R17, R22, R23, R26, R28, R29, R3, R32, R37, R4, R48, R49, R51, R8, RA2, TRB5, TRB6, and U5 from the ComponentID column. However, this would result in losing approximately 19 percent of the training data and 23 percent of the test data, which is not ideal given our already small dataset. Consequently, we decided to go with an alternative: using the handle_unknown='ignore' option during OneHotEncoding. This ensures that if a category is present in the training set but missing in the test set (or vice versa), a corresponding column will still be created in the one-hot encoded output, filled with zeros. By adopting this approach, we not only keep a consistent number of features between the training and test datasets but also preserve most of the original data.

Moving forward, this task's subsequent steps for data preparation closely align with those outlined in the previous chapter (see Section 4.2). After completing data encoding, we will proceed with data cleaning, duplicate removal, feature scaling, and index resetting as the next phases of our data preparation process.

5.3 Model Implementation and Experimental Results

As highlighted in earlier chapters, one of the critical factors aiding us in accurately interpreting model performance is the selection of appropriate metrics. Choosing the right metrics provides a clearer understanding of how well the model is functioning and guides us in fine-tuning it for optimal results. Similarly to Chapter 3 and 4, we first talk about the specific metrics that are practical

in multi-class problems and then go through the details of the results and compare our method with similar cases.

Additionally, it is important to highlight that the encouraging outcomes from Chapter 4 motivated us to stick with similar analytical models. Specifically, We employed Random Forest, Decision Tree, Extra Tree Classifier, TabNet, and also added Gradient Boosting to our toolkit. These models were fine-tuned using feature selection methods and data imbalance solutions discussed in previous chapters. Comparative results for these models can be found in Section 5.3.2.

5.3.1 Metrics

In a multi-class classification problem, metrics including precision, recall, F1-score, and others can be calculated in a manner slightly more complex than binary classification. Below are some methods to compute these metrics:

- **Precision, Recall, and F1-Score:** For multi-class problems, these metrics can be calculated for each class individually by considering it as the **positive** class and aggregating all other classes as the **negative** class. Afterward, these per-class metrics can be averaged in various ways:

  (1) **Micro-Averaging**: Sum up individual true positives, false positives, and false negatives for all classes, and then calculate precision, recall, and F1-score. This approach gives equal weight to each observation:

  $$\text{Micro-Precision} = \frac{\Sigma\text{TP}}{\Sigma\text{TP} + \Sigma\text{FP}}.$$

  (2) **Macro-Averaging**: Calculate precision, recall, and F1-score for each class individually and then take the average, treating all classes equally:

  $$\text{Macro-Precision} = \frac{1}{N}\Sigma(\text{Precision of Class}_i).$$

  (3) **Weighted Averaging**: Similar to Macro-Averaging but when averaging, each metric is weighted by the number of instances in each class. This is useful when class imbalance is present:

$$\text{Weighted-Precision} = \Sigma \left(\text{Weight of Class}_i \times \text{Precision of Class}_i \right).$$

- **Specificity:** Specificity measures the true negative rate for each class, treating it as the **positive** class and all others as **negative**. Similar to precision, recall, and F1-score, specificity can be averaged across all classes for a comprehensive view.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}.$$

In light of our methods for calculating metrics, Weighted Averaging emerges as the most suitable approach for our imbalanced dataset. This technique emphasizes the minority class, effectively reducing any bias toward the majority class. As a result, the performance metrics better represent the model's ability to accurately classify all classes, not just the more prevalent ones. In the following section, we present our results using this Weighted Averaging method to calculate key metrics such as Precision, Recall, F1-score, and Specificity, which are all averaged accordingly.

5.3.2 Results

Table 5.1 provides an in-depth look at how various ML models perform in multi-class classification tasks, specifically after applying feature selection. The table is sorted by the "Implemented Model" and highlights two main factors: the model used and the number of features selected for the "Feature Selection" approach. It is essential to note that these models have been trained under different settings, and **GridSearch** was used to identify optimal parameters. The table evaluates key performance indicators, including Precision, Recall, F1-score, and Specificity for each model. Notably, the Decision Tree model using the ET method and 50 selected features achieved the highest F1-score of 0.70. This result comes from a specific configuration— criterion set to 'gini', max_depth at 7, min_samples_split at 2, and splitter set to 'random'—indicating this setup offers an ideal balance between Precision and Recall, making it potentially the most effective choice for this multi-class classification task. Further, the table segregates the results based on different types of feature selection techniques, such as LR, ET, PCT, LR+PCT, and LR+ET. It is clear that the number

of features used for training also varies, indicating that the feature set size could be a crucial factor in model performance. Although the F1-scores are relatively close across the board, the Specificity values show a broader range, from as low as 0.40 in Gradient Boosting under LR to as high as 0.60 in Decision Tree under ET with 50 features. This variance in Specificity indicates that while some models may be excellent in terms of F1-score, they may not be as effective when it comes to correctly identifying negative cases.

Table 5.1: Multi-Classification with Feature Selection

Implemented Model	**Feature Selection**		**Metrics**			
	Model	Featur Number	Percision	Recall	F1-score	Specificity
Decision Tree	LR	40	0.61	0.70	0.65	0.50
Extra Tree			0.62	0.74	**0.68**	0.55
Gradient Boosting			0.57	0.67	0.61	0.40
Random Forest			0.60	0.73	0.66	0.49
Decision Tree	ET	50	0.66	0.74	**0.70**	0.60
Extra Tree			0.67	0.73	0.69	0.59
Gradient Boosting			0.62	0.72	0.66	0.46
Random Forest			0.64	0.76	0.69	0.51
Decision Tree	PCT	30	0.63	0.71	0.67	0.57
Extra Tree			0.64	0.70	**0.67**	0.56
Gradient Boosting			0.59	0.69	0.63	0.43
Random Forest			0.61	0.73	0.66	0.48
Decision Tree	LR+PCT	50/25	0.66	0.71	**0.67**	0.57
Extra Tree			0.57	0.69	0.62	0.44
Gradient Boosting			0.62	0.74	0.67	0.53
Random Forest			0.63	0.71	0.64	0.44
Decision Tree	LR+ET	40/20	0.65	0.70	**0.66**	0.56
Extra Tree			0.57	0.67	0.60	0.43
Gradient Boosting			0.61	0.72	0.65	0.52
Random Forest			0.63	0.70	0.63	0.43

Table 5.2 displays the top-performing results for four machine learning models, extending the analysis presented in Table 5.1. The data reveals that performance metrics differ based on the mix of models, feature selection methods, and data imbalance solutions used. Like before, we employed GridSearch to pinpoint the best parameters for each scenario. Notably, the Extra Tree model, when paired with LR+PCT for feature selection with 40 features and SMOTE (KN=7) for balancing data, yielded the most consistent metrics: a precision of 0.66, a recall of 0.73, an F1-score of 0.71, and a specificity of 0.65. These metrics were achieved using tuned parameters—criterion set to 'gini,'

max_depth of 30, min_samples_split of 4, n_estimators: of 20. This indicates that while there is no universal solution for optimizing performance on imbalanced, multi-class datasets, some strategies are more effective than others. Hence, applying hybrid methods for feature selection in combination with data imbalance techniques can improve our results, bringing us one step closer to automating the repair label prediction process.

Table 5.2: Model Performance by Implementing Feature Selection and Data Imbalance Techniques for Multi-class Classification

Implemented Model	**Feature Selection**		**Imbalance Techniques**		**Metrics**			
	Model	Feature Number	Method	Sampling_Strategy	Percision	Recall	F1-score	Specificity
Decision Tree	LR	20	SMOTE (KN=5)	Auto	0.63	0.69	**0.65**	**0.62**
Extra Tree					0.61	0.69	0.65	0.60
Gradient Boosting					0.61	0.66	0.63	0.63
Random Forest					0.61	0.69	0.65	0.58
Decision Tree	LR+PCT	40	SMOTE (KN=7)	Minority	0.68	0.74	0.70	0.67
Extra Tree					0.66	0.73	**0.71**	**0.65**
Gradient Boosting					0.64	0.7	0.67	0.67
Random Forest					0.66	0.73	0.69	0.62
Decision Tree	LR+ET	30	SMOTE (KN=9)	Dictionary	0.58	0.65	0.60	0.43
Extra Tree					0.61	0.66	0.63	0.46
Gradient Boosting					0.56	0.62	0.59	0.42
Random Forest					0.60	0.63	**0.64**	**0.44**

Table 5.3 presents the performance of the TabNet model in predicting the ”RepairLabel” when various data imbalance techniques and feature selection methods with combination of class weighting are applied. TabNet outperforms other models in predicting True Negatives, evidenced by its higher Specificity score of 0.69. This indicates TabNet’s effectiveness in handling highly imbalanced data. While the F1-score for Decision Tree and Extra Tree Classifier models appears higher in Tables 5.1 and 5.2, TabNet surpasses them in Specificity. This demonstrates TabNet’s superior ability to accurately identify ”FalseScrap” components, which are non-defective items that should return to the production line. This is particularly important for us as it helps avoid unnecessary costs associated with repairing boards that are already in good condition.

In conclusion, the Extra Tree model consistently shows strong performance in multi-class classification, especially when combined with certain feature selection techniques like LR and LR+PCT. The analysis also reveals that while F1-score is a reliable metric for general performance, Specificity varies more widely among models and is higher when TabNet is employed. This indicates

Table 5.3: TabNet Performance for Multi-class Classification

Feature Selection		**Imbalance Techniques**			**Metrics**			
Model	Feature Number	Method	Sampling_ Strategy	Class Weight	Percision	Recall	F1-score	Specificity
None		None		-	0.52	0.68	0.58	0.33
LR+PCT	50	None		-	0.57	0.64	0.60	0.45
LR+PCT	50	None		Done	0.58	0.57	0.56	0.61
LR+PCT	45	SMOTE (KN=7)	Auto	Done	0.60	0.66	0.63	0.59
LR+ETC	45	SMOTE (KN=5)	Minority	Done	0.57	0.55	0.54	0.58
LR+ET	45	SMOTE (KN=7)	{0: 6200, 1: 4000, 2: 3500}	Done	0.60	0.61	0.60	**0.64**
LR+ET	45	SMOTE (KN=7)	{0: 5520, 1: 4904, 2: 4880}	Done	0.63	0.62	0.68	**0.69**

that model selection should consider the specific requirements of the task, including balancing different performance metrics. These insights highlight the value of using hybrid methods for feature selection and data imbalance to improve model performance in repair label prediction.

5.3.3 Comparison To Similar Work

In evaluating our model's ability to predict repair labels, a direct comparison with other studies is challenging due to significant methodological differences. As illustrated in Table 5.4, our results closely align with those of Case 3 and Case 4 but lag behind Case 1 and Case 2. The disparity in F1-scores arises largely from a crucial divergence in our approach compared to that of Case 1 and Case 2. These studies opted to omit the "NotYetClassified" label, focusing instead on a binary classification between "FalseScrap" and "NotPossibleToRepair", which is ignoring the reality of the dataset. This simplification naturally makes the problem easier to solve, explaining their higher performance metrics. In contrast, we have chosen to incorporate the "NotYetClassified" label, as it captures the nuances of components requiring further analysis, making our task inherently more complex. Furthermore, existing studies have generally overlooked the issue of data imbalance and

have often neglected to select metrics that most accurately reflect the performance of the model. For instance, the metric of Specificity, crucial for indicating how well the model identifies ”FalseScrap” components, is often absent in other studies. In contrast, our research reports a Specificity of approximately 0.69 using the TabNet model, combined with feature selection and data imbalance techniques.

Table 5.4: F1-score Comparison for RepairLabel Prediction

Comparison Cases	**Test**
Our Method (TabNet Result)	0.71
Case 1 (Gaffet et al., 2022)	0.77
Case 2 (Taco et al., 2022)	0.90
Case 3 (Tang et al., 2022)	0.71
Case 4 (Schmidt et al., 2022)	0.70

5.4 Summary of The Chapter

This chapter delved into optimizing the quality control process in PCB manufacturing through the automation of repair labels. Building on previous analyses that evaluated PCB health post-component replacement, the current focus was on automating the ”RepairLabel” assignment currently carried out manually. We introduced ML and DL models capable of predicting these labels with high accuracy, potentially making human oversight redundant. The chapter also elaborated on the dataset used, highlighting specific preprocessing steps and challenges related to feature alignment across training and test sets.

Performance metrics from various ML and DL models showed that the Extra Tree Classifier and TabNet models, when combined with specific feature selection techniques, produced the most balanced results. This underscores the need for a nuanced approach in model selection and configuration, considering both the F1-score and Specificity metrics. Overall, automating the repair label assignment not only speeds up the production process but also contributes to cost-effectiveness and sustainable manufacturing practices.

Chapter 6

Summary and Future Research Directions

This comprehensive chapter serves as a synthesized overview of the extensive research and methodologies outlined in the preceding chapters, all aimed at revolutionizing defect detection and quality control in PCB manufacturing through ML and DL technologies.

The journey begins with the first chapter, which aims to enhance AOI in PCB manufacturing. It emphasized the significance of using advanced data processing techniques like one-hot encoding, feature selection, particularly, the handling of imbalanced datasets. This chapter demonstrated the effectiveness of ML and DL models, especially the Random Forest and Extra Tree classifiers, in defect detection compared to other techniques. A unique contribution here was the novel approach to merging two distinct datasets using a common identifier, which provided a more comprehensive view for analysis.

The second chapter continued in this vein but focused on quality control post-component placement. It assessed the relative merits of ML/DL models against human operators in AOI defect detection. While human input undoubtedly adds value, ML/DL models scored higher on metrics like speed, consistency, and reduced error rates. Nevertheless, dealing with imbalanced data surfaced as a common challenge, which the chapter attempted to address through various techniques.

Here, the Extra Tree Classifier and TabNet models stood out, excelling in crucial metrics like the F1-score and Specificity. Nevertheless, enhancing Specificity remained a challenging objective across all models.

The third chapter broadened the scope to include automation in the repair label assignment process, a function traditionally performed manually. The chapter detailed the careful selection and preprocessing of data sets for this task. Performance evaluation showed that the Extra Tree model, coupled with precise feature selection techniques, was best suited for balancing performance metrics, thus underlining the importance of nuanced model selection.

Across all chapters, a recurrent theme was the criticality of dealing with imbalanced data. The research showed that techniques such as oversampling and specialized feature selection are essential for optimizing model performance. Another common thread was the comparative advantage of the Extra Tree Classifier in various applications, indicating its robustness as a versatile ML model.

Finally, the broader implications of this research extend beyond just improved defect detection. They also highlight the immense potential for cost-saving and sustainability through automated, efficient, and highly accurate systems. Thus, while ML and DL technologies offer transformative benefits, the key to unlocking their full potential lies in the judicious selection of models and techniques tailored to specific challenges and objectives in PCB manufacturing.

6.1 Summary of Thesis Contribution

In this research, we make several key contributions to improve the efficiency, quality, and sustainability of PCB manufacturing through advanced ML and DL models. First, we address the challenge of imbalanced data in PCB defect detection by offering a wide range of solutions that extend from data-level to algorithm-level methods. We also explore alternative resampling methods like SMOTE and ensemble techniques, aiming for unbiased recognition of minority classes. Second, we conduct an in-depth analysis of TabNet, a DL model for tabular data, demonstrating its superior accuracy, interpretability, and computational efficiency. Lastly, we introduce a novel hybrid approach for feature selection that enhances model performance and can be applied to various data types. Overall, our work advances both the field of machine learning and the PCB manufacturing industry.

6.2 Future Research

Future research in the realm of PCB manufacturing with ML and DL has several promising avenues. One critical focus area should be improving the Specificity metric, which has remained a consistent challenge across all models. This could be coupled with the exploration of hybrid methods that combine the strengths of ML and DL to enhance predictive accuracy. The persistent issue of imbalanced data also warrants deeper investigation, possibly through more sophisticated oversampling and undersampling techniques.

Another promising avenue would be to explore alternative techniques for managing imbalanced data. Generative Adversarial Networks (GANs) could be employed as a data augmentation method to improve model performance on minority classes. Additionally, while this study focused on the TabNet model, there are other specialized models like TransTab that are tailored for handling intricate tabular data and could prove beneficial in automating defect detection in PCBs.

A further compelling topic for future research would be the application of Multi-task learning in PCB defect detection. Rather than sequentially predicting labels for each manufacturing stage, a Multi-task learning approach could enable simultaneous label prediction across multiple stages. This could not only streamline the defect detection process but also improve the system's efficiency and accuracy. Such advancements would be crucial in enhancing the overall quality of PCB manufacturing.

References

Adibhatla, V. A., Chih, H.-C., Hsu, C.-C., Cheng, J., Abbod, M. F., & Shieh, J.-S. (2021). Applying deep learning to defect detection in printed circuit boards via a newest model of you-only-look-once.

Albee, A. J. (2013). The evolution of ict: Pcb technologies, test philosophies, and manufacturing business models are driving in-circuit test evolution and innovations. In *Ipc apex expo conference and exhibition* (Vol. 1, pp. 381–401).

Anoop, K., Sarath, N., & Kumar, V. (2015). A review of pcb defect detection using image processing. *Intern. J. Eng. Innov. Technol*, *4*, 188–192.

Arik, S. Ö., & Pfister, T. (2021). Tabnet: Attentive interpretable tabular learning. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 35, pp. 6679–6687).

Bi, J., & Zhang, C. (2018). An empirical comparison on state-of-the-art multi-class imbalance learning algorithms and a new diversified ensemble learning scheme. *Knowledge-Based Systems*, *158*, 81–93.

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5–32.

Chaudhary, V., Dave, I. R., & Upla, K. P. (2017). Automatic visual inspection of printed circuit board for defect detection and classification. In *2017 international conference on wireless communications, signal processing and networking (wispnet)* (pp. 732–737).

Chauhan, A. P. S., & Bhardwaj, S. C. (2011). Detection of bare pcb defects by image subtraction method using machine vision. In *Proceedings of the world congress on engineering* (Vol. 2, pp. 6–8).

Chen, J., Zhang, Z., & Wu, F. (2021). A data-driven method for enhancing the image-based

automatic inspection of ic wire bonding defects. *International journal of production research*, *59*(16), 4779–4793.

Chicco, D., & Jurman, G. (2023). The matthews correlation coefficient (mcc) should replace the roc auc as the standard metric for assessing binary classification. *BioData Mining*, *16*(1), 1–23.

Choi, D., Kim, Y., Kim, J., & Kim, H. (2018). Advanced non-destructive fault isolation techniques for pcb substrates using magnetic current imaging and terahertz time domain reflectometry. In *Istfa 2018* (pp. 43–46).

Chun, D., Ang, S. S., Chai, T. C., & Tay, A. (2003). A time-domain-reflectometry characterization technique for packaging substrates. In *Proceedings of the 5th electronics packaging technology conference (eptc 2003)* (pp. 361–365).

Comparing different preprocessing techniques available in scikit-learn. (2007). Retrieved from https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#sphx-glr-auto-examples-preprocessing-plot-all-scaling-py (Accessed: 2023)

Cui, H., & Anderson, C. G. (2016). Literature review of hydrometallurgical recycling of printed circuit boards (pcbs). *J. Adv. Chem. Eng*, *6*(1), 142–153.

Dai, W., Mujeeb, A., Erdt, M., & Sourin, A. (2020). Soldering defect detection in automatic optical inspection. *Advanced Engineering Informatics*, *43*, 101004.

Dong, Z., & Chen, L. (2019). Image registration in pcb fault detection based on infrared thermal imaging. In *2019 chinese control conference (ccc)* (pp. 4819–4823).

Effrosynidis, D., & Arampatzis, A. (2021). An evaluation of feature selection methods for environmental data. *Ecological Informatics*, *61*, 101224.

Enshaei, N., Ahmad, S., & Naderkhani, F. (2020). Automated detection of textured-surface defects using unet-based semantic segmentation network. In *2020 ieee international conference on prognostics and health management (icphm)* (pp. 1–5).

Erickson, B. J., & Kitamura, F. (2021). *Magician's corner: 9. performance metrics for machine learning models* (Vol. 3) (No. 3). Radiological Society of North America.

Fernández, A., Garcia, S., Herrera, F., & Chawla, N. V. (2018). Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial*

intelligence research, *61*, 863–905.

Gaber, L., Hussein, A. I., & Moness, M. (2021). Fault detection based on deep learning for digital vlsi circuits. *Procedia Computer Science*, *194*, 122–131.

Gaffet, A., Roa, N. B., Ribot, P., Chanthery, E., & Merle, C. (2022). A hierarchical xgboost early detection method for quality and productivity improvement of electronics manufacturing systems. In *7th european conference of the prognostics and health management society 2022.*

Galetto, M., Verna, E., Genta, G., & Franceschini, F. (2020). Uncertainty evaluation in the prediction of defects and costs for quality inspection planning in low-volume productions. *The International Journal of Advanced Manufacturing Technology*, *108*, 3793–3805.

Gebus, S., Lorillard, S., & Juuso, E. (2004). *Defect localization on a pcb with functional testing*. University of Oulu.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, *63*, 3–42.

Hassanin, A.-A. I., Abd El-Samie, F. E., & El Banby, G. M. (2019). A real-time approach for automatic defect detection from pcbs based on surf features and morphological operations. *Multimedia Tools and Applications*, *78*, 34437–34457.

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, *13*(4), 18–28.

Huang, R., Gu, J., Sun, X., Hou, Y., & Uddin, S. (2019). A rapid recognition method for electronic components based on the improved yolo-v3 network. *Electronics*, *8*(8), 825.

Jin, J., Feng, W., Lei, Q., Gui, G., Li, X., Deng, Z., & Wang, W. (2021). Defect detection of printed circuit boards using efficientdet. In *2021 ieee 6th international conference on signal and image processing (icsip)* (pp. 287–293).

Jiuqing, W., & Xingshan, L. (2002). Pcb infrared thermal imaging diagnosis using support vector classifier. In *Proceedings of the 4th world congress on intelligent control and automation (cat. no. 02ex527)* (Vol. 4, pp. 2718–2722).

Joseph, L. P., Joseph, E. A., & Prasad, R. (2022). Explainable diabetes classification using hybrid bayesian-optimized tabnet architecture. *Computers in Biology and Medicine*, *151*, 106178.

Joseph, V. R. (2022). Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The*

ASA Data Science Journal, *15*(4), 531–538.

Jurj, S. L., Rotar, R., Opritoiu, F., & Vladutiu, M. (2020). Affordable flying probe-inspired in-circuit-tester for printed circuit boards evaluation with application in test engineering education. In *2020 ieee international conference on environment and electrical engineering and 2020 ieee industrial and commercial power systems europe (eeeic/i&cps europe)* (pp. 1–6).

Kaliyavaradhan, S. K., Prem, P. R., Ambily, P., & Mo, K. H. (2022). Effective utilization of e-waste plastics and glasses in construction products-a review and future research directions. *Resources, Conservation and Recycling*, *176*, 105936.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*.

Khalilian, S., Hallaj, Y., Balouchestani, A., Karshenas, H., & Mohammadi, A. (2020). Pcb defect detection using denoising convolutional autoencoders. In *2020 international conference on machine vision and image processing (mvip)* (pp. 1–5).

Kim, J., Ko, J., Choi, H., & Kim, H. (2021). Printed circuit board defect detection using deep learning via a skip-connected convolutional autoencoder. *Sensors*, *21*(15), 4968.

Kovtun, I., Boiko, J., & Petrashchuk, S. (2017). Nondestructive strength diagnostics of solder joints on printed circuit boards. In *2017 international conference on information and telecommunication technologies and radio electronics (ukrmico)* (pp. 1–4).

Kovtun, I., Boiko, J., & Petrashchuk, S. (2018). Acoustic emission application for nondestructive strength diagnostics of printed circuit boards.

Kovtun, I., Boiko, J., & Petrashchuk, S. (2019). Reliability improvement of printed circuit boards by designing methods for solder joint technical diagnostics with application of acoustic emission method.

Krieger, V., Wondrak, W., Dehbi, A., Bartel, W., Ousten, Y., & Levrier, B. (2006). Defect detection in multilayer ceramic capacitors. *Microelectronics Reliability*, *46*(9-11), 1926–1931.

Kumari, S., Singh, K., Khan, T., Ariffin, M. M., Mohan, S. K., Baleanu, D., & Ahmadian, A. (2023). A novel approach for continuous authentication of mobile users using reduce feature elimination (rfe): A machine learning approach. *Mobile Networks and Applications*, 1–15.

LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., ... others (1995). Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks* (Vol. 60, pp. 53–60).

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM computing surveys (CSUR)*, *50*(6), 1–45.

Lin, C.-H., Wang, S.-H., & Lin, C.-J. (2019). Using convolutional neural networks for character verification on integrated circuit components of printed circuit boards. *Applied Intelligence*, *49*, 4022–4032.

Ling, Q., & Isa, N. A. M. (2023). Printed circuit board defect detection methods based on image processing, machine learning and deep learning: A survey. *IEEE Access*.

Liu, L., Wu, X., Li, S., Li, Y., Tan, S., & Bai, Y. (2022). Solving the class imbalance problem using ensemble algorithm: application of screening for aortic dissection. *BMC Medical Informatics and Decision Making*, *22*(1), 1–16.

Ma, J. (2017). Defect detection and recognition of bare pcb based on computer vision. In *2017 36th chinese control conference (ccc)* (pp. 11023–11028).

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, *405*(2), 442–451.

Mazzoleni, M., Sarda, K., Acernese, A., Russo, L., Manfredi, L., Glielmo, L., & Del Vecchio, C. (2022). A fuzzy logic-based approach for fault diagnosis and condition monitoring of industry 4.0 manufacturing processes. *Engineering Applications of Artificial Intelligence*, *115*, 105317.

McDonnell, K., Murphy, F., Sheehan, B., Masello, L., & Castignani, G. (2023). Deep learning in insurance: Accuracy and model interpretability using tabnet. *Expert Systems with Applications*, *217*, 119543.

Mohammed, R., Rawashdeh, J., & Abdullah, M. (2020). Machine learning with oversampling and undersampling techniques: overview study and experimental results. In *2020 11th international conference on information and communication systems (icics)* (pp. 243–248).

Mourtzis, D., Angelopoulos, J., & Panopoulos, N. (2020). Intelligent predictive maintenance and remote monitoring framework for industrial equipment based on mixed reality. *Frontiers in*

Mechanical Engineering, *6*, 578379.

Mukherjee, M., & Khushi, M. (2021). Smote-enc: A novel smote-based method to generate synthetic data for nominal and continuous features. *Applied System Innovation*, *4*(1), 18.

Neubauer, C., & Hanke, R. (1993). Improving x-ray inspection of printed circuit boards by integration of neural network classifiers. In *Proceedings of 15th ieee/chmt international electronic manufacturing technology symposium* (pp. 14–18).

Nguyen, V.-T., & Bui, H.-A. (2022). A real-time defect detection in printed circuit boards applying deep learning. *EUREKA: Physics and Engineering,(2)*, 143–153.

Pahwa, K., & Kumar, R. (2017). Prediction of heart disease using hybrid technique for selecting features. In *2017 4th ieee uttar pradesh section international conference on electrical, computer and electronics (upcon)* (pp. 500–504).

Parlak, I. E., & Emel, E. (2023). Deep learning-based detection of aluminum casting defects and their types. *Engineering Applications of Artificial Intelligence*, *118*, 105636.

PCBGOGO. (2019). *What is the difference between solder mask and solder paste?* Retrieved from https://www.pcbgogo.com/Blog/What_Is_The_Difference_Between_Solder_Mask_And_Solder_Paste_.html (Accessed: 2023-08-28)

PHM_Society. (2022). *data challenge: 7th european conference of the prognostics and health management society 2022*. Retrieved 18.06.2022, from https://phm-europe.org/data-challenge

Picek, S., Heuser, A., Jovic, A., Bhasin, S., & Regazzoni, F. (2019). The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 209–237.

Pohl, E. A., & Dietrich, D. L. (1995). Environmental stress screening strategies for complex systems: A 3-level mixed distribution model. *Microelectronics Reliability*, *35*(4), 637–656.

Powers, D. M. (2020). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.

Putera, S. I., & Ibrahim, Z. (2010). Printed circuit board defect detection using mathematical morphology and matlab image processing tools. In *2010 2nd international conference on education technology and computer* (Vol. 5, pp. V5–359).

Pycaret — pycaret 2.3.5 documentation. (n.d.). Retrieved from https://pycaret.readthedocs.io/en/stable/ (Accessed: 2022-11-10)

Pytorch. (n.d.). Retrieved from https://pytorch.org/ (Accessed: 2022-11-10)

Quinlan, J. R. (1987). Simplifying decision trees. *International journal of man-machine studies*, *27*(3), 221–234.

Reitermanova, Z., et al. (2010). Data splitting. In *Wds* (Vol. 10, pp. 31–36).

Remeseiro, B., & Bolon-Canedo, V. (2019). A review of feature selection methods in medical applications. *Computers in biology and medicine*, *112*, 103375.

Salahouelhadj, A., Martiny, M., Mercier, S., Bodin, L., Manteigas, D., & Stephan, B. (2014). Reliability of thermally stressed rigid–flex printed circuit boards for high density interconnect applications. *Microelectronics Reliability*, *54*(1), 204–213.

Sankar, V. U., Lakshmi, G., & Sankar, Y. S. (2022). A review of various defects in pcb. *Journal of Electronic Testing*, *38*(5), 481–491.

Sarawade, A. A., & Charniya, N. N. (2018). Infrared thermography and its applications: a review. In *2018 3rd international conference on communication and electronics systems (icces)* (pp. 280–285).

Schmidt, I., Dingeldein, L., Hünemohr, D., Simon, H., & Weigert, M. (2022). Application of machine learning methods to predict the quality of electric circuit boards of a production line. In *Phm society european conference* (Vol. 7, pp. 550–555).

Shamsudin, H., Yusof, U. K., Jayalakshmi, A., & Khalid, M. N. A. (2020). Combining over-sampling and undersampling techniques for imbalanced classification: A comparative study using credit card fraudulent transaction dataset. In *2020 ieee 16th international conference on control & automation (icca)* (pp. 803–808).

Shwartz-Ziv, R., & Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, *81*, 84–90.

Spinzi, S. (2017). The evolution of industry 4.0, through the eyes of the pcb manufacturer. *EE-Evaluation Engineering*, *56*(7), 20–21.

Taco, J., Gore, P., Minami, T., Kundu, P., Suer, A., & Lee, J. (2022). A novel methodology for health assessment in printed circuit boards. In *Phm society european conference* (Vol. 7, pp.

556–562).

Tang, H., Tian, Y., Dai, J., Wang, Y., Cong, J., Liu, Q., ... Fu, Y. (2022). Prediction of production line status for printed circuit boards. In *Phm society european conference* (Vol. 7, pp. 563–570).

Thara, D., PremaSudha, B., & Xiong, F. (2019). Auto-detection of epileptic seizure events using deep neural network with different feature scaling techniques. *Pattern Recognition Letters*, *128*, 544–550.

Tiwari, D. (2014). Handling class imbalance problem using feature selection. *International Journal of Advanced Research in Computer Science & Technology*, *2*(2), 516–520.

Vafeiadis, T., Dimitriou, N., Ioannidis, D., Wotherspoon, T., Tinker, G., & Tzovaras, D. (2018). A framework for inspection of dies attachment on pcb utilizing machine learning techniques. *Journal of Management Analytics*, *5*(2), 81–94.

Vasilyev, F. V., Medvedev, A. M., Barakovsky, F. A., & Korobkov, M. A. (2021). Development of the digital site for chemical processes in the manufacturing of printed circuit boards. *Inventions*, *6*(3), 48.

Verna, E., Genta, G., Galetto, M., & Franceschini, F. (2023). Zero defect manufacturing: a self-adaptive defect prediction model based on assembly complexity. *International Journal of Computer Integrated Manufacturing*, *36*(1), 155–168.

Wang, Y., Wang, J., Cao, Y., Li, S., & Kwan, O. (2022). Integrated inspection on pcb manufacturing in cyber–physical–social systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *53*(4), 2098–2106.

Wang, Z., Yuan, H., Liu, Y., Li, J., Xu, H., & Zhang, X. (2022). A method for extracting features of infrared cloud image data of the printed circuit board. In *2022 ieee 17th conference on industrial electronics and applications (iciea)* (pp. 1257–1262).

Wongvorachan, T., He, S., & Bulut, O. (2023). A comparison of undersampling, oversampling, and smote methods for dealing with imbalanced classification in educational data mining. *Information*, *14*(1), 54.

Wu, C., Awasthi, A. K., Qin, W., Liu, W., & Yang, C. (2022). Recycling value materials from waste pcbs focus on electronic components: A review on technologies, obstruction and prospects.

Journal of Environmental Chemical Engineering, 108516.

Wu, H., Lei, R., & Peng, Y. (2022). Pcbnet: A lightweight convolutional neural network for defect inspection in surface mount technology. *IEEE Transactions on Instrumentation and Measurement*, *71*, 1–14.

Yang, J., Li, S., Wang, Z., Dong, H., Wang, J., & Tang, S. (2020). Using deep learning to detect defects in manufacturing: a comprehensive survey and current challenges. *Materials*, *13*(24), 5755.

Yuk, E. H., Park, S. H., Park, C.-S., & Baek, J.-G. (2018). Feature-learning-based printed circuit board inspection via speeded-up robust features and random forest. *Applied Sciences*, *8*(6), 932.