PREDICTING CONNECTION FAILURES IN WIRE BONDER MACHINE MATERIAL AND QUALITY DATA

ELINA VALDMANE

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

STUDENT NUMBER

2104590

COMMITTEE

Dr. Gonzalo Napoles
prof. dr. Chris Emmery

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

January 15th, 2024

WORD COUNT

7976/8000

ACKNOWLEDGMENTS

My appreciation goes out to my supervisor dr. Gonzalo Nápoles for the supervision during the writing of this thesis. My heartfelt gratitude goes to my dearest people Sandis and Diana for being a great support. A special shout-out to my close insta-friends for following this thesis journey and showing care and my course-mates Suzy and Leo. Additionally, I express my thanks to ELEO Technologies for providing the dataset and to Koen for his help.

CONTENTS

PREDICTING CONNECTION FAILURES IN WIRE BONDER MACHINE MATERIAL AND QUALITY DATA

ELINA VALDMANE

Abstract

While the wire bonding process has been extensively examined from an engineering standpoint, the application of machine learning to predict connection failures remains an under-explored area. With the growing importance of explainable AI, machine learning now holds the potential to provide valuable insights into high-complexity domains, assisting domain experts to gain a deeper understanding. This research employs four machine learning algorithms: Logistic Regression, Light Gradient Boosting Machine, Random Forest, and Support Vector Machine. Given the common association of the manufacturing domain with high volume and imbalanced datasets, this study introduces the use of Random undersampling and a novel, Presumably Correct Undersampling (PCU). Additionally, the research explores the trade-off between computational power and predictive performance when undersampling techniques are used. The study is based on a dataset with over 400 features and 200,000 instances. While the Light Gradient Boosting Machine achieved the highest score, Random Forests proved to be more consistent and reliable across multiple scenarios. Furthermore, PCU for Random Forest, demonstrated minimal decline in F1-score and Cohen's Kappa (0.74 to 0.73) and increased in AUC from 0.80 to 0.85 when compared to the model trained on the original dataset, therefore reducing computational load while preserving model's predictive power.

Data Source, Ethics, Code and Technology Statement

Work on this thesis did not involve collecting data from human participants or animals. The dataset used in this thesis was provided by ELEO Technologies B.V. and the original owner retains ownership of the data and code during and after the completion of this thesis. However, the company was informed about the use of this data for this thesis and potential research publications. All the figures belong to the author. The thesis code can be partly accessed through the GitHub repository following the link [https://github.com/valdmanee/MasterThesis]. In terms of writing, the author used assistance with the language of the paper. A generative language model, QuillBot and ChatGPT-3.5 was used to improve the author's original content for paraphrasing, spell checking and grammar. No other typesetting tools or services were used.

1 INTRODUCTION

This research aims to predict connection failures in wire bonder machines from machine-generated data using multiple Machine Learning algorithms and to test the application of undersampling specifically for this issue while taking into account the importance of features to gain an understanding of the connection failure problem.

1.1 *Motivation and Relevance*

As the world is transitioning towards energy sources that have a reduced environmental impact, the automotive industry is progressively shifting its focus to the utilization of electric traction motors powered by energy stored in batteries (Chu & Cui, 2023). Consequently, the activities related to battery production play a significant role in advancing the cause of a more sustainable world.

In the process of building these kinds of batteries, the wire bonding, a process where two metallic materials (wire and pad surface) are welded together, is an important state-of-the-art process of electronic packaging (Caiyuan & Ronglu, 2009). It is a time consuming, expensive and resource demanding process. With an ongoing digital transformation of the manufacturing sector and the ability to integrate data from production processes and quality assessments, there is significant potential to use machine learning and deep learning technologies to improve quality. Predictive quality, within this framework, allows manufacturers to make informed predictions about product quality by utilizing process data (Tercan & Meisen, 2022).

While the wire bonding process has been thoroughly examined from an electrical engineering standpoint, Machine Learning offers a fresh perspective, particularly in cases where applying practical quality tests is excessively time-consuming and costly. The application of explainable AI tools can provide valuable insights into the tasks of field specialists. According to a recent review on the research progress of wire bonding for microelectronic packaging by (Zhou et al., 2023): "It is the future trend to study the wire profile models and the reliability of wire bonding by FEA and ML algorithms."

The manufacturing sector poses two challenges: a high volume of generated data and the class imbalance problem. The data used in this thesis also employs these characteristics. Undersampling methods are preferred when dealing with extensive imbalanced datasets and limited computational resources, as they offer a more efficient approach within the available time constraints (Devi et al., 2020). A novel undersampling method called Presumably Correct Undersampling (Nápoles & Grau, 2023) will be applied to evaluate its effectiveness in addressing this problem.

Another challenge comes from the dual objectives of achieving high predictive performance while ensuring model explainability, which is a research domain that has experienced growth and innovation in the recent past (Amarasinghe et al., 2023). For this reason, this research is motivated to explore insights offered by explainability tools such as SHAP (SHapley Additive exPlanations) and permutation feature importance (PFI).

Hence, a notable research gap exists in the domain of machine learning applications, specifically addressing wire bonder connection failures. Also, the application of Presumably Correct Undersampling has not yet been tested outside the original paper. In response to that, the main research goal of this paper is to enhance the understanding and predictability of wire bonding failures. As well as contributing to the advancement of quality assurance processes in the manufacturing sector and evaluating the application of Presumably Correct Undersampling.

Therefore, this research contributes to the advancement of the state-of-the-art in predictive modeling for connection failures in wire bonding machines, offering a transparent, interpretable and computationally efficient solution.

1.2 *Research questions*

Considering the identified research gap and the societal significance, the **main research question** (RQ) of this study is as follows:

> *"To what extent is it possible to predict the Connection Failures in Wire bonder quality and material data using different Machine*

Learning algorithms (Logistic Regression, Light Gradient Boosting, Random Forests and Support Vector Machines) with different undersampling methods?"

The selection of the chosen algorithms is motivated by their demonstrated performance in classification tasks, with additional motivation from the Systematic review of machine learning (ML) methods for manufacturing processes (Simon et al., 2020). Also considering the nature of the data used, characterized by high volume and severe imbalance, this research has chosen to explore the applicability of undersampling methods.
To explore the machine learning models ability to generalize and gain insights on feature importance, the following sub-questions are posed:

RQ1: *Which Machine Learning algorithm (Logistic Regression (LR), Light Gradient Boosting (LGBM), Random Forests (RF) and Support Vector machine(SVM)) has the highest predictive power based on performance metrics F1, AUC and Cohen's Kappa score?*

To evaluate the models, three performance metrics will be used: the F1 measure, Cohen's Kappa, and the Area Under the ROC Curve measure. Additionally, the training set will be evaluated.

RQ2: *"To what extent Presumably Correct Undersampling (PCU) and Random Undersampling (RU) impact computational efficiency without compromising predictive power, as compared to models where no undersampling is applied? "*

The main drawback of undersampling is considered data loss, while a notable advantage is the reduction of computational power. A way to preserve the most relevant data that can represent the underlying problem while removing instances that do not add any predictive value is crucial. By training all models on three distinct imbalance treatment strategies: without undersampling, Random undersampling and Presumably Correct undersampling, the trade-off between computational power and predictive power is looked at in detail.

RQ3: *"To what extent the quality of feature importance explanations generated by SHAP compare to those produced by permutation feature importance and what insights on feature importance can be drawn from these methods?"*

The models developed in this research are intended to be used by operators in the field. Ensuring that these models provide actionable insights is crucial, enabling operators to identify and resolve machine-related problems,

thereby optimizing operational efficiency. To draw a comparison between feature explanations, a feature marginalization-performance curve will be made, which is explained in more detail in Section 3.9.5.

All the acronyms used in this research can be found in Appendix A.

2 LITERATURE REVIEW

2.1 *Machine Learning Applications in Wire bonding and Manufacturing*

Wire bonding is a process used widely and is a crucial component in power electronics, therefore, widely applied among manufacturers (Schuettler & Stieglitz, 2013). Specifically, wedge wire bonding is a subtle and fast process where ultrasonic waves are applied to aluminum wire using a tool called a wedge. Wedge bonding typically occurs very quickly, within a few hundred milliseconds, at ambient temperature (Kang et al., 2020). To assess the quality of wire bond interconnection various tests can be conducted, including the Bond pull test, ball-bond shear test, visual inspection and some specific quality examinations (Caiyuan & Ronglu, 2009).

Quality control and process management are of crucial importance in the manufacturing industry. As a result, there is an increasing tendency towards using machine learning (ML) models to find problems, evaluate parameters, and manage operations. This trend has been confirmed by the review conducted by (Simon et al., 2020), which underscores the extensive use of neural networks (NN) and decision tree algorithms in manufacturing and factory applications between 2015 and 2020 (Simon et al., 2020).

Comprehensive review on ML and data mining in manufacturing by (Alican & Derya, 2021) has shed light on several challenges, including the high dimensionality of data and the problem of data imbalance in manufacturing contexts (Alican & Derya, 2021). Additionally, (Kailong et al., 2022) addresses the issue of interpretability, highlighting that while various research have promising results in data-driven analyses and in forecasting of battery properties in manufacturing, limitations persist. For instance, many studies rely on conventional ML methods like NN and support vector machines (SVM), which can provide single-property predictions for battery products but lack sensitivity analysis of relevant manufacturing or control parameters (Kailong et al., 2022). This prevents engineers to gain a deeper understanding of their manufacturing processes.

According to the literature available for review, there has been limited research conducted specifically on applying machine learning models or deep learning techniques to draw predictions on wire bonding material and quality data. However, research has been conducted in related areas within the wire bonding and manufacturing domains. For instance, there

has been research on Failure Prediction Using Sequential Pattern Mining in the Wire Bonding Process. This work primarily focuses on sequential pattern mining techniques, which can help identify patterns leading to failures in the wire bonding process (Lim et al., 2017).

In a study by (Wenjie et al., 2023), the crucial challenge of detecting defects in wire bonding processes, vital for the reliability and performance of microwave components, is tackled. The author presents a three-stage defect detection method that combines deep learning with conventional image processing techniques. This method categorizes complex bonding images into four groups and systematically performs the detection process. (Wenjie et al., 2023).

Another study by (Kim et al., 2018) explores data-driven quality control techniques in smart factories. It concentrates on classifying manufacturing process conditions into normal and defective products, specifically addressing the challenge of imbalanced datasets with rare defective cases. Although the paper effectively handles this issue using a cost-sensitive decision tree ensemble algorithm, it predominantly relies on decision trees (Kim et al., 2018). Hence, there may be potential for further research into integrating deep learning techniques to improve the classification process.

2.2 *Data imbalance and Undersampling*

The characteristics of manufacturing systems encounter increasingly intricate, dynamic, and occasionally even chaotic behaviors (Thorsten Wuest & Klaus-Dieter, 2016), and many real-world classification problems, such as predicting manufacturing equipment failures, frequently involve imbalanced datasets (Lee & Seo, 2022). Data imbalance can negatively impact the detection capabilities of conventional machine learning algorithms. These algorithms are often designed to achieve high accuracy, leading to an overemphasis on the majority class and a neglect of the minority class (Meiling He & Francisco, 2023). There are commonly known two approaches to treat class imbalance in the data set: oversampling and undersampling. Despite the widespread use of the oversampling approach, driven in part by the increased availability and cost-effectiveness of technologies like GPU computing power and cloud computing environments (ElRafey & Wojtusiak, 2017), and acknowledging the potential information loss in undersampling (Susan & Kumar, 2021), there is ongoing discourse suggesting that intelligent undersampling might be the preferable approach, simultaneously addressing both the challenges of imbalance and high dimensionality.

The primary and fundamental undersampling technique is Random Undersampling (RUS). However, there are multiple widely known and

commonly used undersampling methods, such as Condensed Nearest Neighbor (CNN), Employed K-NN and Tomek-Link. These methods are categorized as "Pure Undersampling Approaches" by (Devi et al., 2020). Furthermore, "Hybrid Undersampling Approaches", integrating undersampling with the clustering process, have been widely developed. In this approach, clustering involves partitioning the majority class set into distinct clusters, each exhibiting unique underlying data properties. (Devi et al., 2020). Despite numerous undersampling methods being available, research on additional undersampling methods is ongoing. For example, (Paria et al., 2023) proposes an undersampling approach based on a meta-heuristic method in which the under-sampling problem is mapped into an optimization problem (Paria et al., 2023). A two-phase clustering-based undersampling method for imbalanced classification problems is created by (Farshidvard et al., 2023). The majority class is divided into clusters in such a way that the convex hull of each cluster's majority samples contains no minority samples; simultaneously, the size of each cluster is regulated (Farshidvard et al., 2023). Another more complex undersampling method proposed by (Shuo et al., 2024) offers the Learning-To-Rank Undersampling technique where undersampling process is considered as a learning-to-rank task. This means that a linear model is optimized to rank majority class instances and subsequently remove them from the bottom of the rank reducing the class imbalance, also introducing termination conditions (Shuo et al., 2024).

2.3 *Explainability and feature importance*

While there are advantages in using Artificial Neural Networks (NN) in fault diagnosis and prognosis for machine centers, such as fault tolerance, generalization and adaptability, a significant challenge is the lack of an explanation function (Li et al., 2017). Moreover, the lack of transparency and interpretability makes it difficult to identify weaknesses and make improvements in AI algorithms. While artificial neural networks demonstrate excellent classification performance, their internal model operates unclearly. This lack of transparency can create problems for the application of their results in practical decision-making contexts, such as in smart manufacturing (Alfeo et al., 2023). Hence, there is a demand for developing AI models with clear and interpretable behavior that can offer explanations, enabling end-users to understand decision-making processes, explore influential input factors, understand model mechanics and respond appropriately (Carletti et al., 2019).

There are generally two categories for feature importance determining techniques: model-specific and model agnostic. Model-specific techniques

look inside the model to understand the results including studying the structure of algorithm and its in-between steps. Model-agnostic techniques deals with inspecting features relationships with results, and the overall data pattern (Molnar, 2023). Some examples of model-agnostic methods are Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro et al., 2016), SHapley Additive exPlanations (SHAP) (Lundberg & Lee, 2017), Partial Dependence Plots (PDP) (Friedman, 2001) and Permutation-based Feature Importance (PFI) (Hastie et al., 2009). There is a noteworthy development in model-specific techniques arguing for being more informative for the specific model. For example, developing feature importance evaluation approach for Isolation Forest (Carletti et al., 2019), or a Long-term cognitive network (LTCN) designed for interpretable pattern classification of structured data, with its own method for providing explanations by calculating the importance of each feature in the decision-making procedure (Napoles et al., 2022). However, this is out of the scope for this research so only model-agnostic techniques will be discussed later.

3 METHODOLOGY

The upcoming section outlines and provides the methodology selected to address the proposed research questions and work setup used in the current study. Initially, the chosen models will be briefly outlined and compared. Next, data processing and EDA will be discussed. Following that, undersampling methods will be evaluated. Finally, feature relevance scores derived from SHAP and PFI will be compared. Various detailed graphics will be presented across different sections, offering insights into the chosen methodology.

3.1 *Models: LR, LGBM, RF and SVM*

The machine learning problem addressed in this paper is binary classification on an imbalanced, high-dimensional dataset with a focus on achieving high predictive performance and interpretability, which is crucial in the manufacturing domain for problem prevention. The choice for employing the LR model is inspired by the previously mentioned paper on failure prediction in the wire bonding process (Lim et al., 2017), providing a strong foundation for further research.

Next model selected for comparative analysis is LGBM. This choice serves as a performance benchmark and enables the evaluation of generalization capabilities. LGBM stands out as a state-of-the-art option for binary classification tasks involving structured datasets. It is known for its user-friendly nature, significantly faster training process and accuracy

(Schmitt, 2022), making it a noteworthy competitor to Deep Learning (DL) models.

The random forest (RF) algorithm (Breiman, 2001) is an ensemble learning approach based on decision trees. It forms a collection of decision trees used in both classification and regression tasks. RF uses bagging to enhance tree diversity, growing them from bootstrap samples and random subsets of input features. The predictions from these decision trees effectively reduce the impact of dataset noise, resulting in reduced overall model variance. Compared to traditional machine learning algorithms, RF demands minimal data preparation, offers a straightforward interpretation, and are less likely to overfit the data (Zhu et al., 2023).

A Support Vector Machine (SVM) (Cortes & Vapnik, 1995) is a machine learning algorithm designed for classification tasks. It works by identify decision hyperplanes in a high-dimensional space to separate different classes of data points. The hyperplane is positioned to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class, known as support vectors (Zhao, 2021). SVM is effective in handling high-dimensional data and are memory-efficient as they rely on a subset of training points, but they can be sensitive to outliers (Borah & Gupta, 2021).

3.2 *Description of the data set*

The data used for this research is ELEO Technologies B.V. property and was gathered from wire bonder machines used in ongoing battery module production in the period of February 2023 until June 2023. The data set consists of 424 features and 260,647 instances. Originally, this data set did not hold the classification feature "Connection Failure," since such failures are only identified through multiple tests conducted after the wire bonding process. To address this, a straightforward rationale is applied to create this feature: instances previously recorded are considered failed connections if the exact bond is detected again on the machine, indicating the introduction of rework. The data types identified in this data set were 'int64', 'float64', and 'object'.

3.3 *Data cleaning*

After understanding the characteristics of the data, the data cleaning is introduced which is graphically showed in Figure 1. The first step was to label the data accordingly as information is stored in database. Next, the missing data treatment was conducted. There were 2005 instances with missing data on two parameters. After exploring the missing data,

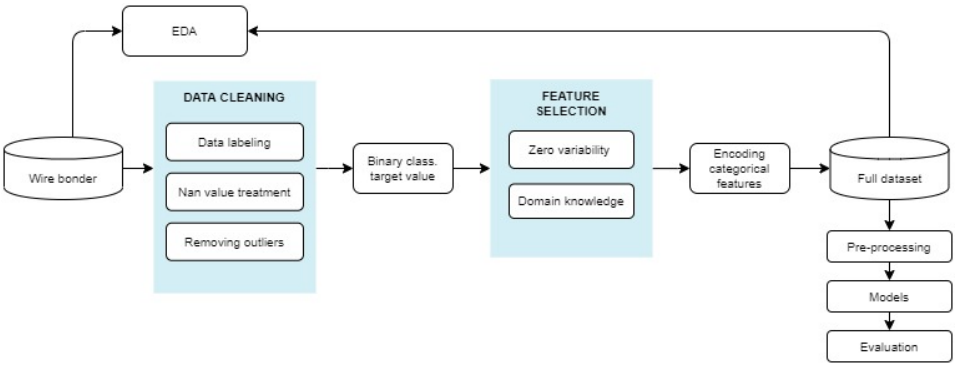

Figure 1: Illustration of data set pre-processing workflow.

it became evident that the absence of certain data points resulted from not setting up some parameters at the initial stages of working with the machine. Consequently, the decision was made to drop these instances, as it was an insignificant count of instances regarding this data set.

To identify potential outliers in battery model production—specifically, to find out whether additional tests or uncommon failures occurred during the production period—a summary was created for battery modules with more than 10 bond failures. Figure 2 shows the summary plot, revealing that one module stood out as an outlier. Consequently, all instances containing this PartId were removed from the data set to reduce bias in the predictions.

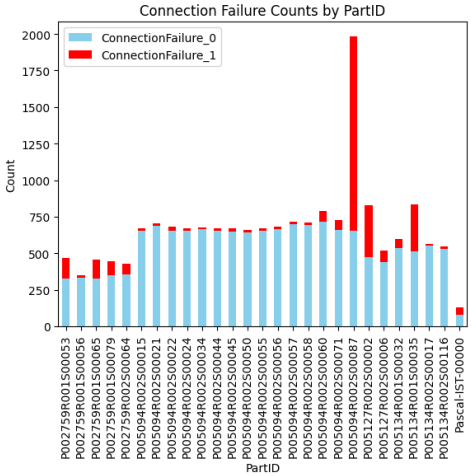

Figure 2: Connection failure counts by Part Id to identify outliers.

3.4 *Feature selection*

To attack the high dimensionality of this dataset, feature selection was carried out. It was decided to remove all features containing non-unique values as non-existing variability possibly could not influence the performance or give any insight. The Figure 3 shows that the count of features with no variability reaches 167, so all of them were removed. Furthermore, domain experts were consulted regarding features containing object data types to determine their relevance. Only features lacking practical information were excluded. For instance, features containing unit measure abbreviations or explanations were removed, as were features containing various types of identification numbers.

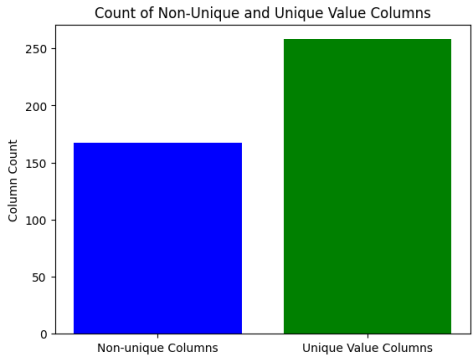

Figure 3: Illustration of data split and training set pre-processing workflow.

Following the feature removal process, the data set, initially comprising 424 features, was reduced to 192 features. However, four features (MachineID, PgmName, ChipName_1, ChipName_2) holding "object" data types were one-hot encoded to align with model requirements, as no ordinal relationship was found within these features. Consequently, the resulting data set consisted of 226 features.

Following the data pre-processing steps, which included data cleaning and handling missing values, the data set took on a final shape of 256,517 rows and 226 columns.

3.5 *Data split*

The dataset was divided into training and test sets, with 80% of the data allocated to the training set and the remaining 20% to the test set. Sklearn library data split function was used to do the task.

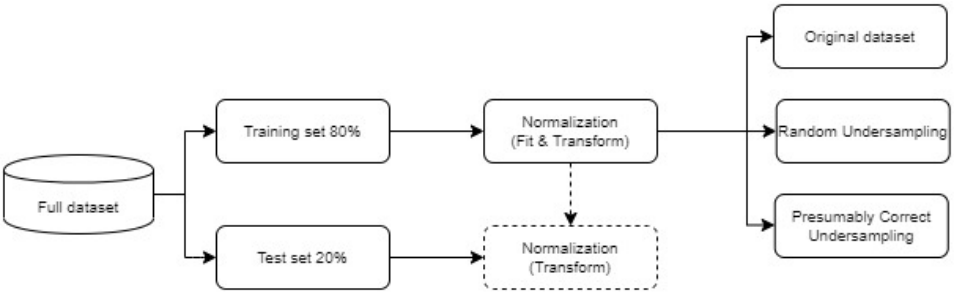

Figure 4: Illustration of data split and training set pre-processing workflow.

3.6 *Feature transformation*

To maintain the consistency, performance and interpretability, while also reducing ambiguity, Standartization (Raju et al., 2020) is applied as the features exhibit varying scales within the data set. For that StandardScaler from sklearn library was chosen. First the transformation function on training set was applied to calculate the mean and standard deviation of each feature in the training set. After that the test data is scaled using the parameters (mean and standard deviation) calculated from the training data, as displayed in Figure 4, to ensure consistency (Ali et al., 2014).

3.7 *Treating data imbalance*

To address the data imbalance in the dataset, as illustrated in Table 1, two undersampling techniques—Random Undersampling (RUS) and Presumably Correct Undersampling (PCU)—were selected. The choice to employ undersampling, as opposed to oversampling, was driven by considerations of computational power and to investigate the trade-off between computational and predictive power. Therefore, all models will be trained also using original dataset size.

Table 1: Target variable "Connection Failure" label count in data set.

"ConnectionFailure"	Count
0	254,203
1	2,304

3.8 *Undersampling methods*

Considering the research gap and the introduction of novelty, the choice of PCU was motivated to assess its performance, while RUS was selected due to its conventional application in similar problems.

While numerous undersampling methods exist, the most effective and straightforward technique is Random Undersampling (RUS). This non-heuristic algorithm aims to balance target distributions by randomly eliminating instances from the majority class. Although this approach may discard potentially valuable data crucial for classifier models, it proves beneficial when dealing with extensive datasets (Mohammed et al., 2020).

Presumably correct undersampling (PCU), introduced by (Nápoles & Grau, 2023), is an undersampling method where the basis of the algorithm is built upon the concept of Presumably Correct Decision Sets. These sets distinguish between instances that are presumed to be correct (those with neighbors sharing the same class label) and instances that are presumed to be incorrect (those with neighbors belonging to a different decision class). By substituting presumably correct instances from the majority class with prototypes, the algorithm seeks to achieve improved class balance without making substantial changes to the decision boundaries. The method involves reducing the size of the majority class while maintaining the decision boundaries in order to improve the classifier's generalization capabilities (Nápoles & Grau, 2023). The post-undersampling training set size and the time (in seconds) spent for undersampling is detailed in Table 2.

Table 2: Training set volume after undersampling and recorded undersampling time in seconds for different methods and the time.

	Trainig set size	Undersampling time (s)
Original training set	205,205	-
Random Undersampling	3,742	0.03
Presumably Correct Undersampling	42,537	25092.16

3.9 *Model evaluation and experiment workflow*

This section outlines the structure of the study and then explores it in more specific components in the subsequent sections. Figure 5 shows a workflow after the pre-processing and data split is done, continuing with model hyper-parameter tuning using grid search with cross-validation, after which a model evaluation is done by testing models on best hyper-parameters. When the best performing model is chosen, an feature

marginalization-performance curve is created on feature rankings created by SHapley Additive exPlanations (SHAP) and permutation feature importance (PFI).

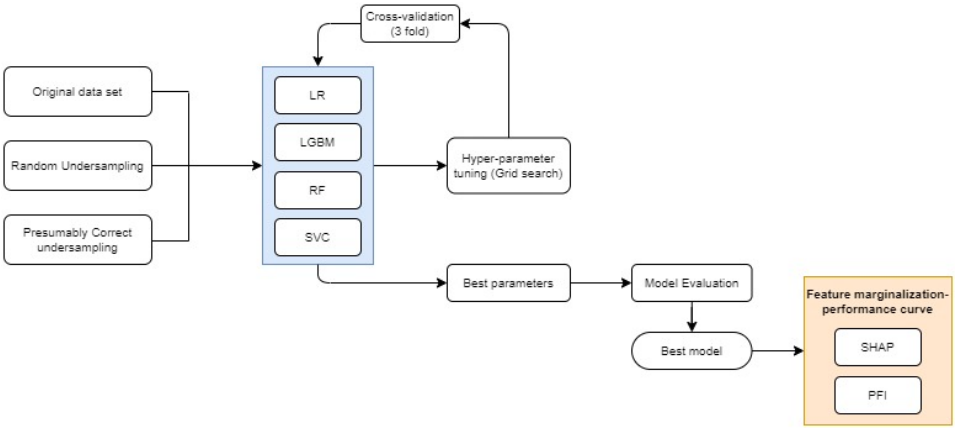

Figure 5: Illustration of model evaluation and experiment workflow.

3.9.1 *Hyper-parameter tuning*

To identify optimal configurations for each model, a grid search with 3-fold cross-validation is done on the training data and time for searching best parameters is recorded. This involves training the models on a training set that has been K-folded three times, where the model is trained using K-1 of the folds as the training data, and the remaining fold is used as the validation set, as represented in the Figure 6. As a grid search scoring method the F1 metric was chosen for its balancing nature, considering this as more robust measure.

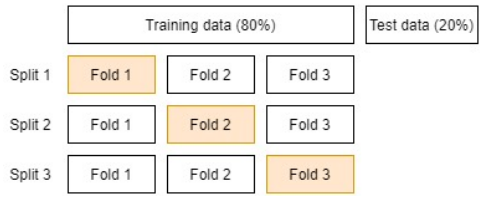

Figure 6: Illustration of cross-validation.

Hyperparameter settings are then selected based on the model's performance on these validation sets. Following this, the optimal parameters will be saved and applied to test the models on new, unseen data to evaluate their generalizability. For all model hyperparameter tuning, a parameter grid consisting of specified values was used.

The Logistic Regression (LR) parameter grid included the regularization parameter, the penalty type, the optimization algorithm and the maximum number of iterations. As the regularization parameter controls the inverse of regularization strength to prevent over-fitting by penalizing large coefficients, it was set to 0.001, 0.01, 0.1, 1, 10, and 20. Penalty included the values 'l1' (Lasso) and 'l2' (Ridge). For the algorithm to use in the optimization problem, the values 'lbfgs', 'sag' and 'saga' are chosen and the parameter for the maximum number of iterations, ensuring that the algorithm does not run indefinitely, values include 50, 100, 200, and 300.

The Light Gradient Boosting Machine (LGBM) parameter grid encompasses the following parameters: boosting learning rate, number of boosted trees, maximum three depth, maximum number of leaves in each tree, and maximum number of bins. For the learning rate, which controls the step size during the optimization, values of 0.01, 0,04 and 0.1 are included. Regarding the number of boosting rounds or trees in the ensemble, values of 50, 100, and 200 are set. The maximum depth parameter is explored with values 1, 3, 5, and 10. The maximum number of leaves in each tree has values of 5, 10, and 35. Finally, the maximum number of bins parameter, controlling the discretization of continuous features, is considered with values 15, 40 and 70 in the parameter grid.

The Random Forest (RF) parameter grid contained number of trees with values: 20, 50 and 100; maximum depth (5, 10, 15); minimum number of samples to split including 2, 5, 10 and 15; minimum number of samples required to be at a leaf with values 1, 2 and 4; maximum number of features that are considered for splitting a node where values are 'auto', 'sqrt' and 'log2'; and whether bootstrap samples are used when building trees holding boolean values.

The Support Vector Machine parameter grid holds regularization parameter with values 0.1, 1 and 10; kernel type including values 'linear', 'rbf', 'poly' and 'sigmoid'; an independent term in kernel function holding 0, 1 and 2; and limit on iterations with values 100, 200, 500, 600 and 700.

Also, Presumably Correct Undersampling (PCU) holds three parameters. Number of neighbors, inclusion and ratio. A higher number of neighbors implies fewer instances considered correct, making the identification of pure neighborhoods more challenging. The ratio is in the range [0, 1] and determines the reduction ratio, indicating the number of prototypes formed from isolated correct instances. In brief, larger ratio values yield fewer prototypes, with ratio = 1.0 leading to the removal of all correct instances without prototype construction and ratio = 0.0 designating each correct instance as a prototype (Nápoles & Grau, 2023). In this PCU application default hyper-parameter values were used, where parameters are set to: number of neighbors = 5, inclusion = partial and ratio = 0.8.

3.9.2 *Evaluation*

To determine the best-performing model and assess their performance under various undersampling strategies, the evaluation will rely on the F1-score, Cohen's Kappa, and AUC metrics.

The choice of the F1-score is motivated by its balancing nature between recall and precision. This metric considers both false positives and false negatives, making it a valuable measure when seeking a balance between these two aspects. Moreover, F1-score is regarded as more robust measure, especially in the presence of imbalanced classes. Also (Jiang et al., 2020) and (Fathy et al., 2020) use F1-score when dealing with an imbalance class.

The inspiration to use Cohen's Kappa comes from the original LTCN paper (Napoles et al., 2022), where it is employed as a robust classification score for running simulations. This metric assesses the agreement between raters for categorical items and has a scale from -1 to 1. A score of -1 signifies no agreement between the prediction and the ground-truth values; 0 indicates no learning (equivalent to random prediction), and 1 signifies total agreement or perfect performance. Cohen's Kappa provides a reliable measure of agreement beyond chance and is particularly suited for classification tasks.

Area Under the ROC Curve (AUC) shows the ability of a binary classifier to distinguish between positive and negative classes. The ROC curve is commonly illustrated as a plot. The AUC summarizes classifier performance with a value between 0 and 1, where a higher AUC indicates better discrimination ability. Support for reporting on AUC was found in the work by (Kim et al., 2018), as it involved imbalanced classification.

3.9.3 *Feature explanation methods*

In this research two feature explanation methods have been chosen: SHAP (SHapley Additive exPlanations) and PFI (Permutation Feature Importance) for assigning a score to each feature based on its significance in the model. SHAP (Lundberg & Lee, 2017) values is a technique used to explain the output of machine learning models by assigning each feature's contribution to the prediction. These values are based on cooperative game theory and are designed to fairly distribute the "credit" for a particular prediction among the different features ("SHAP", n.d.). SHAP holds an advantage in providing transparent and interpretable explanations at the local level. This means that each observation can have its unique SHAP values, offering personalized insights into why a specific instance receives its prediction and the contributions made by its predictors (Dwivedi et al., 2023). Including SHAP in this research allows a comparative analysis of feature importance rankings.

The PFI method is a two-step process that is repeated for each feature. The first step involves permuting, or, in other words, rearranging the column related to the chosen feature. In the second step, the decline in prediction performance is evaluated. Features that show the most significant performance decline following permutation are likely to be the most pertinent (Carletti et al., 2019). This method is chosen due to its simplicity and ease of interpretation.

3.9.4 *Retrieving feature rankings*

In the process of generating feature importance rankings from SHAP values first step is to compute SHAP values for each instance in the dataset. The contribution of each feature to the output of the model for a particular instance is quantified by the SHAP values, which are obtained from the Python SHAP library. However, a number of actions are taken in order to convert these numbers into a meaningful feature importance ranking. Calculating the average absolute SHAP values for every feature across all instances is the second step in aggregating the SHAP values. Each feature's impact can be determined by calculating the absolute value of each SHAP value, regardless of the impact's direction—positive or negative—and this is represented in equation (1). The final step involves ranking the features based on their aggregated SHAP values. Features with higher average absolute SHAP values are considered to have a greater influence on the model's predictions.

$$\text{SHAP\_feature\_importance\_scores} = \frac{1}{N}\sum_{i=1}^{N}|\text{shap\_values}_i| \qquad (1)$$

In the context of feature importance ranking, the PFI involves systematically evaluating individual features within a machine learning model to discern their influence on the overall model performance.

The process to obtain a feature importance ranking is that for each feature within the dataset, a permutation function is applied to shuffle the values of the selected feature within the testing set. This ensures the disruption of relationships between the feature and the target variable. Subsequently, the model's performance is assessed using the testing set, which contains permuted feature values. The performance metric F1 is then computed for the scenario where the feature values have been permuted.

After that an importance score is calculated for each feature by evaluating the difference in performance between the baseline scenario (original feature values) and the permuted scenarios. The importance score is calculated using the formula displayed in equation (2). The importance score represents the degree to which a feature contributes to the model's predictive power.

$$\text{Importance Score}_i = \text{Baseline Performance} - \text{Permutated Performance}_i \tag{2}$$

After obtaining feature rankings, features are subsequently ranked based on their computed importance scores in descending order. Features with higher importance scores are considered to have a higher rank, signifying a higher impact on the model's overall performance.

3.9.5 *Feature ranking comparison*

Post-model evaluation, an experiment will be conducted to evaluate the importance of different features. The area under the marginalization-performance curve will be used to evaluate the quality of these explanations. This experiment involves iteratively altering individual features by marginalizing them within different subsets of the data. The focus will be on observing the rate at which the evaluation metric (e.g., model performance) changes as features are manipulated. A faster decline in the metric suggests that the corresponding feature set holds greater importance.

3.10 *Software*

Every phase were executed utilizing the programming language Python 3.11.4 using Visual Studio Code 1.84.2 The subsequent libraries were used:

- Pandas (version 2.1.3)
- Numpy (version 1.24.3)
- Scikit-learn (version 1.3.0)
- Imbalanced-learn (version 0.11.0)
- Shap (version 0.43.0)
- Seaborn (version 0.12.2)
- Matplotlib (version 3.8.1)

4 RESULTS

The following section presents the outcomes for four models: Logistic Regression, Light Gradient Boosting Machine, Random Forests and Support Vector Machines. To start, the hyperparameter optimization for all models is revealed. Following that, and in accordance with the framework of the

research questions, the performance of all models will be shown. Next, the impact of the employed undersampling methods will be evaluated. Lastly, the varieties observed in feature importance rankings will be addressed.

4.1 *Hyperparameter tuning*

This section offers an insight into the hyperparameter values identified through grid search on the Original dataset, Random Undersampling (RU) and Presumably Correct Undersampling (PCU) configurations. Figure 7 visualize the grid search results for Logistic Regression (LR) to observe trends and determine the best hyperparameter settings for each model. Appendix B includes all the grid search plots for model hyper-parameter tuning, showing figures created for each model. Table 3 shows an overview of the best hyper-parameters for the models using different undersampling methods.

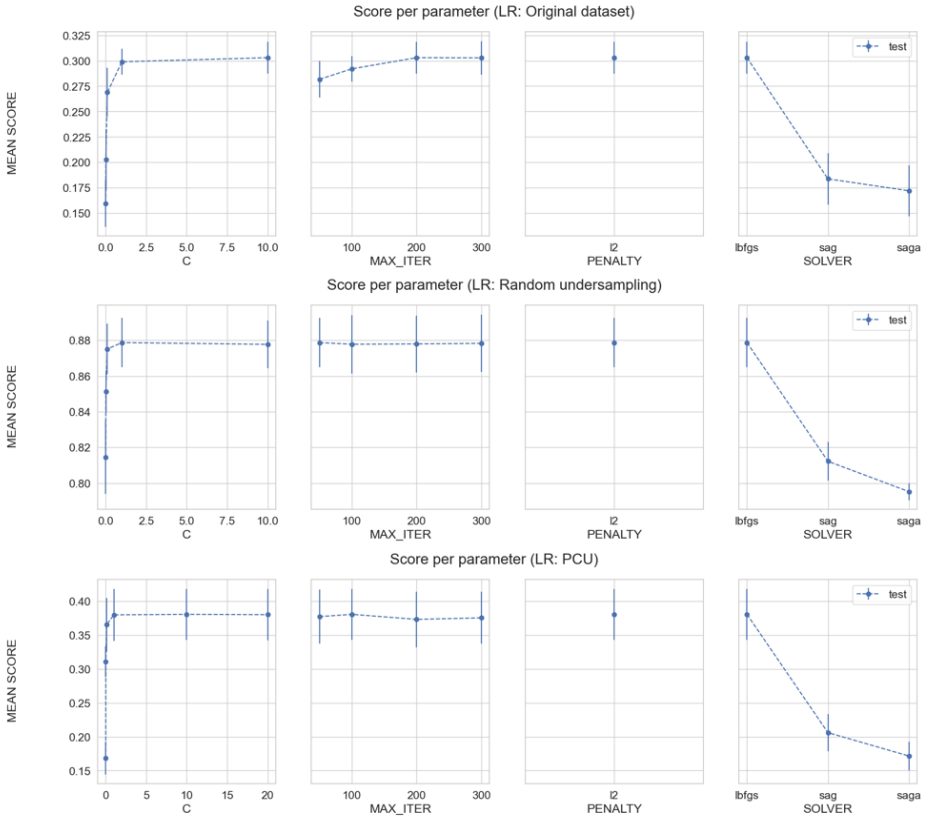

Figure 7: Illustration of grid search results for Logistic regression under all undersampling configurations

In the hyperparameter grid search for Logistic Regression across various undersampling scenarios, the 'l2' penalty consistently optimized the algorithm, with 'lbfgs' selected as the preferred optimization method. Random undersampling indicated a regularization parameter of 1, unlike the original dataset and PCU scenarios, where it was set to 10. Iteration limits differed with random undersampling at 50, PCU at 100 and the original dataset at 200. This difference suggests that the iteration limits may need to be increased for larger datasets.

Table 3: Best Model Hyper-parameters on different undersampling methods

Model	**Hyper-parameter**	**Original**	**RU**	**PCU**
LR	C	10	1	10
	max_iter	200	50	100
	penalty	l2	l2	l2
	solver	lbfgs	lbfgs	lbfgs
LGBM	learning_rate	0.1	0.1	0.1
	max_bin	70	40	40
	max_depth	10	10	10
	n_estimators	200	200	200
	num_leaves	35	35	35
RF	bootstrap	FALSE	FALSE	FALSE
	max_depth	15	15	15
	max_features	sqrt	sqrt	sqrt
	min_samples_leaf	1	1	1
	min_samples_split	2	5	2
	n_estimators	20	50	100
SVM	C	10	10	10
	coefo	0	0	0
	kernel	rbf	rbf	rbf
	max_iter	700	500	700

In all scenarios for LGBM, the majority of hyperparameters remained consistent. However, there was an exception to the maximum number of bins assigned for feature bucketing. The original dataset required 70 bins, whereas the random undersampling and PCU scenarios required 40 bins.

A grid search for Random Forest model hyperparameters found that using the complete dataset for tree construction is the best method across all undersampling conditions, which is done by setting bootstrapping to false. The other determined parameters included a maximum tree depth of 15 (15 being the largest number in the parameter grid), a maximum features parameter, which was considered the square root of the total number of features, and a minimum of 1 sample required to be a leaf

node, indicating that these models are encouraged to create deeper trees with fewer terminal nodes. In contrast, the minimum number of samples to split for the LGBM model under random undersampling was set at 5, while the original dataset and PCU scenarios had this parameter set to 2. Furthermore, the number of parameter estimators varied across all undersampling conditions.

The grid search for Support Vector Machines found that the optimal parameters across all scenarios were a regularization parameter of 10, an independent term in the kernel function set to 0, and the Radial Basis Function as the chosen kernel. However, the maximum iteration parameter varied, with random undersampling requiring 500 iterations, while the original dataset and PCU required 700. This result implies that increasing the quantity of the training data may entail increasing the number of iterations required for effective convergence.

The time spent on grid search for each undersampling scenario was recorded (in seconds) to compare the computational resources and is displayed in Table 4. Evidently, the grid search on a randomly undersampled training set was the fastest, followed by the PCU, while the grid search on the original dataset took the most time. This pattern is consistent with the findings in Table 2 which shows the sizes of the training sets.

However, as compared to the original dataset, PCU displayed efficiency that was more than three times faster across all methods. Furthermore, the grid search for both the Original data and PCU training sets spent the most time on the models Random Forest and Support Vector Machine. In the case of Random undersampling, however, the majority of time was spent on grid searches for Light Gradient Boosting and Random Forest models.

Table 4: Grid search time recorded in seconds

Model	**Original**	**RU**	**PCU**
LR	2,652.98	39.81	630.55
LGBM	2,484.06	132.22	648.82
RF	12,357.82	215.54	3,415.4
SVM	5,111.98	85.39	1,022.81

4.2 *Performance of the models*

The table 5 displays an overview of the performance on both training set and test set to evaluate and compare the model behaviour and ability to generalize unseen data.

To begin, it is clear that Logistic Regression (LR) and Support Vector Machines (SVM) consistently show comparatively lower performance

across all scenarios. Although LR performs better when trained on the Original and PCU sets, SVM outperforms LR when trained on a randomly undersampled dataset.

On the other hand, the most prominent models are Light Gradient Boosting (LGBM) and Random Forest (RF). While LGBM consistently demonstrates strong overfitting across all undersampling scenarios, obtaining perfect scores (1.00) across all metrics when trained on different training sets, it performs quite well. Noteworthy, the overall highest metrics are obtained by LGBM when trained on Original dataset, achieving 0.8 on F1-score, indicating that the model is performing well in terms of both precision and recall, and 0.8 on Cohen's Kappa, suggesting strong agreement between the model's predictions and the actual values. Also achieving 0.86 on AUC. The LGBM performance on unseen data when trained on the PCU set is moderate, with an F1-score of 0.52 and a Cohen's Kappa score of 0.51. However, its ability to distinguish between positive and negative classes is notable, achieving an AUC of 0.87.

When comparing Random Forests (RF) to LGBM under the Original dataset scenario, Random Forests come as a close second, achieving F1 and Cohen's Kappa scores of 0.74 and an AUC of 0.80. However, when models are trained with the PCU dataset, RF outperforms all other models with F1 and Cohen's Kappa scores of 0.73 and an AUC of 0.85.

Table 5: Results on training and test set under different undersampling methods

Method	**Model**	**Training set**			**Test set**		
		F1	**Kappa**	**AUC**	**F1**	**Kappa**	**AUC**
Original dataset	LR	0.32	0.32	0.60	0.32	0.32	0.60
	LGBM	1.00	1.00	1.00	**0.80**	**0.80**	**0.86**
	RF	0.83	0.83	0.86	0.74	0.74	0.80
	SVC	0.22	0.21	0.69	0.20	0.19	0.67
Random undersampling	LR	0.89	0.78	0.89	0.10	0.09	0.87
	LGBM	1.00	1.00	1.00	**0.23**	**0.21**	**0.94**
	RF	1.00	1.00	1.00	0.22	**0.21**	0.93
	SVC	0.95	0.89	0.95	0.14	0.12	0.91
Presumably Correct Undersampling	LR	0.40	0.39	0.64	0.30	0.30	0.63
	LGBM	1.00	1.00	1.00	0.52	0.51	**0.87**
	RF	0.93	0.92	0.93	**0.73**	**0.73**	0.85
	SVC	0.42	0.39	0.71	0.12	0.10	0.67

When comparing the impact of different undersampling scenarios, it is evident that all models trained on a randomly undersampled training set tend to overfit, failing to generalize to unseen data when compared to models trained on the original or PCU dataset. Consequently, achieving

low F1 and Cohen's Kappa scores, however, obtained the highest AUC when compared to other undersampling scenarios.

In contrast, the models exhibited their highest performance when trained on the original data. Nevertheless, when trained on the PCU set, there was only a slight reduction in performance for Logistic Regression (LR) and Random Forest (RF), but the decrease in performance for Light Gradient Boosting (LGBM) and Support Vector Machine (SVM) was more substantial.

Considering that the grid search using PCU was three times less expensive than the Original dataset and that the Random Forest model's performance with the Original dataset is relatively close to LGBM, yet it clearly outperforms LGBM when trained on the PCU set, we conclude that Random Forest is the best-performing model.

4.3 *Feature importance rankings and performance-marginalization curve*

To discover insights within the model features, a detailed analysis including feature importance rankings and a progressive performance-marginalization experiment was conducted. Identifying the features that contribute the most to the model's predictive capabilities not only aids in generating actionable insights, but also in detecting potential faults within the machine or environment setup.

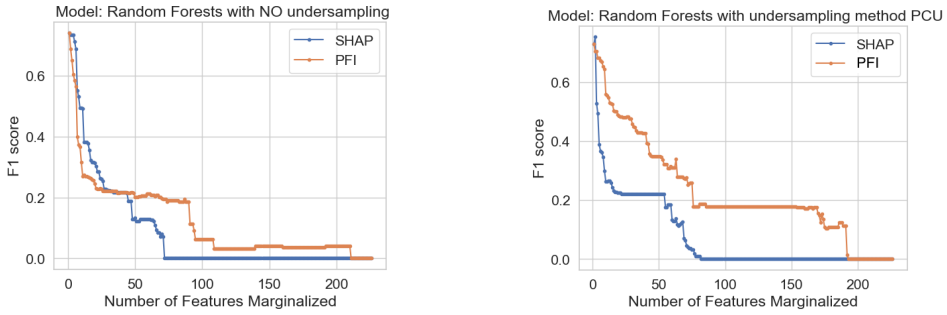

Figure 8: Comparison of SHAP and PFI feature ranking when progressively marginalizing most important features on Model performance. Random Forest model in two different training scenarios: trained on Original dataset; trained on PCU dataset

Two feature importance methods, namely SHAP and Permutation Feature Importance (PFI), were tested within the context of a performance-marginalization curve. The goal was to figure out which method provided the most valuable insights into the key features of the best-performing model. This experimentation was carried out in two scenarios: a model

trained on the Original dataset and a model trained on the dataset using the PCU method for undersampling. The outcomes of this experiment are illustrated in Figure 8.

The graphs show that the feature importance approach and the undersampling scenario have a considerable impact on feature importance detection. Under the scenario of no undersampling, both SHAP and PFI display a similar decline for the first 15 features. However, around the 25th feature, PFI begins to stabilize, while SHAP continues to decline, reaching 0.00 in F1 with the 65th feature, whereas PFI achieves 0.00 in performance, only marginalizing at the 210th feature.

In the scenario where Random Forest is trained on undersampled data using the PCU method, SHAP outperforms PFI. It reduces performance from 0.74 to 0.22 when only the top 17 features are marginalized.

Upon comparing both graphs, the fastest decline in model performance, indicating the most influential features, is observed for the SHAP method applied to Random Forest trained on a training set undersampled by PCU. Table 6 presents the top 13 most important features as indicated by SHAP under PCU.

Table 6: Top 13 most important features indicated by SHAP

Rank	Feature
1	BondProcessId_1
2	BondProcessId_2
3	BoatNr
4	BondPosZ_2
5	BondHeight_2
6	MachineID_WB-3 Timothy 62-600504
7	WireGuideCount
8	CutterCount
9	BondPosX_2
10	USG_Phase_1
11	BondPosZA_2
12	ModuleNr
13	BondPosX_1

The most important feature table shows that machine setup-related features, such as Bond Process ID and Machine ID, along with features associated with Bond position, have the highest influence on the model's predictive power. Additionally, specific product identification features, such as Boat Nr. and Module Nr., account for the model's ability to generalize.

5 DISCUSSION

The main objective of this research was to assess the generalization capabilities of different machine learning algorithms in predicting failures in wire bonder machines and to explore the impact of different undersampling scenarios on computational efficiency without reducing predictive power. Another aim was to extract insights from the models regarding feature importance, providing valuable information to domain experts. To address the main research question, the results will be analyzed from three perspectives: model performance, the impact of undersampling, and the insights derived from feature importance rankings and performance-marginalization curves.

5.1 *Model performance*

The results in Table 5 reveal differences and impressive outcomes between several models trained on both the original and undersampled datasets.Notably, Light Gradient Boosting (LGBM) and Random Forest (RF) models outperformed Logistic Regression (LR) and Support Vector Machines (SVC). SVC's performance as the worst model could be related to its sensitivity to imbalanced datasets or its failure to efficiently navigate decision boundaries within this particular dataset.

LGBM and RF demonstrated exceptional performance, establishing themselves as the best-performing models capable of optimally handling the high-volume imbalanced dataset. This also demonstrates the possibility of generating accurate predictions for the binary classification problem addressed in this study.

While LGBM exhibited outstanding results, its behavior displayed some inconsistencies. Because of the overfitting on all training sets, LGBM's performance on the test set was fluctuating. This erratic performance raises concerns regarding model generalizability and reliability in real-world scenarios. Looking at the Hyperparameter graph in Appendix B, it appears that fixing the overfitting problem in LGBM may have been accomplished by limiting both the maximum depth and the number of leaves.

On the other hand, RF showed more stable and consistent behavior, particularly in returning results that were more representative of its performance on the test set. Although it showed evidence of overfitting, its impact on the test set was smaller than that of LGBM. This implies that RF may be a more dependable option in this scenario, demonstrating a compromise between high performance and generalizability.

5.2 *Discussion on the trade-off between predictive performance and computational power*

Although models trained on the original dataset generally received the highest scores on the test set, the results for Random Forest, when trained on the original dataset and PCU, experienced only a minimal decrease in F1-score and Cohen's Kappa score (from 0.74 to 0.73) while actually increasing in AUC. This implies that almost no predictive power was lost. Moreover, referring to Table 4, the grid search conducted on PCU took three times less time than the original dataset, significantly enhancing computational efficiency. PCU, therefore, demonstrates noteworthy performance for the Random Forest algorithm. Similar behavior could be observed with the Logistic regression model, where model performance slightly decreased when trained on the PCU dataset.

However, one drawback of PCU is the computational resource required to create the undersampled dataset, as illustrated in Table2. The time investment is considerably extensive and incomparable to the speed of random undersampling. It is essential to note that PCU has hyperparameters that can be tuned to enhance undersampling speed by reducing the ratio, which indicates the number of prototypes formed from isolated correct instances. Nevertheless, this might carry the risk of losing information if the undersampling becomes too aggressive. A reasonable strategy would be to tune these PCU hyperparameters for optimal performance from this undersampling method.

5.3 *Feature importance*

The performance-marginalization curve for comparing different feature importance ranking methods proved to be a valuable approach. This method is simple with easy-to-understand graphs, yet it provides a lot of information. Since there are various ways to rank features, this has helped to compare those methods.

The Figure 8 make it evident that SHAP provides more clarity in explaining the model compared to PFI. This conclusion is drawn from the faster decline in performance observed with SHAP, suggesting that the features marginalized first in the ranking hold more predictive power in the model. Notably, the most significant drop in performance was observed with PCU, emphasizing the utmost importance of certain features in this model. The Table 6 displays that the most critical features aligns well with real-world considerations. For instance, the bond position is crucial in the wire bonding process, making these insights highly valuable.

To explore important features further and reduce dataset dimensionality, an additional step could involve performing feature selections based on these rankings. Following that, systematically assessing the models' performance with revised feature sets would provide a more complete understanding of the impact of various features on overall predictive capabilities.

5.4 *Limitations and Further Recommendations*

The limitation that has already been emphasized in this work is the dataset, as it introduced high dimensionality, substantial volume, and class imbalance. Due to limited computational resources, a hyperparameter tuning for the Presumably Correct Undersampling (PCU) method and the application of nested cross-validation for model hyperparameter tuning were not possible.

Also, considering the specific dataset used in this research, generalizing the model to unseen data from different period in time might pose some challenges. Certain features related to unique part identification could introduce bias, as the model might learn to recognize parts that are generally more prone to faults.

Further recommendations to improve the work would be to conduct more extensive hyperparameter tuning for the Light Gradient Boosting Machine (LGBM) to prevent overfitting. Additionally, using insights from feature importance methods to reduce dimensionality can contribute to making the model more generalizable.

5.5 *Contributions and Societal Impact*

This research has made a novel contribution to the field of predicting wire bonding connection failures by employing multiple machine learning algorithms and introducing model explainability. The inclusion of model explainability has significantly narrowed down the search space across all machine settings, sharing insights on which features contribute most to failures. Additionally, the emphasis on data processing has established an optimal framework for addressing future applications of this problem.

Another novelty is the introduction of Presumably Correct Undersampling (PCU) in this research. Since this method has not been tested in previous studies, its application in this work marks the first instance of its use. The outcomes of PCU performance proved to be significant, establishing PCU as an undersampling method that addresses the imbalance issue while preserving data and reducing computational demands. This work

supports the application of PCU as an undersampling method that can be employed in addressing other imbalances or high-volume tasks.

6 CONCLUSION

In conclusion, the responses to the proposed sub-questions will collectively contribute to answering the main research question:

> **RQ1**: *"Which Machine Learning algorithm (Logistic Regression (LR), Light Gradient Boosting (LGBM), Random Forests (RF) and Support Vector machine(SVM)) has the highest predictive power based on performance metrics: F1, AUC and Cohen's Kappa score?"*

Table 5 contains the response to the first research question, indicating that LGBM and RF are the best-performing algorithms. When models are trained on the original dataset, LGBM outperforms RF, but when trained on the PCU dataset, RF outperforms LGBM. Nevertheless, when performance on training sets is taken into account, RF with its consistent performance, emerges as a more reliable choice under the specified parameters.

> **RQ2:** *"To what extent Presumably Correct Undersampling (PCU) and Random Undersampling (RU) impact computational efficiency without compromising predictive power, as compared to models where no undersampling is applied?"*

Random undersampling used significantly less computational power, being at least 18 times faster in grid search compared to the search on the original dataset, as can be observed in Table 4. However, there was a notable drop in predictive power, for example, by reducing the F1-score of the LGBM model from 0.80 to 0.23.

PCU proved to be at least three times faster in grid search than the search on the original dataset, showing only a minor drop in performance for LR and RF. Although LGBM and SVM showed a greater loss in performance, the trade-off between computational efficiency and predictive capability under PCU was justified for RF and LR.

> **RQ3:** *"To what extent the quality of feature importance explanations generated by SHAP compare to those produced by permutation feature importance and what insights on feature importance can be drawn from these methods?"*

As Figure8 illustrates, the feature explanations generated by SHAP provided more clarity in explaining the model compared to PFI. Table 6 displays insight into the top 13 important features, suggesting that machine setup and bond position are the most important.

REFERENCES

Alfeo, A. L., Cimino, M. G. C. A., & Gagliardi, G. (2023). Concept-wise granular computing for explainable artificial intelligence. *Granular Computing*, *8*(4), 827–838. https://doi.org/10.1007/s41066-022-00357-8

Ali, P. J. M., Faraj, R. H., Koya, E., Ali, P. J. M., & Faraj, R. H. (2014). Data normalization and standardization: A technical report. *Mach Learn Tech Rep*, *1*(1), 1–6.

Alican, D., & Derya, B. (2021). Machine learning and data mining in manufacturing. *Expert Systems with Applications*, *166*, 114060. https://doi.org/https://doi.org/10.1016/j.eswa.2020.114060

Amarasinghe, K., Rodolfa, K. T., Lamba, H., & Ghani, R. (2023). Explainable machine learning for public policy: Use cases, gaps, and research directions. *Data & Policy*, *5*, e5.

Borah, P., & Gupta, D. (2021). Robust twin bounded support vector machines for outliers and imbalanced data. *Applied Intelligence*, *51*(8), 5314–5343.

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

Caiyuan, W., & Ronglu, S. (2009). The quality test of wire bonding. *Modern Applied Science*, *3*. https://doi.org/10.5539/mas.v3n12p50

Carletti, M., Masiero, C., Beghi, A., & Susto, G. A. (2019). Explainable machine learning in industry 4.0: Evaluating feature importance in anomaly detection to enable root cause analysis. https://doi.org/10.1109/SMC.2019.8913901

Chu, Y., & Cui, H. (2023, September). Annual update on the global transition to electric vehicles: 2022. https://theicct.org/wp-content/uploads/2023/06/Global-EV-sales-2022_FINAL.pdf

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*, 273–297.

Devi, D., Biswas, S. K., & Purkayastha, B. (2020). A review on solution to class imbalance problem: Undersampling approaches. https://doi.org/10.1109/ComPE49325.2020.9200087

Dwivedi, R., Dave, D., Naik, H., Singhal, S., Omer, R., Patel, P., Qian, B., Wen, Z., Shah, T., Morgan, G., & Ranjan, R. (2023). Explainable ai (xai): Core ideas, techniques, and solutions. *ACM Comput. Surv.*, *55*(9). https://doi.org/10.1145/3561048

ElRafey, A., & Wojtusiak, J. (2017). Recent advances in scaling-down sampling methods in machine learning [Export Date: 19 November 2023; Cited By: 18]. *Wiley Interdisciplinary Reviews: Computational Statistics*, *9*(6). https://doi.org/10.1002/wics.1414

Farshidvard, A., Hooshmand, F., & MirHassani, S. A. (2023). A novel two-phase clustering-based under-sampling method for imbalanced classification problems. *Expert Systems with Applications*, *213*, 119003. https://doi.org/https://doi.org/10.1016/j.eswa.2022.119003

Fathy, Y., Jaber, M., & Brintrup, A. (2020). Learning with imbalanced data in smart manufacturing: A comparative analysis. *IEEE Access*, *9*, 2734–2757.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of statistics*, 1189–1232.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (Vol. 2). Springer.

Jiang, D., Lin, W., & Raghavan, N. (2020). A novel framework for semiconductor manufacturing final test yield classification using machine learning techniques. *Ieee Access*, *8*, 197885–197895.

Kailong, L., Mona Faraji, N., Geanina, A., Michael, L., David, G., & James, M. (2022). Interpretable machine learning for battery capacities prediction and coating parameters analysis. *Control Engineering Practice*, *124*, 105202. https://doi.org/https://doi.org/10.1016/j.conengprac.2022.105202

Kang, H., Sharma, A., & Jung, J. P. (2020). Recent progress in transient liquid phase and wire bonding technologies for power electronics. *Metals*, *10*(7), 934. https://doi.org/10.3390/met10070934

Kim, A., Oh, K., Jung, J.-Y., & Kim, B. (2018). Imbalanced classification of manufacturing quality conditions using cost-sensitive decision tree ensembles. *International Journal of Computer Integrated Manufacturing*, *31*(8), 701–717. https://doi.org/10.1080/0951192x.2017.1407447

Lee, W., & Seo, K. (2022). Downsampling for binary classification with a highly imbalanced dataset using active learning. *Big Data Research*, *28*, 100314. https://doi.org/https://doi.org/10.1016/j.bdr.2022.100314

Li, Z., Wang, Y., & Wang, K.-S. (2017). Intelligent predictive maintenance for fault diagnosis and prognosis in machine centers: Industry 4.0 scenario. *Advances in Manufacturing*, *5*(4), 377–387. https://doi.org/10.1007/s40436-017-0203-8

Lim, H. K., Kim, Y., & Kim, M.-K. (2017). Failure prediction using sequential pattern mining in the wire bonding process. *IEEE Transactions on Semiconductor Manufacturing*, *30*(3), 285–292. https://doi.org/10.1109/tsm.2017.2721820

Lundberg, S., & Lee, S.-I. (2017, December). A unified approach to interpreting model predictions.

Meiling He, M. P. P. L. W. O., & Francisco, M. (2023). Learning with supervised data for anomaly detection in smart manufacturing. *International Journal of Computer Integrated Manufacturing*, *36*(9), 1331–1344. https://doi.org/10.1080/0951192X.2023.2177747

Mohammed, R., Rawashdeh, J., & Abdullah, M. (2020). Machine learning with oversampling and undersampling techniques: Overview study and experimental results. *2020 11th International Conference on Information and Communication Systems (ICICS)*, 243–248. https://doi.org/10.1109/ICICS49469.2020.239556

Molnar, C. (2023). Interpretable machine learning. https://christophm.github.io/interpretable-ml-book/

Napoles, G., Salgueiro, Y., Grau, I., & Espinosa, M. L. (2022). Recurrence-aware long-term cognitive network for explainable pattern classification. *IEEE Transactions on Cybernetics*, 1–12. https://doi.org/10.1109/tcyb.2022.3165104

Nápoles, G., & Grau, I. (2023). Presumably correct undersampling. *Iberoamerican Congress on Pattern Recognition*, 420–433.

Paria, S., Feizi-Derakhshi, M. R., & Mahdi, H. (2023). Addressing the class-imbalance and class-overlap problems by a metaheuristic-based under-sampling approach. *Pattern Recognition*, *143*, 109721. https://doi.org/https://doi.org/10.1016/j.patcog.2023.109721

Raju, V. N. G., Lakshmi, K. P., Jain, V. M., Kalidindi, A., & Padma, V. (2020). Study the influence of normalization/transformation process on the accuracy of supervised classification. *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 729–735. https://doi.org/10.1109/ICSSIT48917.2020.9214160

Ribeiro, M., Singh, S., & Guestrin, C. (2016, June). "why should I trust you?": Explaining the predictions of any classifier. In J. DeNero, M. Finlayson, & S. Reddy (Eds.), *Proceedings of the 2016 conference of the north American chapter of the association for computational linguistics: Demonstrations* (pp. 97–101). Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-3020

Schmitt, M. (2022). Deep learning vs. gradient boosting: Benchmarking state-of-the-art machine learning algorithms for credit scoring.

Schuettler, M., & Stieglitz, T. (2013). 4 - microassembly and micropackaging of implantable systems. In I. Andreas & H. Diana (Eds.), *Implantable sensor systems for medical applications* (pp. 108–149). Woodhead Publishing. https://doi.org/https://doi.org/10.1533/9780857096289.1.108

Shap [SHAP documentation: Introduction to explainable AI]. (n.d.). Retrieved December 2, 2023, from https://shap.readthedocs.io/

en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html

Shuo, F., Jacky, K., Yan, X., Peichang, Z., Xiao, Y., & Xiaochun, C. (2024). Improving the undersampling technique by optimizing the termination condition for software defect prediction. *Expert Systems with Applications*, *235*, 121084. https://doi.org/https://doi.org/10.1016/j.eswa.2023.121084

Simon, F., Christopher, P., & Bernd, K. (2020). Systematic review on machine learning (ml) methods for manufacturing processes – identifying artificial intelligence (ai) methods for field application [53rd CIRP Conference on Manufacturing Systems 2020]. *Procedia CIRP*, *93*, 413–418. https://doi.org/https://doi.org/10.1016/j.procir.2020.04.109

Susan, S., & Kumar, A. (2021). The balancing trick: Optimized sampling of imbalanced datasets—a brief survey of the recent state of the art. *Engineering Reports*, *3*(4), e12298. https://doi.org/https://doi.org/10.1002/eng2.12298

Tercan, H., & Meisen, T. (2022). Machine learning and deep learning based predictive quality in manufacturing: A systematic review. *Journal of Intelligent Manufacturing*, *33*(7), 1879–1905. https://doi.org/10.1007/s10845-022-01963-8

Thorsten Wuest, D. W. C. I., & Klaus-Dieter, T. (2016). Machine learning in manufacturing: Advantages, challenges, and applications. *Production & Manufacturing Research*, *4*(1), 23–45. https://doi.org/10.1080/21693277.2016.1192517

Wenjie, P., Tang, T., Ming, C., & Fan, M. (2023). Automatic detection of wire bonding defects in microwave components using multi-stage hybrid methods based on deep learning. *Measurement Science and Technology*, *34*(11), 115001. https://doi.org/10.1088/1361-6501/ace926

Zhao, J. (2021). The development and application of support vector machine. *Journal of Physics: Conference Series*, *1748*(5), 052006. https://doi.org/10.1088/1742-6596/1748/5/052006

Zhou, H., Zhang, Y., Cao, J., Su, C., Li, C., Chang, A., & An, B. (2023). Research progress on bonding wire for microelectronic packaging. *Micromachines*, *14*(2), 432. https://www.mdpi.com/2072-666X/14/2/432

Zhu, M., Yang, Y., Feng, X., Du, Z., & Yang, J. (2023). Robust modeling method for thermal error of cnc machine tools based on random forest algorithm. *Journal of Intelligent Manufacturing*, *34*(4), 2013–2026. https://doi.org/10.1007/s10845-021-01894-w

APPENDIX A

Table 7: Explanation of acronyms used in the work

Acronym	Meaning
EDA	Exploratory data analysis
RUS	Random Undersampling
PCU	Presumably Correct Undersampling
LR	Logistic Reggression
LGBM	Light Gradient Boosting Machine
RF	Random Forests
SVM	Support Vector Machine
SHAP	SHapley Additive exPlanations
PFI	Permutation Feature Importance

APPENDIX B

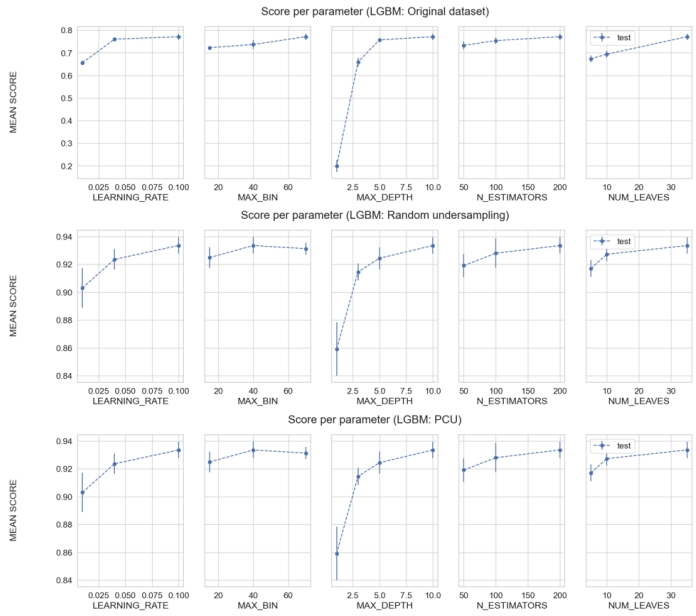

Figure 9: Illustration of grid search results for Light Gradient Boosting under all undersampling configurations

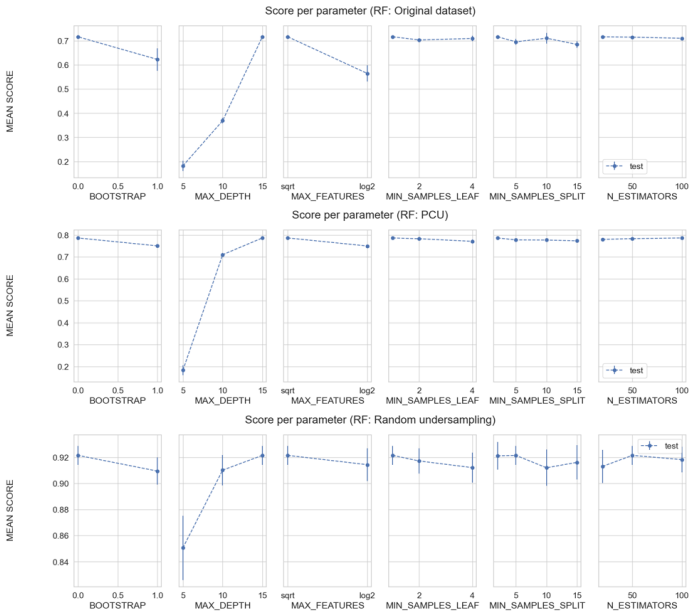

Figure 10: Illustration of grid search results for Random Forest under all undersampling configurations

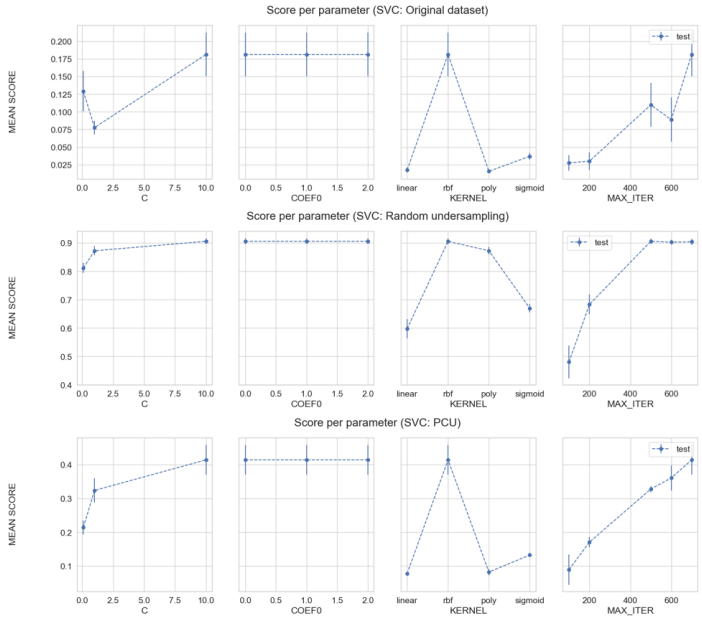

Figure 11: Illustration of grid search results for Support Vector Machine under all undersampling configurations