YAŞAR UNIVERSITY

GRADUATE SCHOOL

MASTER IN SCIENCE

MACHINE LEARNING FOR PREDICTIVE MAINTENANCE

SEJMA CICAK

THESIS ADVISOR: ASSIST. PROF. UMUT AVCI, PH.D.

DEPARTMENT OF COMPUTER ENGINEERING

BORNOVA / İZMİR
DECEMBER 2023

JURY APPROVAL PAGE

We certify that, as the jury, we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Jury Members:	**Signature:**
Assist.Prof. (PhD) Umut AVCI Yaşar University	.....
Assoc.Prof. (PhD) Korhan KARABULUT Yaşar University	.....
Assist.Prof. (PhD) İlker KORKMAZ İzmir University of Economics	.....

Prof. (PhD) Yücel Öztürkoğlu
Director of the Graduate School

ABSTRACT

MACHINE LEARNING FOR PREDICTIVE MAINTENANCE

Cicak, Sejma
MSc, Computer Engineering
Advisor: Assist.Prof. (PhD) Umut Avcı
December 2023

Imbalanced data is a common problem in many areas, significantly impacting the performance and generalizability of machine learning models. This is primarily because the models struggle to construct a robust representation of the instances in the minority class. Our study focuses on enhancing classification success in predictive maintenance tasks, where data imbalance is a prevalent challenge. To address this, we employ resampling methods designed to create balanced datasets.
We introduce a variety of oversampling and undersampling methods and implement them on synthetic and real-world datasets. Additionally, we explore the effectiveness of hybrid approaches by combining multiple sampling methods to enhance classification results. We conduct classification experiments using both imbalanced and balanced datasets employing diverse classifiers. We compare the performances of these classifiers and, of particular significance, assess the efficacy of resampling techniques to gain insights into their effectiveness in addressing class imbalance. Our study contributes to the expanding literature on mitigating class imbalance in classification tasks and offers practical guidance for selecting suitable sampling methods based on dataset characteristics.

Keywords: machine learning, big data, classification, imbalanced data, predictive maintenance, sampling methods

ÖZ

BAKIM ÖNGÖRÜSÜ İÇIN MAKINE ÖĞRENIMI

Cicak, Sejma
Yüksek Lisans, Bilgisayar Mühendisliği
Danışman: Dr. Öğr. Üyesi Umut Avcı
Aralık 2023

Dengesiz veriler birçok alanda yaygın bir sorundur ve makine öğrenimi modellerinin performansını ve genelleştirilebilirliğini önemli ölçüde etkiler. Bunun temel nedeni, modellerin azınlık sınıfındaki örneklerin sağlam bir temsilini oluşturmakta zorlanmasıdır. Çalışmamız, veri dengesizliğinin yaygın bir sorun olduğu tahmine dayalı bakım görevlerinde sınıflandırma başarısını artırmaya odaklanıyor. Bu sorunu çözmek için dengeli veri kümeleri oluşturmak üzere tasarlanmış yeniden örnekleme yöntemlerini kullanıyoruz. Çeşitli aşırı örnekleme ve yetersiz örnekleme yöntemlerini tanıtıyoruz ve bunları sentetik ve gerçek dünya veri kümelerine uyguluyoruz. Ek olarak, sınıflandırma sonuçlarını geliştirmek için birden fazla örnekleme yöntemini birleştirerek hibrit yaklaşımların etkinliğini araştırıyoruz. Çeşitli sınıflandırıcılar kullanan hem dengesiz hem de dengeli veri kümelerini kullanarak sınıflandırma deneyleri yapıyoruz. Bu sınıflandırıcıların performanslarını karşılaştırıyoruz ve özellikle önemli olan, sınıf dengesizliğini gidermedeki etkinliklerine ilişkin içgörü kazanmak için yeniden örnekleme tekniklerinin etkinliğini değerlendiriyoruz. Çalışmamız, sınıflandırma görevlerinde sınıf dengesizliğinin azaltılmasına ilişkin genişleyen literatüre katkıda bulunmakta ve veri kümesi özelliklerine göre uygun örnekleme yöntemlerinin seçilmesi için pratik rehberlik sunmaktadır.

Anahtar Kelimeler: makine öğrenimi, büyük veri, sınıflandırma, dengesiz veri, öngörü bakım, örnekleme yöntemleri

ACKNOWLEDGEMENTS

First, I would like to extend my sincere gratitude to my supervisor, Umut Avci, whose guidance and support played a pivotal role in navigating through the challenges of this academic journey. His invaluable insights and unwavering mentorship provided constant support and motivation, helping me navigate and overcome the complexities encountered during the thesis.

I extend gratitude to my parents for their constant love, support, and care throughout every phase of my life. I also appreciate my brother and father for their instrumental assistance in configuring the virtual machines essential for my computational work. Their technical expertise and collaboration significantly streamlined the process, saving valuable time.

Lastly, I acknowledge my husband for his unwavering belief in my capabilities and continual support and assistance throughout this academic endeavor.

Sejma Cicak

İzmir, 2023

TEXT OF OATH

I declare and honestly confirm that my study, titled “MACHINE LEARNING FOR PREDICTIVE MAINTENANCE” and presented as a Master’s Thesis, has been written without applying any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

Sejma Cicak

December 1, 2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SYMBOLS AND ABBREVIATIONS

SYMBOLS:

TN	True Negatives
TP	True Positives
FN	False Negatives
FP	False Positives

ABBREVIATIONS:

ROC	Receiver operating characteristic
PHM	Prognostics and Health Management
WEKA	Waikato Environment for Knowledge Analysis
SMOTE	Synthetic Minority Oversampling Technique
ADASYN	Adaptive Synthetic Sampling
ROS	Random Oversampling
TOMEK	Tomek Links
CLUSTER	Cluster Centroids
ENN	Edited Nearest Neighbor
RUS	Random Undersampling
ML	Machine Learning
PM	Preventative Maintenance
PdM	Predictive Maintenance
RM	Reactive Maintenance
HPP	Harzing's Publish or Perish
DB	Database
CRM	Customer Relationship Management
CLI	Command Line Interface

1. INTRODUCTION

Classification in machine learning involves predicting class labels for input data examples. It entails training a machine learning model on a labeled dataset, where each example includes input features and their corresponding class labels. The model uses statistical and computational techniques to identify patterns, relationships, and decision boundaries within the data. By learning from these labeled examples, the model aims to generalize its understanding and make accurate predictions on unseen instances. In recent years, one of the major issues faced in classification tasks is handling imbalanced data. Imbalanced data is any set of data that consists of remarkably more members of one class (the majority) than another (the minority) (Hoens & Chawla, 2013). Some real-world examples include but are not limited to the identification of fraud (Kraiem et al., 2021), the classification of spam emails (Kraiem et al., 2021), spotting oil spills in satellite radar photos (Kulkarni et al., 2020), the diagnosis of rare diseases (Ramyachitra & Manikandan, 2014), and summarizing and filtering data (Kotsiantis et al., 2005). This class imbalance can occur due to various factors, such as the rarity of certain events, biased data collection processes, or inherent imbalances in the problem domain. Data is highly imbalanced in all these domains since the interesting cases are fewer than usual ones (Kotsiantis et al., 2005).

Dealing with imbalanced data introduces several complexities and biases that impact the performance of classification models. Traditional machine learning algorithms are often designed with an assumption of balanced class distributions, where each class has a comparable number of instances. However, when dealing with imbalanced data, these algorithms often display an inclination toward the majority class, resulting in less-than-optimal predictions for the minority class. Consequently, the models may struggle to capture the characteristics and nuances of the minority class, leading to diminished accuracy and potential misclassifications.

Correctly categorizing the minority class is crucial as its misclassification will have greater negative effects than misclassifying the majority group. In diagnosing rare

diseases, for instance, the cost of misclassifying an ill person as healthy would lead to vital consequences (Kraiem et al., 2021). Imbalanced data also causes problems in the evaluation of machine learning models as the model accuracy becomes an improper evaluation metric. This is illustrated by the example where 99% of the dataset comprises the majority class, and only 1% is made up of the minority class instances. Suppose a classifier predicts all instances as the majority class. In that case, it can achieve an accuracy of 99%, but this accuracy is highly misleading as it disregards the classifier's performance on the minority class. In this scenario, accuracy is an unsuitable metric as it neglects to consider the distinct costs associated with the two types of errors (false positives and false negatives). Thus, in imbalanced datasets, accuracy alone cannot accurately measure the model's performance (Hoens & Chawla, 2013).

To address the limitations of accuracy as a metric, we must use evaluation metrics specifically designed to consider the performance of the minority class. One commonly used metric is recall, also known as sensitivity or true positive rate. Recall evaluates a classifier's capability to accurately recognize instances of the minority class, capturing the ratio of correctly classified actual positive instances. It emphasizes minimizing false negatives, representing instances of the minority class incorrectly labeled as the majority class.

In imbalanced datasets, incorporating recall into the evaluation process provides a more comprehensive understanding of the model's performance. It allows us to assess how well the model identifies instances of the minority class, often of particular interest in applications involving rare events or critical anomalies. By considering recall alongside other metrics like precision, F1-score, or area under the ROC curve, we can obtain a more nuanced evaluation of the model's performance, considering both the majority and minority classes, as well as the different costs associated with false positives and false negatives. In our research, we recognize the importance of accurately assessing the performance of classification models on imbalanced data. By utilizing recall, we aim to better understand how well our models can identify instances of the minority class.

In recent years, addressing the challenge of imbalanced data has received significant attention in the literature, leading to the development of various data- and algorithm-

level solutions (Kotsiantis et al., 2005). Data-level studies present rebalancing approaches that can enhance prediction model performance and show how data impacts the performance of the models (Tantithamthavorn et al., 2020; Bennin et al., 2019).

On the other hand, algorithm-level approaches have been involved in addressing the imbalanced data challenge to some extent. These adjustments involve modifying various aspects of the algorithms, including class costs, probabilistic estimates, decision thresholds, and the utilization of recognition-based learning techniques. Adjusting class costs allows for assigning different weights or penalties to misclassifications, enabling the algorithm to prioritize accurate predictions for the minority class. By appropriately modifying these parameters, the algorithm is incentivized to mitigate the impact of class imbalance and give more emphasis to the minority class. These algorithm-level modifications enable the learning process and decision-making criteria to adapt to the unique challenges of imbalanced class distributions. Mixture-of-experts approaches are also used, combining multiple classifiers induced after applying data-level solutions. Small disjuncts can be error-prone due to rarity, so understanding why they are error-prone can help explain the problem. Most classifier induction systems prevent overfitting to remove non-meaningful disjuncts. Inductive bias favors more common classes in the presence of uncertainty. Techniques for handling imbalanced datasets include adjusting class priors and mining rare cases (Kotsiantis et al., 2005).

This study focuses on using data-based solutions, specifically sampling techniques, to address data imbalance. These techniques aim to generate a new dataset with a balanced distribution of classes by resampling the original dataset (Hoens & Chawla, 2013). This is achieved mainly by oversampling or undersampling approaches. The former is used to increase the samples in the minority class, while the latter decreases the samples in the majority class (Kraiem et al., 2021). Furthermore, in addition to applying oversampling and undersampling individually, we will also explore the utilization of a combination of these techniques. By employing a mixed approach of undersampling and oversampling, our aim is to enhance the balance in the dataset and improve the performance of our classification models. Leveraging the benefits of both techniques contributes to addressing the complexities associated with imbalanced data from multiple perspectives. This all-encompassing methodology seeks to attain a

representation of the underlying class distributions that is both robust and accurate, ultimately enhancing the effectiveness of learning and classification outcomes.

SMOTE, ADASYN, and ROS are among the most popular oversampling approaches. The ROS method simply duplicates the existing examples in minority classes without modification (Kulkarni et al., 2020). SMOTE generates new examples of minority classes based on the neighboring minority classes (Rathore et al., 2022). ADASYN, a variation of SMOTE, aims to reduce bias and deploy adaptive learning by generating more synthetic data from minority class samples that are harder to learn (Kulkarni et al., 2020), (Ramyachitra & Manikandan, 2014). Common approaches to undersampling, comprising TOMEK, CLUSTER, ENN, and RUS, are used to handle imbalanced datasets by eliminating majority class instances to create a more balanced representation of minority and majority classes. Random undersampling is a basic method that involves removing instances from the majority class at random (Kulkarni et al., 2020). However, to address its limitations, clustering groups similar samples (Lin et al., 2017), while Tomek Link creates links using distance measures to eliminate majority class instances (Kulkarni et al., 2020). Additionally, Edited Nearest Neighbors strategically applies a nearest neighbor rule to intelligently remove noise, thereby contributing further to the refinement of imbalanced datasets (Batista et al., 2004).

Sampling methods have plusses and minuses. As the oversampling techniques enhance the representation of the minority class, they are likely to perform better than the undersampling techniques. On the other hand, there may be scaling problems as the oversampling methods increase the number of examples. Increasing the number of examples through oversampling may result in a larger dataset, which can pose challenges related to computational resources and training time. This is not an issue for the undersampling techniques as they work on a downscaled version of datasets. However, undersampling methods may suffer from losing useful information by discarding majority class examples (Rathore et al., 2022). This loss of information may lead to a degradation in the model's ability to generalize and make accurate predictions for the majority class instances. Therefore, striking a balance between preserving the crucial information from the majority class and emphasizing the minority class becomes a critical consideration when choosing sampling techniques. Selecting the optimal sampling method for predictive maintenance requires a comprehensive

evaluation of dataset characteristics, including size, while considering the specific needs of the domain and the importance of precise predictions for both majority and minority class instances.

One specific domain where data imbalance can pose a significant challenge is predictive maintenance for industrial machines. Machines used in industry have been designed to work reliably for long periods of time, and failures occur rarely during the life of the machine. Hence, the majority of the measurements taken from a manufacturing process represent normal working conditions. This results in heavily unbalanced predictive maintenance datasets with few examples of erroneous cases.

This study aims to contribute to the growing body of literature on addressing the issue of class imbalance in classification tasks for predictive maintenance. We evaluate the performance of various oversampling and undersampling techniques and analyze the effectiveness of different classifiers in improving classification success. In addition to evaluating individual oversampling and undersampling techniques, we will further explore the potential performance gains achieved by utilizing a combined approach that integrates both oversampling and undersampling techniques. Toward this end, we use both synthetic and real datasets. We utilized the AI4I 2020 Predictive Maintenance Dataset (Matzka, 2020) as a common benchmark for the synthetic data. For real-world evaluation, the PHM Data Challenge 2018 corpus (Bonatakis et al., 2018) was employed. This dataset serves as a real-world scenario to assess the benefits and performance of the applied sampling methods. We have performed our experiments with the WEKA toolkit, which provides a robust and standardized platform for evaluating the effectiveness of various classification techniques. By utilizing both synthetic and real datasets, we aim to ensure the generalizability of our findings across different data scenarios and demonstrate the practical applicability of the proposed sampling methods. This research's findings will help develop more accurate and efficient classification models for datasets with imbalanced classes. These findings will not only advance the field of class imbalance handling but also enable practitioners to make informed decisions when addressing imbalanced data challenges in real-world predictive maintenance scenarios.

2. LITERATURE REVIEW

1.1. Predictive Maintenance and Its Application Domain

In manufacturing industries, system failure is a common event that can lead to significant disruptions. To mitigate such risks, the concept of Preventative Maintenance (PM) has emerged, focusing on preventing asset failures and their associated consequences. In recent years, Predictive Maintenance (PdM) utilizing machine learning techniques has gained prominence as a proactive approach to identifying patterns of system failure and scheduling maintenance based on real-time data analysis (Kane et al., 2022).

The prevailing trend in data processing and analysis centers on the application of Machine Learning (ML) methods, renowned for their efficacy in deploying sophisticated, intelligent algorithms that can deliver precise predictions (Samatas et al., 2021). ML exhibits the ability to process large volumes of data, allowing for the effective discovery of hidden correlations within complex and ever-changing environments (Kane et al., 2022). These environments span various sectors, including heavy industry, the automotive industry, aviation/aeronautics, business assets, and even household appliances. PdM within these sectors can yield significant economic benefits (Samatas et al., 2021).

This ML-based predictive approach analyzes live data, harnesses ML technology to identify correlations among various parameters, and effectively predicts system failures or schedules maintenance as needed. By leveraging ML's capabilities, organizations can accurately identify potential faults, predict failures in a timely manner, and optimize resource utilization. This approach ensures a balanced allocation of maintenance requirements and resources, enabling efficient operations within manufacturing industries (Kane et al., 2022).

Moreover, the advancement of modern technologies, such as the Internet of Things, sensing technology, and artificial intelligence, has shifted maintenance strategies from RM to PM to PdM (Ran et al., 2019). While Reactive Maintenance (RM) is solely

carried out after equipment failure occurs, resulting in delays and high costs, Preventive Maintenance follows a predetermined schedule to prevent breakdowns but can lead to unnecessary maintenance and high prevention costs. In contrast, Predictive Maintenance relies on real-time estimations of equipment "health" to enable timely pre-failure interventions (Ran et al., 2019). By leveraging online health assessments, Predictive Maintenance allows for interventions before failures occur, reducing downtime and minimizing the costs associated with reactive repairs. This shift towards Predictive Maintenance achieves the optimal balance between reactive and preventive approaches, minimizing maintenance frequency and preventing unplanned Reactive Maintenance, thereby avoiding excessive costs associated with excessive Preventive Maintenance (Ran et al., 2019).

Predictive Maintenance (PdM) utilizing Machine Learning is applied across various sectors, leading to the identification of six categories. These categories encompass Aviation/Aeronautics, Business, Electronics, Production, Energy, and Transportation sectors. The Aviation and Aeronautics sector encompasses maintenance implementations specific to flying vehicles, aircraft, and helicopters. The Business category includes studies focused on damages to company buildings, copiers, and even freezers in grocery stores. The electronics applications of PdM are predominantly concerned with monitoring various electronic components. Within the Production category, PdM plays a vital role in maintaining the efficient operation of machines involved in manufacturing processes, enabling the production of diverse products. The energy sector applications of PdM are dedicated to systems involved in energy production, while the Transportation category focuses on implementing PdM techniques in transportation systems (Samatas et al., 2021).

As an illustration, Vafaei et al. (2019) introduced a fuzzy alarm system in their study to anticipate premature equipment degradation within a car production line. The primary objective was to mitigate expenses associated with abrupt shutdowns. Similarly, Wei et al. (2019) proposed a condition-based maintenance strategy to determine the optimal course of action, such as taking no action or implementing corrective replacement, based on the system's state. The ultimate goal of this strategy was to minimize the average cost rate.

A wide array of sensors is utilized to collect essential data for machine condition monitoring to enable effective predictive maintenance. Based on comprehensive

scientific studies, twelve distinct sensor categories have emerged, each serving a specific purpose and function. These categories encompass Temperature, Vibration, Noise/Earphone, Pressure, Accelerometer, Rotary, Air Flow, Deceleration, Humidity, Gyroscope, Current, Speed, and Voltage sensors. By collecting data through these sensors, predictive maintenance systems can effectively monitor machine health, enabling proactive maintenance actions and minimizing unplanned downtime. Among these categories, Temperature, Vibration, and Noise/Acoustic sensors are the most commonly used, representing 60.71%, 46.42%, and 32.14% respectively. Pressure sensors account for 28.57% of the identified categories, while Accelerometer and Rotary sensors comprise 25% and 14.28%, respectively. Air Flow and Deceleration sensors, as well as Humidity sensors, share the same percentage at 10.71%. Finally, Gyroscopes, Current, Speed, and Voltage sensors have an equal representation at 7.14% (Samatas et al., 2021).

During the period from 2011 to 2020, an extensive analysis was conducted to identify publications that focused on the implementation of PdM with Machine Learning. Appendix 1 explores a comprehensive overview of datasets used in various works. Grouped by algorithms, this appendix provides insights into the type of data, references, publication years, data availability, data sources, and data descriptions. The study involved querying the HPP database (DB), which presented challenges in selecting appropriate titles and keywords due to the preexistence of PdM without mandatory ML integration. Out of the initial pool of 847 publications, a subset of 44 demonstrated the implementation of PdM models with ML, while 33 publications specifically examined the analysis of methods. This analysis provides valuable insights into the integration of ML techniques in PdM applications, shedding light on the advancements made in this field during the specified period (Samatas et al., 2021).

When applying Machine Learning to Predictive Maintenance, a systematic process is followed, consisting of several stages. The primary objective is to achieve accurate maintenance predictions through model training, utilizing more efficient data collection and utilization techniques. This process commences with the selection and pre-processing of historical data recorded over past working cycles. In Predictive Maintenance (PdM), this crucial phase, also known as the data acquisition step, plays a fundamental role in determining how data is collected and stored. It involves the careful curation of valuable information to design effective machine learning models,

ensuring the acquisition of relevant data for assessing the overall health of the system (Jardine et al., 2006).

These data are collected through sensors within an IoT network, constantly updated with new measurements. Valuable data is then exported and pre-processed to ensure compatibility with the ML model. The subsequent stages involve the model's selection, training, and validation, resulting in the identification of the most suitable ML model to be applied in PdM (Samatas et al., 2021). In the realm of Predictive Maintenance (PdM), this stage is commonly denoted as the maintenance decision-making phase. Its key aim is to identify the optimal algorithm for the PdM application. This strategic decision-making process ensures that the chosen ML model aligns seamlessly with the specific requirements and goals of the Predictive Maintenance system (Carvalho et al., 2019).

The availability and quality of datasets play a crucial role in the design of solutions for predictive maintenance problems. Appendix 1 highlights a significant pattern: there is a prevalence of non-available datasets compared to available ones, emphasizing a common challenge in the field. Artificial intelligence learning processes heavily rely on data that meets quantitative and qualitative criteria. In the context of PdM, the data is related to the equipment's operation, encompassing both normal functioning and error states. However, a challenge arises when there is an imbalance in the data, with a substantial amount of data from normal operation and limited data from faulty conditions. This issue, known as imbalanced data, is a common problem in real-world datasets and poses difficulties in training models effectively.

Imbalanced data results in an asymmetry between classes, wherein the minority class usually signifies the fault conditions. Addressing this imbalance necessitates the implementation of diverse strategies to guarantee that the model effectively captures the patterns and features of the minority class.

The issue of imbalanced data is particularly important in predictive maintenance, as accurate identification and prediction of faulty conditions are essential for effective maintenance strategies. Addressing the challenges posed by imbalanced data can lead to more robust and reliable predictive maintenance models, enhancing maintenance operations' overall performance and efficiency (Samatas et al., 2021).

1.2. Imbalanced Data in Predictive Maintenance: Solutions and Strategies

In recent years, there has been a significant surge in interest regarding the imbalanced learning problem among academia, industry, and government funding agencies. The core of this problem lies in the adverse impact that imbalanced data can have on the performance of standard learning algorithms. These algorithms are poorly adapted to handle complicated imbalanced datasets because they often assume balanced class distributions or equivalent misclassification costs. Consequently, they fail to accurately represent the underlying characteristics of the data, resulting in inaccurate results for various classifications (He & Garcia, 2009).

In classification tasks, when dealing with imbalanced data, one class is typically represented as the majority class due to its higher number of samples, while the remaining classes constitute the minority class. An important application of this scenario is anomaly detection, where the focus of interest lies in identifying rare and unusual behaviors, representing the minority class. Conventional machine learning algorithms can perform well for the majority class due to their abundance of samples, but they may struggle to provide accurate predictions for the minority class due to their limited representation. Addressing this challenge requires specialized techniques to improve the performance and accuracy of models in detecting and handling rare anomalies effectively (Rekha et al., 2021).

Addressing the real-world implications of the imbalanced learning problem is of great significance, highlighting it as a critical issue that merits additional exploration and attention. Ensuring more accurate representations and predictions in imbalanced datasets requires the development of effective approaches.

Addressing the challenge of class imbalance is crucial for achieving better identification rates for the minority class using conventional classification algorithms. Recently, there has been an increased focus on class imbalance classification studies, reflecting the growing demand to tackle this issue. Considerable effort is being invested in improving classification models to address the problem of data imbalance. Researchers have identified four key approaches to tackle this challenge and enhance the handling of imbalanced information in various contexts. By leveraging these diverse approaches, researchers and practitioners can effectively tackle the challenges posed by imbalanced datasets and enhance the performance of classification models

in various real-world applications. In the following sections, we will explore each approach to gain insights into its applications and effectiveness.

2.1.2 Data-level methods

These methods focus on modifying the training set to make it suitable for standard learning algorithms. One approach is to balance the distribution by either generating new instances for the minority class (over-sampling) or reducing instances from the majority class (under-sampling) (Rekha et al., 2021).

Sampling techniques are frequently used in imbalanced learning applications to correct unbalanced datasets and produce a more equal distribution. The objective is to enhance classifier accuracy by providing a fair representation of both majority and minority classes. Using balanced datasets has been shown to improve overall classification performance, justifying the adoption of sampling techniques in imbalanced learning (He & Garcia, 2009).

To address the issue of imbalanced data, various sampling methods come into play. Oversampling involves increasing instances in the minority class, while undersampling reduces the majority class. Both methods balance class distribution, aiding the model in recognizing patterns in the minority class. A hybrid approach combines oversampling and undersampling to strike a balance effectively.

- **Oversampling:** Oversampling is a method employed in machine learning to tackle class imbalance. It involves augmenting the number of instances in the minority class, often by duplicating or generating synthetic examples. This approach helps the model better learn patterns associated with the minority class. In our research, we will explore three oversampling methods: random oversampling, SMOTE, and ADASYN.
    - Random Oversampling (ROS): Random Oversampling is a non-heuristic approach that aims to balance class distribution by randomly replicating minority class examples. Nevertheless, the method raises concerns about potential overfitting owing to the exact duplication of minority examples. Moreover, its applicability may be limited for large and imbalanced datasets, introducing computational challenges (Bennin et al., 2019).

- Synthetic Minority Over-sampling Technique (SMOTE): SMOTE produces additional instances of the minority class through interpolation between existing minority class instances situated closely in feature space. By doing so, SMOTE avoids the potential overfitting issues associated with randomly replicating minority class instances, as in the case of Random Oversampling (Ramyachitra & Manikandan, 2014; Lin et al., 2017). The algorithm randomly selects the k nearest neighbors for each training instance of the minority class and adds a new instance to the training set by selecting a random point along the line segment between the instance and its nearest neighbor. SMOTE is particularly effective for highly imbalanced datasets and can be further improved by combining it with undersampling techniques (Tantithamthavorn et al., 2020).
- Adaptive Synthetic Sampling (ADASYN): ADASYN is a notable extension of SMOTE, developed by Haibo et al., that effectively addresses class imbalance. ADASYN takes a distinctive approach by generating synthetic data for minority class samples based on their density distribution. The core concept is to concentrate on generating additional synthetic data points for minority instances that pose a greater learning challenge compared to those that are relatively easier to learn. The dual objectives of ADASYN are to mitigate bias and facilitate adaptive learning. Through the application of ADASYN, both researchers and practitioners can significantly improve the efficacy of machine learning models when dealing with imbalanced datasets (Kulkarni et al., 2020).

- **Undersampling:** Undersampling is used when the proportion of majority class examples is much higher than that of the minority class. Unlike oversampling, undersampling involves removing some of the majority class examples to achieve a balanced distribution between the minority and majority classes (Kulkarni et al., 2020; Rathore et al., 2022). Undersampling can be done using a variety of methods, each with its own strengths and weaknesses. Here, we focus on four of the most used undersampling techniques: Random Undersampling (RUS), Tomek Links, ENN, and Cluster Centroids, which have

been shown to effectively reduce bias towards the majority class while maintaining the information content of the minority class.

- Random Undersampling (RUS): Random Undersampling is a popular, easy-to-implement method for balancing imbalanced datasets. It involves randomly selecting and removing instances from the majority class until the dataset is balanced, with an equal number of positive and negative class examples (Ramyachitra & Manikandan, 2014). While this technique is simple and computationally efficient, it can result in the loss of potentially useful information, which could impact the classification performance **(**Kotsiantis et al., 2005). Nevertheless, RUS can prove beneficial for datasets with sufficient examples in the minority class, where a model can be effectively fit.
- Tomek Link is an effective heuristic undersampling technique that tackles class imbalance through a distance-based approach. It establishes links between instances from different classes based on their distances. The primary objective is to identify pairs of instances (A_i, A_j) with a Tomek Link, which occurs when two samples belonging to different classes, denoted as A_i and A_j, have a distance measure $d(A_i, A_j)$, and there is no other sample Ak for which either $d(A_i, A_k) < d(A_i, A_j)$ or $d(A_j, A_k) < d(A_i, A_j)$ (Pereira, 2020). In simpler terms, a Tomek Link is recognized when no other sample is closer to either A_i or A_j than they are to each other. By identifying these Tomek Links, the technique aims to identify instances from the majority class that are close to instances from the minority class. Subsequently, the instances belonging to the majority class are removed from the dataset, retaining only the instances from the minority class. This process helps in improving class separation and achieving a more balanced dataset for training machine learning models, ultimately enhancing performance on imbalanced datasets (Kulkarni et al., 2020).
- Edited Nearest Neighbor (ENN): Stands out as a sophisticated undersampling technique pivotal in mitigating imbalances within machine learning datasets. Going beyond the conventional randomness of undersampling, ENN meticulously examines the nearest neighbors

of each majority-class sample, scrutinizing label consistency and surgically eliminating potential noise (Xu et al., 2020). This intricate methodology markedly boosts classification accuracy, notably fortifying the handling of minority samples. In the landscape of imbalanced data classification, ENN emerges as a powerful and refined approach, paving the way for more accurate and robust model outcomes.

    - Cluster Centroids: This clustering analysis-based approach has been proposed to address the limitations of Random Undersampling. Clustering is a technique designed to group similar data samples into clusters, where objects within each cluster share common feature representations. When employing clustering to undersample the majority class, the process involves generating a set of clusters, each encapsulating similar data. The k-means algorithm is used to calculate the mean of similar data in the same group, which becomes the cluster centroid. These centroids are used to represent the data in the whole group, replacing the original data and reducing the size of the majority class. This clustering-based undersampling strategy has been shown to reduce the risk of removing useful data from the majority class and can lead to better performance of constructed classifiers, including both single classifiers and classifier ensembles, compared with random undersampling (Lin et al., 2017).

- **Hybrid sampling approaches:** The hybrid approach for handling class imbalance is a combination of oversampling and undersampling techniques strategically applied to strike a balance between the classes. In contrast to pure oversampling or undersampling techniques, the hybrid method aims to capitalize on the benefits of both approaches while mitigating their individual limitations.

Oversampling and undersampling techniques have been applied in various fields to address the issue of imbalanced datasets. Japkowicz (2000) experimented with undersampling and resampling strategies on artificial 1D data. Ling and Li (1998) combined the oversampling of the minority class with the undersampling of the majority class, focusing on marketing analysis. Solberg and Solberg (1996) used

oversampling and undersampling to improve oil slick classification from SAR imagery.

The imbalanced learning domain has seen the development of numerous effective oversampling approaches, with the synthetic minority oversampling technique (SMOTE) standing out as the most renowned one. SMOTE has gained widespread popularity due to its effectiveness in addressing class imbalance. Over time, the method has been subject to significant research and refinement, leading to the emergence of over 90 SMOTE extensions published in various scientific journals and conferences (Fernández et al., 2018).

In the domain of Business Management, the Customer Relationship Management (CRM) field benefited from the use of these techniques for Customer Churn Prediction (Geiler et al., 2019). In the Information Technology domain, oversampling and undersampling were applied to Network Analysis and Mobile Malware Detection (Fernández et al., 2018). The Engineering sector leveraged these techniques for Fault Detection in Semiconductors (Nuhu et al., 2022). In the field of Medicine, multiple applications were observed, including Quality Control and Prediction of Survival Status of poly-trauma patients (He et al., 2023).

2.1.3 Algorithm-level methods

Algorithm-level methods encompass adaptations of existing learning algorithms designed to minimize biases against imbalanced data and tailor them for handling skewed data distributions. This category often employs cost-sensitive strategies, which involve modifying the learning algorithms to assign different costs or penalties to misclassifications of minority and majority class instances. These adjustments aim to improve the model's sensitivity to the minority class, thereby enhancing its overall performance on imbalanced datasets. Algorithm-level methods offer a diverse set of techniques to address class imbalances, ranging from algorithmic modifications to the introduction of specialized cost functions.

2.1.4 Hybrid approaches

Hybrid methods combine the strengths of data-level and algorithm-level methods. They aim to mitigate the weaknesses of each individual approach, resulting in a more robust solution.

2.1.5 Ensemble methods

Ensemble algorithms are employed to train models on imbalanced data. These methods utilize multiple classifiers to make predictions, which can improve the overall performance and robustness in dealing with class imbalance.

3. METHODOLOGY

In our comprehensive approach to evaluating classification performance within the realm of predictive maintenance, we leverage the power of two distinct datasets, each presenting its unique challenges. The first dataset is a synthetic construct, the AI4I 2020 Predictive Maintenance Dataset (Matzka, 2020), intentionally designed to offer a controlled environment for experimentation, albeit with a limited number of instances. Our second dataset hails from the real world—the PHM Data Challenge 2018 (Bonatakis et al., 2018) —a dataset teeming with many examples but plagued by a pronounced class imbalance. This real-world dataset is where the class imbalance issue becomes notably more severe, reflecting the complexities often encountered in practical applications of predictive maintenance.

Class imbalance in predictive maintenance is paramount, as it mirrors the challenges encountered in actual industrial scenarios, where the majority of data points represent normal machine operation, and rare failure events must be accurately detected. While we use synthetic and real-world datasets, the methodology's applicability to practical predictive maintenance scenarios is underscored by its consistent application to both, reflecting the real-world challenges we aim to address.

As depicted in Figure 3.1, our methodology initiates with the dataset, followed by the preprocessing phase, encompassing essential tasks such as Data Cleaning, Normalization, and Data Encoding. It is important to note that this process is consistently applied to both synthetic and real-world datasets. The dataset is thoughtfully divided into training and test subsets, with 80% allocated for training and 20% for testing, ensuring robust model evaluation. To address the class imbalance, Figure 3.1 highlights the utilization of oversampling, undersampling, and hybrid sampling techniques applied to the training data for both datasets. Moving forward, models are created from the balanced training data using the Weka toolkit, offering a diverse array of classification algorithms. Finally, as illustrated in the diagram, the models are employed to predict outcomes on the test data, yielding Classification Results that illuminate the models' effectiveness in the context of predictive

maintenance tasks for both datasets. This comprehensive methodology guides our evaluation process, consistently applied to both datasets, encompassing the entire journey from data preprocessing to result analysis.

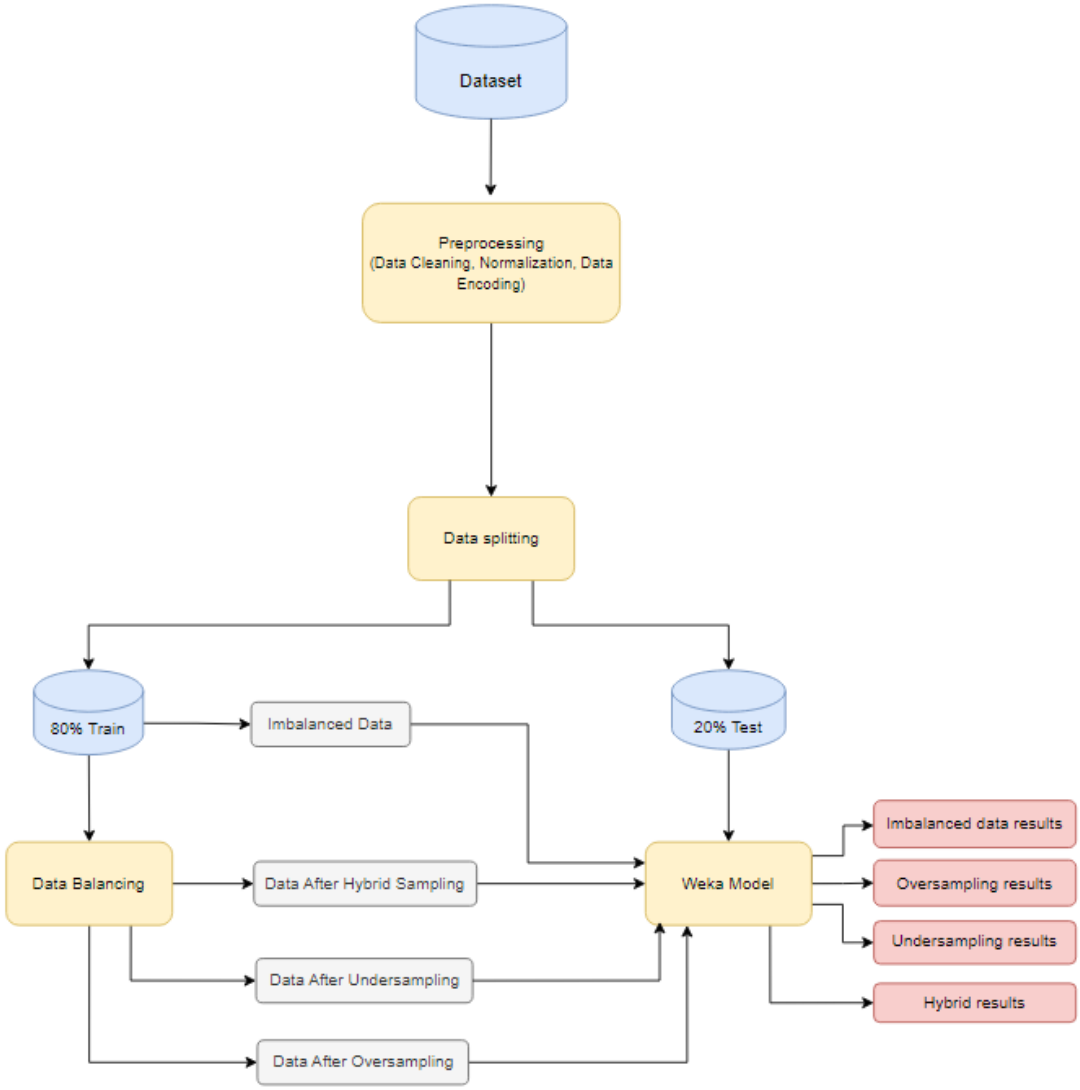


Figure 3.1. Sequential Methodology for Evaluating Classification Performance.

Source: Author

3.1. Datasets

In this study, we employ two publicly available datasets, each characterized by significant class imbalance, to underpin our investigation: the AI4I 2020 Predictive Maintenance Dataset (Matzka, 2020) and the PHM Data Challenge 2018 (Bonatakis

et al., 2018). These datasets were chosen to encompass both controlled synthetic scenarios and real-world industrial complexities.

The AI4I 2020 Predictive Maintenance Dataset has been thoughtfully curated to replicate maintenance data typically encountered in industrial settings. This benchmark dataset delineates five distinct types of machine failures, each characterized by specific configurations of six critical features. A total of 10,000 samples have been meticulously recorded, with 339 instances representing machine failures.

Conversely, the PHM Data Challenge 2018 dataset represents real-world industrial conditions captured from 16 ion mill etch tools integral to the wafer production process. Within this vast dataset, our focus narrows to three specific tools—identified as 01M02, 03M02, and 08M01—each meticulously selected to serve dual purposes. Firstly, this selection effectively mitigates computational complexities arising from the dataset's considerable size. Secondly, it ensures the comprehensive evaluation of our methodology, as it enables us to observe all types of machine failures in both the training and testing phases. Of note, the dataset derived from these three chosen tools is notably voluminous, comprising an extensive 13,887,132 samples, of which 80 represent machine failures.

Table 3.1. provides a comprehensive overview of the dataset features and the distinct types of machine failures contained within both datasets, facilitating a comprehensive understanding of their composition and significance for our study.

The datasets underwent a meticulous preprocessing phase to ensure the quality and compatibility of the data for subsequent analysis. This multifaceted process encompassed several critical steps. As part of this, we selectively removed certain features, such as time, stage, Lot, runnum, recipe, and recipe_step, from the real-world dataset to streamline the analysis. This decision was based on the understanding that these features, being constant values, did not contribute meaningful variation and were unlikely to affect the results significantly. Data cleaning procedures were rigorously applied to identify and rectify any missing or erroneous data points. Normalization methods were employed to standardize the feature scales, ensuring uniformity, and avoiding any single feature from exerting undue influence on the analysis because of its magnitude.

Table 3.1. List of Features and types of machine learning failures

Dataset	Features	Machine Failures
AI4I 2020 Predictive Maintenance Dataset	Product ID, air temperature, process temperature, rotation speed, torque, tool wear	Tool wear failure (TWF), heat dissipation failure (HDF), power failure (PWF), overstrain failure (OSF), and None.
PHM Data Challenge 2018	Time, stage, Lot, runnum, recipe, recipe_step, FIXTURESHUTTERPOSITION, IONGAUGEPRESSURE, ETCHBEAMVOLTAGE, ETCHBEAMCURRENT, ETCHSUPPRESSORVOLTAGE, ETCHSUPPRESSORCURRENT, FLOWCOOLFLOWRATE, FLOWCOOLPRESSURE, ETCHGASCHANNEL1READBACK, ETCHPBNGASREADBACK, FIXTURETILTANGLE, ROTATIONSPEED, ACTUALROTATIONANGLE, ETCHSOURCEUSAGE, ETCHAUXSOURCETIMER, ETCHAUX2SOURCETIMER, ACTUALSTEPDURATION, class	FlowCool Pressure Dropped Below Limit, Flowcool Pressure Too High Chech Flowcool Pump, Flowcool leak and None.

Source: Author

Additionally, data encoding was implemented to transform categorical data into a structure compatible with the requirements of machine learning algorithms. Notably, this encoding involved converting numerical values into string representations, enabling compatibility with the Weka toolkit, a crucial step for accurate analysis of class errors. Furthermore, to facilitate Weka's analysis, the data was meticulously

ordered based on the 'class' attribute, ensuring that the dataset was structured for seamless utilization within the toolkit. These preprocessing steps, as depicted in Figure 3.1, were consistently applied to both datasets, synthetic and real-world, thereby ensuring uniform data quality and compatibility throughout the study.

3.2. Sampling Techniques

Our study is dedicated to providing effective solutions for mitigating the inherent class imbalance found in our datasets. To tackle this issue, we harness both oversampling and undersampling techniques. These methods are pivotal in reshaping our datasets, ensuring a balanced distribution of classes by either increasing or decreasing the number of instances within each class.

In our data analysis, we will apply seven key techniques for balancing data in the context of addressing class imbalance. We provide a step-by-step breakdown of how each technique is implemented in Python, utilizing the panda's library. These techniques include SMOTE, ADASYN, and ROS (Random Over-Sampling) for oversampling, RUS (Random Under-Sampling), Cluster Centroids, ENN (Edited Nearest Neighbor) and Tomek Links for balancing data through undersampling.

Moreover, our approach extends to hybrid methods, which combine oversampling and undersampling techniques. One approach involves oversampling the minority class, followed by undersampling the majority class. In the second approach, we will reverse the order, starting with undersampling and then oversampling. By comparing the results from both approaches on synthetic and real-world datasets, we aim to understand the impact of hybrid sampling on model performance and identify the most effective approach for our specific tasks. We will explore these combinations, as they have the potential to provide thorough solutions to the class imbalance challenge within our dataset. The combinations under investigation include ADASYN with Tomek Links, ADASYN with Cluster Centroids, SMOTE with Tomek Links, SMOTE with Cluster Centroids, SMOTE-ENN (built-in), Tomek Links with ADASYN, and Cluster Centroids with ADASYN. It's important to note that SMOTE-ENN is essentially SMOTE-ENN, a built-in combination that automatically incorporates both methods without the need for separate application. This holistic approach enables us to tackle class imbalance comprehensively within our dataset.

Now, let's include a code snippet that illustrates the application of the SMOTE technique, as depicted in Code 3.1.

```
# Code for SMOTE (Oversampling)
def smote(data, strategy):
    x = replace_tool_string(data.drop('class', axis=1))
    y = data['class']
    x_res, y_res = SMOTE(sampling_strategy=strategy).fit_resample(x, y)
    sampled_data = pd.concat([replace_tool_number(x_res), y_res], axis=1)
    return sampled_data
```

Code 3.1. Code for SMOTE technique.
Source: Author

The code snippet in Code 3.1. demonstrates the application of the Synthetic Minority Oversampling Technique (SMOTE) to address the class imbalance in our dataset. The process commences with data preparation, separating the feature data (x) and the class labels (y). With the data ready, SMOTE is introduced to perform oversampling of the minority class.

SMOTE's primary function is to generate synthetic samples for the minority class, effectively balancing the dataset. It achieves this by interpolating between the existing minority class samples, thus creating a more equitable distribution of class instances.

The result of the SMOTE operation yields resampled feature data (x_res) and the corresponding class labels (y_res), which are then concatenated to construct a new dataset. This newly formed dataset is characterized by an improved balance between class instances and is ready for use in machine learning models, ensuring more robust and accurate predictions.

This detailed code breakdown offers insight into how SMOTE addresses class imbalance and generates a more balanced dataset for subsequent analysis and modeling. Similar code structures are applied to other techniques, including ADASYN, ROS, RUS, Cluster Centroids, ENN and Tomek Links.

In Code 3.2, we find the sample function, a versatile tool for applying resampling techniques to address class imbalance within a dataset. It simplifies the process by

allowing the selection and application of specific resampling methods, such as SMOTE, ADASYN, ClusterCentroids, TomekLinks, ENN, ROS, and RUS.

The sample function streamlines the task of systematically applying the chosen resampling technique, providing a straightforward means to address class imbalance in machine learning projects.

```python
def sample(data, sampling_name, sampling_strategy=None):
    # Choose the resampling method based on the sampling_name
    match sampling_name:
        case 'SMOTE':
            return smote(data, sampling_strategy)
        case 'ADASYN':
            return adasyn(data, sampling_strategy)
        case 'ClusterCentroids':
            return cluster_centroids(data, sampling_strategy)
        case 'TomekLinks':
            return tomek_links(data, sampling_strategy)
        case 'ENN':
            return edited_nearest_neighbors(data, sampling_strategy)
        case 'ROS':
            return random_over_sampling(data, sampling_strategy)
        case 'RUS':
            return random_under_sampling(data, sampling_strategy)
```

Code 3.2. Code for selecting the resampling method.

Source: Author

3.3. Weka Classifiers

The Weka toolkit provides classifiers categorized into seven groups: Bayes, functions, lazy, meta, misc, rules, and trees. These categories encompass a wide range of classification algorithms, totaling 42 different options, as presented in Table 3.2.

For our study, we comprehensively evaluated all these classification algorithms on both imbalanced datasets, including synthetic and real-world datasets. Our assessment aimed to identify the best-performing classifier for each category within each dataset. These top-performing classifiers were subsequently employed for the classification of balanced datasets. Notably, all experiments were conducted using Weka's default parameter values.

Table 3.2. Weka Classifiers

Classifiers	Classification Algorithms
Bayes	BayesNet, NaiveBayes, NaiveBayesMultinominal, NaiveBayesMultinominalText, NaiveBayesMultinominalUpdateable, NaiveBayesUpdateable
Functions	Logistic, MultilayerPerceptron, SimpleLogistic, SMO
Lazy	IBk, Kstar, LWL
Meta	AdaBoostM1, Bagging, ClassificationViaRegression, CVParameterSelection, FilteredClassification, LogitBoost, MultiBoostAB, MultiClassClassifier, MultiClassClassifierUpdateable, MultiScheme, RandomCommittee, RandomizableFilteredClassifier, RandomSubSpace, Stacking, Vote, WeightedInstancesHandlerWrapper
Misc	InputMappedClassifier
Rules	DecisionTable, Jrip, OneR, PART, ZeroR
Trees	DecisionStump, J48, LMT, RandomForest, RandomTree, RepTree

Source: Author

3.4. Evaluation Metrics

Evaluation metrics are instrumental in assessing the performance of classifiers. Their roles are threefold: First, they quantify a classifier's correctness when applied to previously unseen data. Secondly, they aid in selecting the most promising classifiers for optimal future performance with unseen data. Thirdly, these metrics are tools for distinguishing and selecting the most suitable solution during classifier training. For instance, accuracy is frequently employed to pinpoint the best solution generated by a classification algorithm. However, it's essential to acknowledge that accuracy is limited, particularly in scenarios characterized by imbalanced class distributions. In such cases, alternative metrics come into play, helping to identify superior solutions and construct more effective classifiers. These metrics are paramount for enhancing

classifier generalization when confronted with previously unseen data, contributing to developing robust and precise classification systems (Hossin & Sulaiman, 2015).

In the domain of model evaluation, the selection of appropriate metrics is pivotal for assessing the performance of predictive models. Accuracy, a commonly employed metric, quantifies the total instances correctly predicted when tested with data that has not been previously observed (Hossin & Sulaiman, 2015). This assessment is quantified using the formula, as we can see in the formulation (1) (Singh, 2020): Accuracy = (True Positives + True Negatives) / Total Predictions, where Total Predictions encompasses True Positives (correct positive predictions), True Negatives (correct negative predictions), False Positives (incorrect positive predictions), and False Negatives (incorrect negative predictions).

$$Accuracy = \frac{TP + TN}{Total\ Predictions} \quad (1)$$

However, the suitability of accuracy as a metric depends on the specifics of the problem under consideration. In our case, where the class imbalance is a significant factor, and the consequences of false negatives are critical, prioritizing recall over accuracy is paramount.

Predictive maintenance, often dealing with scenarios where the consequences of missing positive instances, such as machine failures or defects, can be significantly more detrimental than falsely identifying normal conditions, benefits immensely from a focus on recall. Maximizing recall, which measures the model's ability to capture all positive cases, becomes essential in such cases.

Consider a predictive maintenance system used in an industrial setting, where the primary objective is to identify machine failures. Missing a true positive, which means failing to detect a machine failure, can lead to costly breakdowns, downtime, and potential safety hazards. In contrast, false positives, which involve identifying a failure when none exists, might result in temporary interruptions but are usually less critical.

By prioritizing recall, we increase the chances of identifying true machine failures accurately, reducing the risk of costly unplanned downtime and potential safety issues. It allows us to proactively address maintenance needs, replacing or repairing equipment before they lead to severe problems. In this context, a higher recall rate

ensures that fewer actual machine failures are missed, enhancing the overall efficiency and reliability of the maintenance process.

Recall, recognized as sensitivity or the true positive rate, gauges the model's capacity to accurately identify instances belonging to the positive class from the true positive instances. It is calculated using the formula we can see in formulation (2) (Singh, 2020), where the number of True Positives (instances correctly identified as positive) is divided by the sum of True Positives and False Negatives (positive instances that were incorrectly classified as negative) (Hossin & Sulaiman, 2015). In simpler terms, recall measures a model's ability to correctly capture all actual positive cases, indicating how effectively it identifies the positive class. It's particularly valuable in scenarios where missing positive instances carries significant consequences.

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

For our predictive maintenance applications, recall will be the central metric we employ to assess model performance, given its aptness for addressing class imbalance and the severe implications of false negatives.

4. EXPERIMENTS

In this comprehensive exploration of experimental procedures and methodology, our primary goal is to carefully assess a variety of classification algorithms. We're investigating how well they perform across different datasets — ranging from imbalanced and balanced synthetic data to real-world datasets. As detailed in our methodology, this multifaceted evaluation unfolds in a structured manner, initiating with the application of 42 distinct Weka classifiers to the imbalanced synthetic dataset, as demonstrated in Table 3.2. This strategic approach allows us to systematically explore and analyze the performance of each classifier, providing comprehensive insights into their effectiveness on imbalanced data. The detailed results and comparisons, as presented in subsequent sections, shed light on the nuanced performance variations among these classifiers, contributing to a deeper understanding of their capabilities and limitations in addressing class imbalances.

In tandem with our systematic approach, we employ an 80/20 data split, designating 80% of the dataset for training and reserving the remaining 20% for testing. This initial phase is crucial to our comprehensive analysis, ensuring a robust classifier performance evaluation. The training set, comprising a substantial portion of the data, facilitates the development of models that capture underlying patterns. In contrast, the testing set serves as an independent benchmark to assess the generalization capabilities of these models.

Subsequently, we move to a balanced version of the synthetic dataset and rerun the same classifiers. This dual-phase approach enables a comprehensive investigation into the impact of class balancing on the performance of various algorithms. This rigorous procedure is consistently replicated in our analysis of real-world datasets, ensuring methodological coherence, and facilitating meaningful comparisons.

To enhance clarity in the result presentation, we organize the outcomes before and after balancing in tabulated form. Furthermore, we discern and select classifiers demonstrating the most promising outcomes on imbalanced data to optimize our

analysis. These selected classifiers are then applied to the newly balanced dataset, mindful of the inherent complexities and time constraints associated with the experimentation process.

In alignment with our outlined methodology, we are dedicated to a thorough examination of hybrid sampling methods. These strategic techniques, involving the judicious combination of oversampling and undersampling, hold substantial promise in alleviating challenges associated with imbalanced datasets. Our objective extends to scrutinizing their efficacy across synthetic and real-world datasets, aiming to provide nuanced insights into their performance and broader applicability. This deliberate approach aims to highlight potential variations in performance across different data domains, providing a holistic understanding of the applicability of these methods. Additionally, we seek to delve into the impact of class imbalance on recall values, recognizing its significance in evaluating the practical utility of these hybrid techniques.

Through this multifaceted exploration, we aim to contribute meaningful insights into the nuanced dynamics of imbalanced dataset analysis, fostering a deeper understanding of the implications and outcomes of hybrid sampling methods. Through this meticulous approach, we strive to contribute valuable knowledge to the field of imbalanced dataset analysis.

4.1. Data Splitting for Evaluation

In this subsection, our focus shifts to a critical aspect of our evaluation process: the data-splitting procedure. Our main objective is to construct two datasets precisely, maintaining an even distribution for both training (which encompasses 80% of the data) and testing (representing 20% of the data) while upholding class balance. This careful approach is pivotal for the comprehensive evaluation of classification algorithms, as it helps us gauge their performance across diverse datasets.

```python
def separate_to_80_20(data):
    class_labels = ['FL', 'FPDBL', 'FPTHCFP', 'NONE']
    train_data, test_data = [], []

    for label in class_labels:
        class_data = data[data['class'] == label]
        part1, part2 = get_percent(class_data, 0.2)
        train_data.append(part2)  # 80% for training
        test_data.append(part1)   # 20% for testing

    # Concatenate data for both training and testing
    train_set = pd.concat(train_data)
    test_set = pd.concat(test_data)

    return train_set, test_set

def get_percent(df, percent):
    part1 = df.sample(frac=percent)
    part2 = df.drop(part1.index)
    return part1, part2
```

Code 4.1. Code for 80/20 Data Split.

Source: Author

The provided Code 4.1. serves as the backbone of our data-splitting process. It guarantees the creation of two distinct datasets – one dedicated to training and the other to testing – each maintaining an equal distribution of class labels. This intentional balance is a crucial step in our experimental journey, laying the groundwork for a nuanced and thorough analysis of the effectiveness of our classification algorithms.

In adhering to this rigorous data-splitting process, we pave the way for a more in-depth understanding of how our models perform under varied conditions, setting the stage for meaningful insights into classification algorithm behavior.

4.2. Understanding Our Experimental Phases

With our data thoughtfully divided into training and testing sets following an 80/20 split, we proceed to the crucial process of creating models and assessing their classification performance. In the pursuit of transparency, we aim to provide a step-by-step walkthrough for constructing models in Weka. To maintain clarity, we will guide you through this process for just one of the classifiers, emphasizing that the process remains consistent across all 42 classifiers. Our approach unfolds in two phases: first, we apply these models to both the imbalanced synthetic and real-world datasets, comprehensively evaluating their performance as mentioned earlier. Then, following the balancing of data, we select the classifiers that yield the most promising results and deploy them for a second round of assessment. These models will be applied to the balanced datasets, ensuring a holistic comparison of their classification capabilities. Our commitment to transparency and methodological rigor guides this exploration of the model creation and evaluation process.

To commence our experimental procedures, we begin by meticulously preparing the data for our classification task, focusing on the detection of machine failures, as outlined in Figure 3.1. The foundation of our approach is rooted in the utilization of an original, albeit imbalanced, synthetic dataset. In our pursuit of a robust and reliable methodology, we employ a systematic partitioning strategy. Specifically, we randomly allocate 80% of the instances from each class in the dataset to the training set while reserving the remaining 20% for the testing set. This strategic division ensures that both the training and testing datasets encompass a representative sample of examples from all fault types, mitigating potential biases and guaranteeing a comprehensive evaluation.

Moving forward, our analysis extends beyond the synthetic dataset to include real-world datasets, introducing an additional layer of complexity and authenticity to our experimental framework. The careful curation of these datasets involves addressing challenges associated with class imbalances, limited anomaly instances, and computational constraints. The exploration of both synthetic and real-world datasets enriches the scope of our investigation, allowing for a nuanced understanding of the performance of classification algorithms under diverse conditions.

Our next step involves feeding the training data into each of the 42 distinct Weka classifiers, a diverse set of machine learning algorithms renowned for their efficacy in classification tasks. Each classifier, when presented with the training data, generates a model that is subsequently employed to evaluate the testing dataset. Our key metric of interest is the average recall value across classes, which offers insights into how effectively the models classify examples within the minority classes.

This phase of our experimentation process serves as an initial benchmark, revealing the classification performance on our imbalanced dataset. It is important to note that we conducted these experiments on the imbalanced dataset as a preliminary step to gauge the initial results. Subsequently, our plan involves creating a balanced dataset and comparing the model performance against this balanced data. This comparative analysis will allow us to discern the impact of class balancing on classification accuracy and provide a comprehensive view of our classification approach.

Now, let's delve into the intricacies of our process, starting with a comprehensive guide on utilizing Weka, selecting the right dataset, and implementing common procedures for building models within the Weka framework. This detailed walkthrough is crafted to help replicate and understand our approach thoroughly.

4.3. Weka Version Selection and Data Loading

In our methodology, the selection of Weka version 3.8.6 is underpinned by its proven compatibility and stability. The data exploration begins with loading datasets through the "Applications -> Explorer" pathway. As illustrated in Figure 4.1, this pathway provides a user-friendly interface, facilitating seamless navigation through the initial stages of our experimentation. Given that our datasets are initially in CSV format, a pivotal step involves converting them to ARFF format to adhere to Weka's requirements. This crucial conversion lays the groundwork for a comprehensive exploration, paving the way for subsequent in-depth analysis and model creation. In Weka, this conversion can be achieved using the "CSVLoader" tool, which allows seamless transformation of CSV files into ARFF format. This process facilitates a smooth transition for our data, ensuring compatibility with Weka's suite of tools and enabling robust analysis.

Figure 4.1. Weka Home Screen Menu.

Source: Author

4.4. Visualizing Data in Weka

Before delving further into our methodology, it is pertinent to reference Figure 4.2, which offers a visual guide on how Weka presents a CSV file. This figure provides valuable insights, with the left section displaying attribute columns and the right side featuring a graphical representation depicting distinct class types within the synthetic data. This visual reference enriches our understanding of the dataset structure and sets the stage for comprehensively exploring our data within the Weka environment. Above this graphical presentation, pertinent details such as the names of each class, their respective counts, and individual weights are thoughtfully provided. This starting phase establishes the groundwork for a systematic and user-friendly exploration of our datasets within the Weka environment.

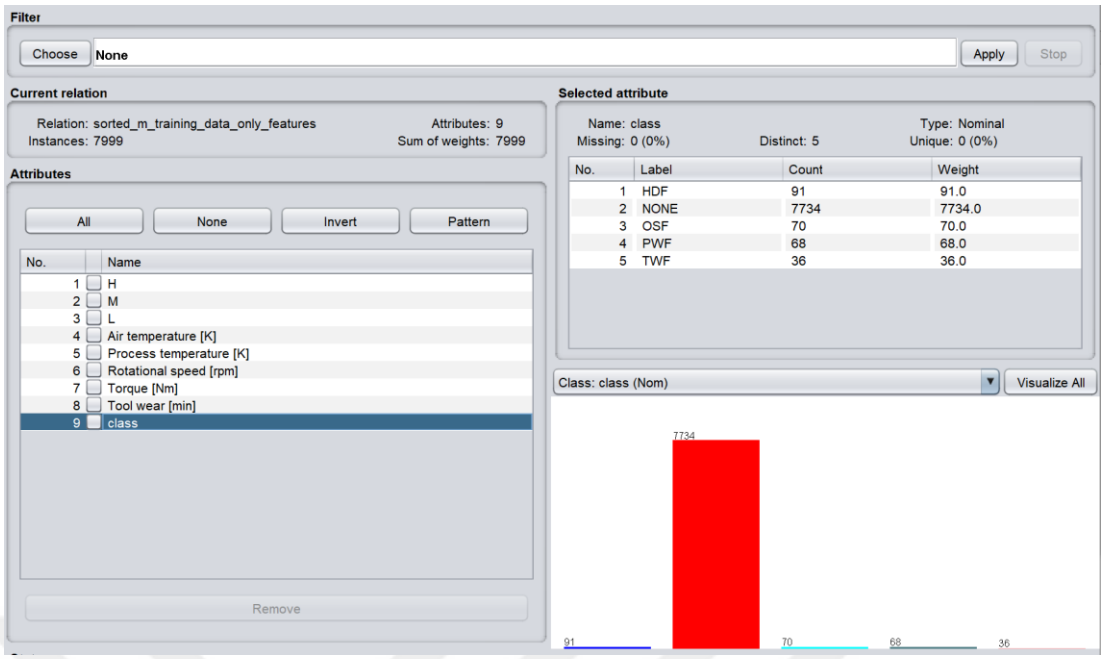


Figure 4.2. Weka CSV File Presentation.

Source: Author

4.5. Classifier Selection and CLI Preference in Weka

Within the Weka environment, classifiers can be chosen directly from the Filter panel for model creation. However, we prefer utilizing the Command Line Interface (CLI) to create models for our specific methodology. This preference arises due to the convenience of selecting both the train and test data simultaneously, streamlining the experimentation process for enhanced efficiency.

We've opted for the "Applications -> Simple CLI" pathway to facilitate ease of use and offer clear, step-by-step explanations. Our choice is driven by several compelling reasons, with efficiency being a key factor. Given the large datasets involved in our analysis, it's paramount to have a resource-efficient approach that doesn't burden the system. The Simple CLI aligns with this need, streamlining the process and allowing users to interact with Weka without consuming excessive system resources. This efficiency ensures a smooth and responsive experience, minimizing the time and effort required to open and work with datasets in Weka. Moreover, the Simple CLI's straightforward nature enhances the clarity of the process, aiding in the reproducibility of our methodology. This is exemplified in Figure 4.1, which showcases the Weka home screen and highlights the menu feature that grants access to this option.

4.6. Training and Evaluation for Imbalanced Data

The command exemplified in Figure 4.3 outlines a structured process within our methodology. In this command, we utilize the Weka Simple CLI to execute the JRip classifier. The initial segment, "java.weka.classifiers.rules.JRip," specifies the classifier of choice. The subsequent components of the command are designed to address distinct aspects of the procedure. The "-t" flag, denoting the training dataset, directs the command to the "train80.arff" dataset, which contains the instances required for the classifier's training. Including the "-d" flag defines the storage location for our "JRip.model" generated during the training process. It's important to note that while we provide this command as an example, we will execute similar commands for other classifiers, ensuring a consistent and replicable approach throughout our methodology.

```
java weka.classifiers.rules.JRip -t C:\dataset\train80.arff -d C:\models\Jrip.model
```

Figure 4.3. Command for Creation Jrip Model Using Weka CLI.

Source: Author

Once we have generated a JRip model, the next step involves evaluating its performance. The command shown in Figure 4.4 represents the process we follow to assess the performance of our JRip model. This command evaluates the model's efficiency on a separate dataset reserved for testing.

```
java weka.classifiers.rules.JRip -l C:\models\Jrip.model -T C:\dataset\normalized_test_20.arff
```

Figure 4.4. JRip Model Evaluation Command.

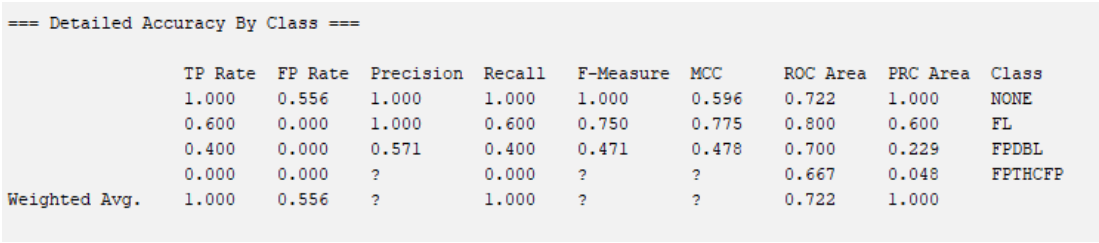
Source: Author

The "-l" flag is used to load a pre-existing model, and in this specific command, it refers to the "Jrip.model" file. This flag instructs the JRip classifier to load the previously trained model stored in the "Jrip.model" for evaluation on new data. The "-T" flag, on the other hand, designates a testing dataset, "normalized_test_20.arff" in

this context. It prompts the classifier to apply its rule-based decision logic to this separate dataset to assess its performance. It measures how well the JRip model generalizes its learned rules to make predictions on previously unseen data, providing valuable insights into its real-world effectiveness.

Following the evaluation of our JRip model, we are poised to delve into the results, shedding light on its performance. The forthcoming Figure 4.5 encapsulates the model's recall values and other related metrics, providing a comprehensive overview of its performance. Recall is a vital metric that measures the model's ability to correctly identify positive instances. In the context of our research, it plays a pivotal role in assessing the model's proficiency in detecting critical events or anomalies, making it a central point of interest.

The recall values presented in Figure 4.5 will provide an in-depth understanding of the model's effectiveness in capturing true positive cases. It offers insights into the model's ability to minimize false negatives, a critical factor in scenarios where missing positive instances carry significant consequences, as in predictive maintenance. By meticulously examining these recall values, we can gain a comprehensive view of the JRip model's strengths and areas for improvement, paving the way for informed decision-making and model optimization.



=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	1.000	0.556	1.000	1.000	1.000	0.596	0.722	1.000	NONE
	0.600	0.000	1.000	0.600	0.750	0.775	0.800	0.600	FL
	0.400	0.000	0.571	0.400	0.471	0.478	0.700	0.229	FPDBL
	0.000	0.000	?	0.000	?	?	0.667	0.048	FPTHCFP
Weighted Avg.	1.000	0.556	?	1.000	?	?	0.722	1.000	

Figure 4.5. JRip Model Recall Performance.

Source: Author

In Figure 4.5, the results illustrate the recall performance of the JRip classifier on real-world data before applying dataset balancing techniques, with an average recall value of 0.5.

The obtained recall result of 0.5 can be expected due to the highly imbalanced nature of our dataset. It is noteworthy to mention that the recall for the “NONE” class is 1, indicating that all instances of this class were correctly predicted. On the other hand, the recall for the “FPTHCFP” class is lower, indicating that none of its instances were predicted, highlighting the impact of class imbalance on the model's performance, especially for minority classes. In this real-world dataset, we have a vast majority of non-anomalous instances, approximately 11 million, in contrast to a relatively small number of anomalous cases, which includes very few instances of anomalous cases. With the majority class significantly outnumbering the minority classes, this extreme class imbalance poses a significant challenge for machine learning methods to create accurate models for the minority classes.

In such a skewed data distribution, where anomalies are significantly outnumbered by non-anomalous instances, achieving a recall of 0.5 is reasonable. This is because the classifier tends to perform well in identifying the abundant non-anomalous cases but might miss out on some rare anomalous instances due to their limited presence in the dataset. It's important to recognize that recall is particularly sensitive to the detection of rare events, and in this context, the rarity of anomalies contributes to the observed result.

Having detailed our approach for the JRip classifier and its performance on real-world data with a recall of 0.5, we will now extend our analysis to encompass all 42 classifiers. Our comprehensive evaluation involves creating models for the same imbalanced real-world dataset and synthetic data, following the previously explained step-by-step procedures in Weka.

4.7. Analyzing Classifier Performance for Imbalanced Datasets

In our exploration of classifier performance, Figure 4.6. serves as a comprehensive overview of the results derived from classifying imbalanced datasets using all 42 classifiers. This analysis encompasses both synthetic and real-world data, presenting a nuanced comparison. The lighter-colored bar on the left corresponds to the synthetic dataset, while the darker-colored bar on the right represents the real-world dataset. Additionally, specific bars are marked with dashes, indicating the best-performing algorithms in each classifier category. Notably, BayesNet excels in the Bayes category for the synthetic dataset, while JRip leads in the Rules category for the real-world

dataset. These top-performing classifiers will be further examined in experiments involving balanced datasets.

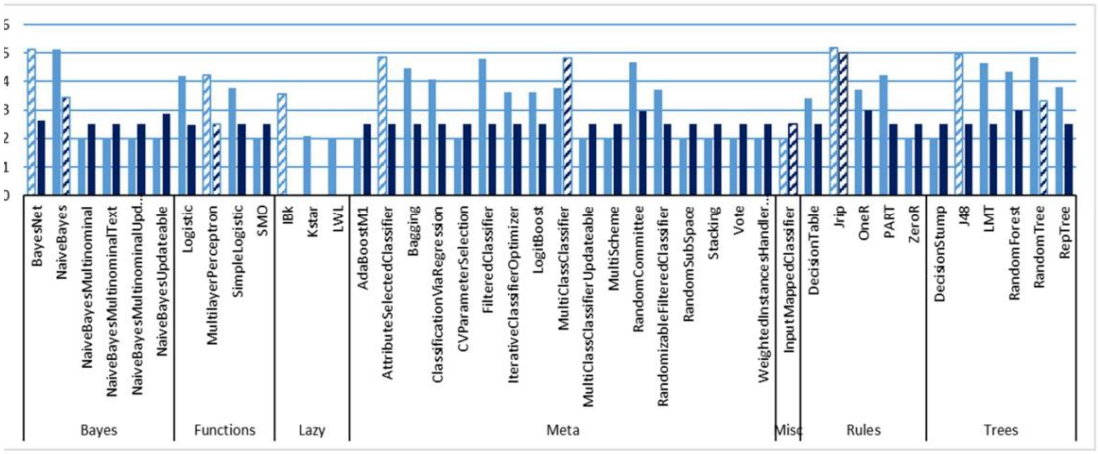


Figure 4.6. Classifiers Performance as Recall for Imbalanced Datasets.

Source: Author

As depicted in the figure, the majority of recall values are below 0.5, indicative of challenging classification scenarios. Many classifiers perform poorly, with recall values around 0.25 and as low as 0.2. The classification performance can be considered unsatisfactory both for the synthetic dataset with five classes and the real-world dataset with four classes. This is the direct consequence of the class imbalance. Since there are few examples of the minority class in the training data, the classification model cannot learn a model that adequately represents the minority classes. As a result, examples in the minority classes are mispredicted in the testing phase. As far as the classifier categories are concerned, one can say that the best-performing classifiers lie in the Trees set. On the other hand, the classifiers in the Lazy category are among the worst ones. The algorithms in this class are highly demanding in terms of computational complexity. Although we have performed our experiments on a powerful machine with 32 GB memory and the AMD Ryzen 7 1800X eight-core processor, IBk, Kstar, and LWL algorithms have crashed during our experiments with real-world data. That's why the recall values for these experiments are missing in the figure.

4.8. Training and Evaluation for Balanced Data

As we delve deeper into addressing the issue of data imbalance, we put our previously discussed sampling methodologies into action. The next phase involves the application of oversampling and undersampling methods, followed by the introduction of hybrid techniques, all in accordance with the methodology outlined earlier. The results of the previous section show that data imbalance leads to poor classification performance. Hence, we proceed with our experiments using balanced datasets obtained after implementing the sampling techniques.

In the previous section, we described the various sampling methods and provided code snippets to apply them, as outlined in our methodology. Now, let's put these methods into action in a real-world context. Our aim is simple: we want to use resampling techniques on our training dataset to create new datasets where the classes are more evenly balanced. This entails either boosting the representation of minority classes or lessening the dominance of majority classes. This step is crucial to tackle the data imbalance issue and enhance the performance of our classifiers.

After successfully generating these new balanced datasets through the application of oversampling and undersampling techniques, our next step is to apply the classifiers that demonstrated the best results in the previous section when dealing with the imbalanced data, as presented earlier in Figure 4.6. Specifically, we will utilize six classifiers: BayesNet, Multilayer Perceptron, IBk, AttributeSelected Classifier, Jrip, and J48. InputMappedClassifier has performed equally poorly in all the following experiments. That’s why we exclude it from our results.

As detailed in Table 4.1, we present the classification results for the imbalanced and balanced synthetic datasets. The values in the second column with the title “Recall for Imbalanced Data” correspond to the results in Figure 4.6. Here, the Jrip algorithm performs the best, with a recall of 52%. ROS, SMOTE, ADASYN, RUS, CLUSTER, TOMEK and ENN columns show the results of the classification experiments conducted with balanced datasets obtained after applying the relative resampling techniques. All the results with the balanced datasets through oversampling outperform those with the imbalanced dataset by around 30%.

Table 4.1. Recall Values for the Synthetic Dataset Before and After Applying the Resampling Methods

Classifier	Recall for Imbalanced Data	Recall After Oversampling			Recall after Undersampling			
		ROS	SMOTE	ADASYN	RUS	CLUSTER	TOMEK	ENN
BayesNet	0.51	0.81	0.82	0.79	0.45	0.77	0.56	0.79
Multilayer Perceptron	0.42	0.81	0.80	0.80	0.34	0.79	0.81	0.78
IBk	0.36	0.50	0.61	0.63	0.30	0.71	0.63	0.65
Attribute Selected Classifier	0.49	0.67	0.80	0.73	0.34	0.86	0.81	0.66
Jrip	**0.52**	0.69	0.77	0.74	0.38	**0.87**	0.77	0.64
J48	0.49	0.71	0.73	0.73	0.34	0.86	0.50	0.72

Source: Author

One can say that the dataset generated with SMOTE and ADASYN is superior to the one produced with ROS because SMOTE and ADASYN produce more representative examples of the minority classes. When comparing SMOTE and ADASYN, their performance appears quite similar on average, with SMOTE exhibiting a slight edge over ADASYN. The results of the undersampling methods are poles apart. The findings for the CLUSTER technique are even better than those for the SMOTE technique from time to time. On the other hand, RUS results fall behind the results for the imbalanced dataset. It's important to highlight that the performance of the ENN method is suboptimal compared to TOMEK for specific algorithms. However, when assessing overall performance, ENN generally demonstrates superiority over TOMEK. Despite this, it consistently falls behind CLUSTER in most cases, although it occasionally outperforming CLUSTER with specific algorithms.

Additionally, in direct comparison with other methods, TOMEK generally demonstrates lower effectiveness than both CLUSTER and ENN, particularly when considering their average performance. This is evident in its consistently lower performance across various scenarios, with CLUSTER and ENN outshining TOMEK in most cases. Nevertheless, TOMEK performs better than RUS, achieving an improvement of approximately 27%. This highlights the impact of randomly removing examples in the original dataset.

Upon close examination of Table 4.2, illustrating the distribution after applying sampling methods to synthetic data, we observe balanced instances for all classes around 7700, similar to the count of the majority class, achieved through oversampling methods. However, for undersampling methods, particularly CLUSTER, there's a substantial reduction in the “NONE” class instances, aligning with the count of the minority class. Additionally, some of the other classes experience reductions.

Table 4.2. Distribution of Class Instances for the Synthetic Dataset After Resampling Techniques

Machine Failure	Imbalanced Data	ROS	SMOTE	ADASYN	RUS	CLUSTER	TOMEK	ENN
NONE	7734	7734	7734	7734	7734	36	7703	7734
TWF	36	7734	7734	7731	36	36	36	36
HDF	91	7734	7734	7761	36	36	80	29
PWF	68	7734	7734	7728	36	36	59	41
OSF	36	7734	7743	7732	36	36	62	20

Source: Author

As we shift from examining synthetic data to real-world datasets, the insights we've gained from analyzing resampling techniques will be instrumental as we examine the results with real-world data.

Table 4.3 presents the classification results obtained using the imbalanced real-world dataset and its balanced versions after applying the undersampling techniques. As in

the previous table, the second column gives the detailed results in Figure 4.6. Again, the Jrip algorithm performs the best for the imbalanced data.

Table 4.3. Recall Values for the Real-world Dataset Before and After Applying the Resampling Methods

Classifier	Recall for Imbalanced Data	Recall after Undersampling			
		RUS	CLUSTER	TOMEK	ENN
NaiveBayes	0.34	0.33	0.39	0.35	0.21
Multilayer Perceptron	0.25	0.25	0.25	0.25	0.25
MultiClassClassifier	0.48	0.25	0.25	0.36	0.25
Jrip	**0.50**	0.31	0.28	0.41	0.40
RandomTree	0.33	**0.46**	0.25	0.32	0.19

Source: Author

Note that no information is provided for the Lazy category here, as the algorithms in this group could not be run for the real-world dataset due to computational issues. Also, note that no results are provided for the oversampling methods. Although we have generated the oversampled data for the real-world dataset, the classification algorithms either remain unresponsive or crash as they run. Here, there are only two improvements with respect to the imbalanced dataset: one for the NaiveBayes case for CLUSTER and TOMEK, and another for the RandomTree classifier with RUS. In other cases, results obtained for the imbalanced dataset are generally better than those obtained for the balanced datasets.

The main reason undersampling methods are ineffective becomes more evident when we carefully examine the detailed distribution of majority and minority classes. The data in Table 4.4 reinforces this observation, highlighting the distribution of examples in each class of the real-world dataset. Here, we find approximately 11 million data points representing normal working conditions of the machines, while the number of examples for machine failures is significantly lower, with only 10, 14, and 38 instances

for different failure types. Machine learning methods can effectively learn representations of normal working conditions, given the substantial amount of data available. However, when it comes to the limited number of examples representing the minority classes, which is less than adequate, it becomes a considerable challenge to create models that accurately represent these infrequent events.

Table 4.4. The Number of Examples in Classes of the Imbalanced and Balanced Real-world Training Data

Machine Failure	Imbalanced Data	Balanced Data with RUS	Balanced Data with CLUSTER	Balanced Data with TOMEK	Balanced Data with ENN
NONE	11,109,641	500,000	10	11,109,598	11,109,212
Flowcool Pressure	38	38	38	38	38
Flowcool Pressure Too High Check Flowcool Pump	10	10	10	10	10
Flowcool Leak	14	14	14	14	14

Source: Author

As we navigate the intricacies of balanced datasets, it becomes apparent that undersampling, while effective in diminishing instances of the majority class, intricately preserves the count of examples in the minority classes. This equilibrium, however, is particularly conspicuous in the Random Undersampling (RUS) technique, where a notable number of majority class instances persist. Consequently, algorithms can accurately predict the majority class, leading to results akin to those obtained with imbalanced data.

The subtleties of the challenge become more pronounced with the Cluster-Based Undersampling (CLUSTER) technique. In this method, the intentional reduction of instances in the majority class reaches a critical point where constructing a robust classifier becomes unattainable. As a result, outcomes for datasets balanced with the

CLUSTER method tend to exhibit a general decline in performance compared to their imbalanced counterparts.

Regarding the Tomek Links (TOMEK) technique, its impact is less pronounced as it dropped only 43 instances of the majority class. This limited reduction may not significantly alter the balance between the classes, and the classifier's performance may still be influenced by the majority class. It's worth noting that Tomek Links might not significantly reduce the majority class instances in situations where the majority class is significantly larger than the minority class, and the instances of both classes are not closely located in the feature space. Due to the limited reduction in the majority class by Tomek Links, the results exhibit more alignment with RUS and imbalanced data compared to the CLUSTER method, which undergoes a more substantial reduction in majority class instances.

Edited Nearest Neighbors (ENN) reduced 429 instances, showing slightly better performance than Tomek Links. Like Tomek Links, ENN aligns more closely with RUS and imbalanced data than the more impactful CLUSTER method. Other undersampling techniques or a combination of multiple techniques might be more effective in such imbalanced scenarios.

This delicate interplay of factors underscores the complexity of achieving optimal results in the pursuit of a balanced dataset. It's not merely about numerical adjustments; rather, it involves a nuanced understanding of the trade-offs inherent in dataset balancing. These considerations are crucial for researchers navigating the intricate terrain of imbalanced datasets, highlighting the need for a thoughtful and tailored approach to dataset balancing for robust model performance.

4.9. Analyzing the Efficacy of Hybrid Sampling Approaches

While navigating the challenges associated with conventional sampling methods, our investigation into a hybrid strategy that merges oversampling and undersampling endeavors to overcome these constraints and present a more potent approach for improving classification performance.

To improve classification results for synthetic and real-world datasets, we are investigating a hybrid method that merges the strengths of oversampling and undersampling techniques. This strategy seeks to capitalize on the advantages of both

methods, addressing the challenges posed by extreme class imbalances. Table 4.5 presents the outcomes of applying the oversampling-undersampling hybrid method on our synthetic dataset. A comparative analysis with Table 4.1, where standard sampling methods were employed, reveals a noteworthy advancement in classification performance. In Table 4.1, the highest recall for Jrip after the cluster was 87%, with an average recall for other classifiers around 59%. The results in Table 4.5 showcase significant improvements, with most recall values reaching 100%, resulting in a new average of approximately 98% for classifiers, outperforming the traditional methods' average of 65.5%.

The consistent achievement of 100% recall by IBk, AttributeSelectedClassifier, and J48 for all classifiers under the hybrid method is particularly noteworthy. Simultaneously, we observe a notable improvement in the performance of the Multilayer Perceptron classifier. In Table 4.1, the highest recall achieved was 81%. However, after applying the hybrid approach, specifically in Table 4.5, we identify SMOTE-ENN, a combination that exhibits a remarkable recall of 97% for this classifier. It's important to highlight that SMOTE-ENN is an integrated resampling method available within the Imbalanced-learn library. This method inherently combines both SMOTE and ENN. Unlike other combinations, where we applied oversampling and then undersampling sequentially, SMOTE-ENN is seamlessly integrated into the library, ensuring a simultaneous application of both methods.

Results in Table 4.5 underscore the exceptional efficacy of the hybrid approach in significantly improving classification performance. Additionally, the substantial increase to 100% recall for classifiers such as IBk, AttributeSelectedClassifier and J48 after applying the hybrid method highlights its notable effectiveness for certain classifiers.

The hybrid approach, which combines oversampling and undersampling, appears to create a more balanced and representative training set. This enhancement enables classifiers like IBk, AttributeSelectedClassifier and J48 to better discern and comprehend patterns associated with majority and minority classes.

Table 4.5. Recall Values for the Synthetic Dataset After Combining Oversampling, Undersampling Hybrid Approach

Classifier	**Recall for Imbalanced Data**	**Oversampling + Undersampling**				
		ADASYN + TOMEK	ADASYN + CLUSTER	SMOTE + TOMEK	SMOTE + CLUSTER	SMOTE - ENN
BayesNet	0.51	0.96	0.96	0.95	0.96	0.96
Multilayer Perceptron	0.42	0.92	0.96	0.96	0.96	0.97
IBk	0.36	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
AttributeSelected Classifier	0.49	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
Jrip	**0.52**	1.0	0.99	0.99	1.0	0.99
J48	0.49	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**

Source: Author

Examining Table 4.6 reveals the class distribution post the application of the oversampling and undersampling hybrid method on the synthetic dataset. Notably, all classes exhibit a similar number of instances, closely resembling the initial count of instances in the “NONE” class. This emphasizes the effectiveness of the hybrid resampling approach in achieving a balanced distribution across the diverse classes, resulting in effective recall outcomes for this combined methodology.

Table 4.6. Distribution of Class Instances for the Synthetic Dataset After Combining Oversampling, Undersampling Hybrid Approach

Machine Failure	**Imbalanced Data**	**Oversampling + Undersampling**				
		ADASYN + TOMEK	ADASYN + CLUSTER	SMOTE + TOMEK	SMOTE + CLUSTER	SMOTE - ENN
NONE	7747	7731	7718	7733	7734	7103
HDF	91	7746	7718	7732	7734	7734
OSF	70	7727	7718	7733	7734	7696
PWF	68	7740	7718	7733	7734	7710
TWF	36	7717	7718	7729	7734	7681

Source: Author

Now, let's examine the reverse order of the hybrid approach by first applying undersampling, followed by oversampling, as presented in Table 4.7. It becomes evident that the outcomes closely mirror those obtained through the initial oversampling and then undersampling method. The recall values are nearly identical, with a rounded average of 98%. A detailed examination indicates a subtle advantage for the undersampling-oversampling sequence, but the difference is so marginal that it implies nearly identical performance between the two hybrid approaches. In contrast to the previous hybrid approach, here, the absence of a built-in combination for ENN and SMOTE led us to apply ENN first, followed by SMOTE in this reversed hybrid approach. However, despite this customized arrangement, the results exhibit promising outcomes.

In essence, whether opting for oversampling followed by undersampling or the reverse sequence, the classification results consistently outperform those obtained through singular resampling methods. This highlights the superior effectiveness of the hybrid approach, showcasing notable improvements compared to applying either oversampling or undersampling in isolation.

Table 4.7. Recall Values for the Synthetic Dataset After Combining Undersampling, Oversampling Hybrid Approach

Classifier	**Recall for Imbalanced Data**	**Undersampling + Oversampling**				
		TOMEK + ADASYN	CLUSTER + ADASYN	TOMEK + SMOTE	CLUSTER + SMOTE	ENN + SMOTE
BayesNet	0.51	0.96	0.95	0.96	0.95	0.94
Multilayer Perceptron	0.42	0.96	0.96	0.96	0.96	0.96
IBk	0.36	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
AttributeSelected Classifier	0.49	1.0	0.99	1.0	0.99	1.0
Jrip	**0.52**	1.0	0.99	1.0	1.0	1.0
J48	0.49	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**

Source: Author

Examining the distribution in Table 4.8 after applying the undersampling method followed by oversampling reveals a balanced distribution among classes. Notably, the variance in instances stems from the distinct strategy employed by the CLUSTER method, resulting in fewer instances than the count of the “NONE” class. Remarkably, the outcomes remain effective, nearly the same as those achieved through the oversampling-undersampling sequence.

Now, our focus shifts to the application of hybrid methods on our real-world dataset to gauge their effectiveness. It's essential to bear in mind that our real-world dataset presents a considerable challenge due to its high data imbalance, with a minimal number of anomalies.

Table 4.8. Distribution of Class Instances for the Synthetic Dataset After Combining Undersampling, Oversampling Hybrid Approach

Machine Failure	Imbalanced Data	Undersampling + Oversampling				
		TOMEK + ADASYN	CLUSTER + ADASYN	TOMEK + SMOTE	CLUSTER + SMOTE	ENN + SMOTE
NONE	7747	7713	1280	7713	1236	7481
HDF	91	7719	1303	7713	1236	7481
OSF	70	7710	1296	7713	1236	7481
PWF	68	7689	1314	7713	1236	7481
TWF	36	7707	1294	7713	1236	7481

Source: Author

As we delve into exploring hybrid methods on our real-world dataset, we're approaching it strategically, considering the dataset's significant imbalance and the limited number of anomalies. We've encountered a computational challenge that previously made it tricky to use classifiers on oversampled data, often causing crashes. To work around this, we've decided to concentrate on a specific part of the dataset—tool 03M02, with around 1.5 million records. This focused approach helps us overcome computational issues, allowing us to apply oversampling and undersampling methods in our hybrid approach. Similar to our approach with synthetic data, we will conduct both oversampling followed by undersampling and vice versa.

In our exploration of the hybrid approach, we will utilize the same classifiers previously employed to analyze real-world data, as indicated in Table 4.3.

This systematic application of hybrid techniques aims to explore their impact on the real-world dataset, maintaining a comprehensive and balanced investigative approach.

Before we delve into the application of hybrid methods to our newly created dataset, a crucial preliminary step involves applying the same sampling methods that we employed earlier on our extensive 1.5 million-record dataset. This initial phase aims

to establish a baseline understanding of the dataset's characteristics and classifier performance before introducing the complexity of hybrid approaches.

Table 4.9 presents the results for each sampling method, including ROS, SMOTE, ADASYN, RUS, CLUSTER, TOMEK, and ENN. Upon a comprehensive examination of Table 4.9, it becomes apparent that the improvements achieved by the sampling methods are not substantial compared to the imbalanced data, as the dataset still exhibits a notable level of imbalance. While certain individual improvements can be observed, such as a 6% increase in recall for the RandomTree classifier when employing the ADASYN method, the overall average improvement across all sampling methods is insignificant. Significantly, we observe superior outcomes with methods like SMOTE, ADASYN, CLUSTER, and TOMEK, while ROS and RUS, which employ random strategies, exhibit comparatively lower effectiveness. Additionally, ENN exhibits lower performance compared to CLUSTER and TOMEK, with an average similar to RUS.

Table 4.9. Recall Values for the Modified Real-World Dataset After Applying the Resampling Methods

Classifier	**Recall for Imbalanced Data**	**Recall After Oversampling**			**Recall After Undersampling**			
		ROS	**SMOTE**	**ADASYN**	**RUS**	**CLUSTER**	**TOMEK**	**ENN**
NaiveBayes	0.37	0.38	0.40	0.38	0.34	0.38	0.37	0.32
Multilayer Perceptron	0.25	0.29	0.27	0.25	0.25	0.25	0.25	0.25
MultiClass Classifier	0.48	0.32	0.49	0.45	0.25	0.25	0.29	0.27
Jrip	**0.52**	0.44	0.50	0.49	0.28	0.30	0.33	0.29
RandomTree	0.34	0.36	0.39	0.40	0.29	0.30	0.30	0.28

Source: Author

Examining Table 4.10, illustrating class instances after each method, reveals that ENN reduces minority classes more than other methods. This suggests that crucial information might be removed, potentially contributing to its inferior performance compared to other techniques.

Table 4.10. Distribution of Class Instances for the Modified Real-World Dataset

Machine Failure	Imbalanced Data	ROS	SMOTE	ADASYN	RUS	CLUSTER	TOMEK	ENN
NONE	1,525,596	1,525,596	1,525,596	1,525,596	500,000	109	1,525,516	1,525,376
Flowcool Pressure Too High Check Flowcool Pump	234	1,525,596	1,525,542	1,525,375	234	109	208	144
Flowcool Pressure	577	1,525,596	1,525,311	1,525,559	577	109	524	109
Flowcool Leak	109	1,525,596	1,525,595	1,525,609	109	109	88	55

Source: Author

Motivated by these findings, we aim to investigate the performance of hybrid approaches using the same classifiers employed in conjunction with the sampling methods. The upcoming analysis will offer valuable insights into the efficacy of these hybrid methods in addressing the imbalanced nature of our dataset.

In the context of the modified real-world dataset, Table 4.10 delineates the distribution of class instances, including the initial imbalanced data distribution. Initially, there are approximately 1.5 million instances of the “NONE” class. After applying oversampling methods, the instances for all classes are increased to nearly 1.5 million. Upon applying undersampling methods, we observe that for the “NONE” class in the CLUSTER method, there are 109 instances identical to the lowest minority class count. In contrast, for TOMEK and ENN, we have nearly 1.5 million. We retained the RUS method at 500 instances, in line with the original distribution for the “NONE” class in

the real-world dataset. When considering other classes for undersampling, their instances are also reduced, with some remaining unchanged.

Shifting our focus from individual sampling methods to hybrid approaches in the context of the modified real-world dataset, let's examine the outcomes presented in Table 4.11. This table illustrates the effects of combining oversampling and subsequent undersampling on our extensive 1.5 million-record dataset. Comparing the results from Table 4.11, where we apply hybrid methods, and Table 4.9, which employs only single sampling methods, a comprehensive analysis of the tables reveals notable enhancements across most classifiers.

Upon comparing the outcomes of the hybrid approach, where both SMOTE and CLUSTER methods are combined, with our earlier results in Table 4.9, a significant enhancement is particularly noticeable for the RandomTree classifier. In the hybrid approach, the combination of SMOTE and CLUSTER outperforms individual methods by 20%, establishing itself as the top-performing classifier.

Table 4.11. Recall Values for the Real-World Dataset After Combining Oversampling, Undersampling Hybrid Approach

Classifier	**Recall for Imbalanced Data**	**Oversampling + Undersampling**				
		ADASYN + TOMEK	ADASYN + CLUSTER	SMOTE+ TOMEK	SMOTE + CLUSTER	SMOTE-ENN
NaiveBayes	0.37	0.49	0.52	0.53	0.52	0.31
Multilayer Perceptron	0.25	0.35	0.33	0.33	0.32	0.29
MultiClass Classifier	0.48	0.40	0.38	0.40	0.40	0.27
Jrip	**0.52**	0.47	0.52	0.49	0.51	0.44
RandomTree	0.34	**0.50**	**0.54**	**0.60**	**0.56**	0.21

Source: Author

SMOTE-ENN faced challenges and did not achieve balanced results for RandomTree. Moreover, most values for SMOTE-ENN were lower than the original results for imbalanced dataset, except for MultilayerPerceptron.

Additionally, NaiveBayes demonstrates a noteworthy improvement of 13%. Remarkably, Jrip, which previously fell short of reaching the recall of the original imbalanced dataset in Table 4.9, now attains the same recall value after the hybrid approach.

In Table 4.12, the hybrid approach of combining oversampling and undersampling results in a balanced dataset, with all class instances around 1.5 million, similar to the count of the "NONE" class. Overall, the table illustrates the effectiveness of the hybrid resampling strategy in achieving class balance.

Table 4.12. Distribution of Class Instances for the Modified Real-World Dataset After Combining Oversampling, Undersampling Hybrid Approach

Machine Failure	**Imbalanced Data**	**Oversampling + Undersampling**				
		ADASYN + TOMEK	ADASYN + CLUSTER	SMOTE+ TOMEK	SMOTE + CLUSTER	SMOTE-ENN
NONE	1,525,596	1,517,046	1,521,064	1,521,168	1,523,063	1,524,252
Flowcool Pressure Too High Check Flowcool Pump	234	1,516,917	1,521,064	1,521,917	1,523,063	1,467,643
Flowcool Pressure	577	1,517,039	1,521,064	1,521,039	1,523,063	1,472,620
Flowcool Leak	109	1,517,071	1,521,064	1,520,962	1,523,063	1,494,466

Source: Author

In our exploration of the 1.5 million-record dataset, the results from the implementation of undersampling followed by oversampling, as depicted in Table 4.13, align closely with those observed in Table 4.11. Once again, the RandomTree classifier emerges as the top-performing classifier, showcasing a substantial average improvement, nearly on par with the oversampling-then-undersampling approach, with a difference of only approximately 1%. While subtle variations exist, such as a minor 0.5% decrease in the average recall for Jrip, these differences prove negligible, falling below the 1% threshold. Notably, NaiveBayes demonstrates an improvement of 1%. The most significant change is observed in the MultiClassClassifier, which achieves a higher average recall by 6% compared to the oversampling-then-undersampling approach. This consistent performance across both hybrid approaches underscores the stability and reliability of the hybrid methodology in enhancing classifier performance on our real-world dataset. Additionally, it's noteworthy that, for the real-world dataset, both the combination of ENN followed by SMOTE and the built-in SMOTE-ENN exhibit less effectiveness compared to other combinations.

Table 4.13. Recall Values for the Modified Real-World Dataset After Combining Undersampling, Oversampling Hybrid Approach

Classifier	**Recall for Imbalanced Data**	**Undersampling + Oversampling**				
		TOMEK + ADASYN	CLUSTER + ADASYN	TOMEK + SMOTE	CLUSTER + SMOTE	ENN - SMOTE
NaiveBayes	0.37	0.53	0.48	0.55	0.53	0.33
Multilayer Perceptron	0.25	0.35	0.35	0.30	0.33	0.25
MultiClass Classifier	0.48	0.48	0.44	0.45	0.49	0.30
Jrip	**0.52**	0.49	0.50	0.49	0.49	0.41
RandomTree	0.34	**0.58**	**0.61**	**0.59**	**0.61**	0.24

Source: Author

Observing the distribution in Table 4.14, a balance in data instances is evident after combining undersampling and oversampling techniques. The instance count converges around 1.5 million, aligning closely with the initial count of the majority class. However, for the hybrid method incorporating the CLUSTER combination, the instance count stabilizes around 1 million.

Table 4.14. Distribution of Class Instances for the Modified Real-World Dataset After Combining Undersampling, Oversampling Hybrid Approach

Machine Failure	**Imbalanced Data**	**Undersampling + Oversampling**				
		TOMEK + ADASYN	CLUSTER + ADASYN	TOMEK + SMOTE	CLUSTER + SMOTE	ENN - SMOTE
NONE	1,525,596	1,525,566	1,017,086	1,525,566	1,017,064	1,525,531
Flowcool Pressure Too High Check Flowcool Pump	234	1,525,740	1,017,178	1,525,566	1,017,064	1,525,534
Flowcool Pressure	577	1,525,654	1,017,081	1,525,566	1,017,064	1,525,542
Flowcool Leak	109	1,525,588	1,017,087	1,525,565	1,017,064	1,525,498

Source: Author

In summary, the application of both hybrid approaches to our real-world dataset demonstrates their consistent performance in enhancing classifier results. The robustness of the hybrid methodology is evident, providing comparable outcomes regardless of the sequence of oversampling and undersampling. Notably, the Synthetic dataset exhibits near-perfect recall for all classifiers.

However, the real-world dataset, characterized by extreme class imbalance and minimal anomalies, presents a distinct challenge. The improvement observed, while significant, doesn't match the remarkable success achieved with synthetic data.

The best recall achieved is 61%, reflecting the inherent difficulty of the task due to a lack of variety in the limited anomalous instances. Despite not reaching the same high levels as in synthetic data, the observed improvement of approximately 20% over traditional sampling methods for undersampling-then-oversampling is noteworthy. This leaves us contemplating the potential for more substantial enhancements if the dataset featured a slightly increased number of anomalies. This underscores the intricacies of tackling extreme class imbalances in real-world datasets, highlighting the need for diverse and representative data, a challenge often compounded by difficulties in collecting public datasets. This circumstance underscores the significance of employing specialized strategies, such as hybrid approaches, to navigate through challenging scenarios of this nature.

In our effort to enhance classification performance on imbalanced datasets, we intentionally chose not to include the combination of Random Oversampling (ROS) and Random Undersampling (RUS) in our hybrid sampling methods. This decision comes from our past observations, where these techniques showed a somewhat straightforward and, at times, overly simple approach. Our previous experiences with ROS indicated that it might increase the risk of overfitting by copying minority class instances, potentially leading to biased models. On the other hand, by removing majority class instances without discrimination, RUS could oversimplify the learning process, possibly compromising the model's ability to understand the data's complexities.

Instead, we focused on more advanced hybrid methods, such as combining ADASYN with TOMEK and ADASYN with CLUSTER. These approaches aim to handle the challenges in imbalanced datasets more effectively. ADASYN, for instance, generates synthetic data in areas where the classifier may struggle, making the model more robust. TOMEK and CLUSTER, used in the undersampling phase, strategically remove redundant and borderline instances, contributing to a more balanced and representative dataset. Additionally, we utilized SMOTE with TOMEK, CLUSTER, and SMOTE-ENN, introducing synthetic data strategically and removing redundant and borderline instances to effectively address challenges in imbalanced datasets. We observed lower results for the combinations of SMOTE with ENN and ENN followed by SMOTE compared to other hybrid combinations.

5. CONCLUSION AND FUTURE RESEARCH

Balanced data poses a significant challenge in real-world scenarios, where normal data outnumber anomalies. This study employs data-level sampling techniques to balance synthetic and real-world datasets. The JRip algorithm performed exceptionally well for both synthetic and real-world imbalanced datasets. Resampling techniques, except for Random Undersampling, contribute significantly to the classification success of the synthetic dataset. For instance, after balancing the data with CLUSTER undersampling, the recall value for JRip improved by an impressive 35%. However, randomly removing samples from datasets leads to the loss of valuable class information, resulting in poor classification performance.

When dealing with datasets of manageable size, oversampling techniques appear to be a superior approach, consistently providing better results for the generated balanced data compared to imbalanced data. However, oversampling methods become impractical for large datasets like our real-world dataset due to the computational demands on machine learning algorithms. This is particularly challenging unless substantial computational power is available. Moreover, in cases where a large dataset comprises only a limited number of examples in minority classes, undersampling techniques may prove ineffective due to the inadequate information available for learning the accurate representation of the minority classes. This common problem in machine learning highlights the need to gather more minority class examples whenever possible.

For large datasets with a decent number of examples in the minority classes, a hybrid approach that applies undersampling followed by oversampling emerges as a viable solution. This hybrid strategy, as explored in this study, can effectively address the complexities of imbalanced datasets, showcasing its potential as a savior for challenging scenarios.

Combining this insightful analysis with the findings from the implementation of hybrid methods, it becomes evident that while substantial improvements are achievable, they

vary based on the nature of the dataset. The synthetic dataset, with a milder class imbalance, demonstrated more significant gains, reaching perfect recalls with the oversampling-undersampling hybrid method. In contrast, the real-world dataset, characterized by a higher degree of class imbalance, experienced more modest enhancements.

In conclusion, the study underscores the importance of data quality in the success of hybrid methods. As we navigate the landscape of machine learning, the imperative of addressing class imbalances becomes increasingly evident. Future work should prioritize obtaining diverse and representative real-world datasets, fostering a more comprehensive exploration of hybrid methods. This research serves as a foundation, inspiring the continuous exploration of advanced techniques that enhance the resilience and flexibility of machine learning models in real-world applications.

6. REFERENCES

Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, *6(1)*, 20–29.

Bennin, K. E., Keung, J. W., & Monden, A. (2019). On the relative value of data resampling approaches for software defect prediction. *Empirical Software Engineering, 24*, 602–636.

Carvalho, T.P., Soares, F.A.A.M.N., Vita, R., Francisco, R. da P., Basto, J.P., Alcalá, S.G.S. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, *137*. https://doi.org/10.1016/j.cie.2019.106024.

Fernández, A., García, S., Galar, M., Prati, R.C., Krawczyk, B., Herrera, F. (2018). *Learning from Imbalanced Data Sets.* Cham: Springer.

Geiler, L., Affeldt, S., Nadif, M. (2022). An effective strategy for churn prediction and customer profiling. *Data & Knowledge Engineering.* 142, 102100. https://doi.org/10.1016/j.datak.2022.102100.

He, H., & Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering, 21(9),* 1263-1284. https://doi.org/10.1109/TKDE.2008.239

He, W., Fu, X., Chen, S. (2023). Advancing polytrauma care: developing and validating machine learning models for early mortality prediction. Journal of Translational Medicine, 21. https://api.semanticscholar.org/CorpusID:262218605

Japkowicz, N. (2000). The Class Imbalance Problem: Significance and Strategies. *Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000): Special Track on Inductive Learning*.

Jardine, A. K. S., Lin, D., Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing, 20(7),* 1483-1510. https://doi.org/10.1016/j.ymssp.2005.09.012.

Kane, A., Kore, A. S., Khandale, A. N., Nigade, S. S., & Josh, P. P. (2022). Predictive Maintenance using Machine Learning. *arXiv*. https://doi.org/10.48550/arXiv.2205.09402

Kotsiantis, S., Kanellopoulos, D., & Pintelas, P. (2005). Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering, vol. 30,* 25-36.

Kulkarni, A., Chong, D., & Batarseh, F. A. (2020). Foundations of data imbalance and solutions for a data democracy, *Data Democracy*, 83-106. https://doi.org/10.1016/B978-0-12-818366-3.00005-8

Lin, W. C., Tsai, C. F., Hu, Y. H., & Jhang, J. S. (2017). Clustering-based undersampling in class-imbalanced data. *Information Sciences, 409-410,* 17-26. https://doi.org/10.1016/j.ins.2017.05.008

Ling, C., & Li, C. (1998). Data Mining for Direct Marketing: Problems and Solutions. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98),* New York, NY: AAAI Press.

Nuhu, A., Zeeshan, Q., Safaei, B., & Shahzad, A. (2022). Machine learning-based techniques for fault diagnosis in the semiconductor manufacturing process: a comparative study. *The Journal of Supercomputing, 79*. https://doi.org/10.1007/s11227-022-04730-x

Pereira, R. M., Costa, Y. M. G., & Silla Jr., C. N. (2020). MLTL: A multi-label approach for the Tomek Link undersampling algorithm. *Neurocomputing, 383,* 95-105. https://doi.org/10.1016/j.neucom.2019.11.076

Ran, Y., Zhou, X., Lin, P., Wen, Y., & Deng, R. (2019). A Survey of Predictive Maintenance: Systems, Purposes, and Approaches. *IEEE Communications Surveys & Tutorials*.

Ramyachitra, D., & Manikandan, P. (2014). Imbalanced Dataset Classification and Solutions: A Review. *International Journal of Computing and Business Research, 5(4).*

Rathore, S. S., Chouhan, S. S., Jain, D. K., & Vachhani, A. G. (2022). Generative Oversampling Methods for Handling Imbalanced Data in Software Fault Prediction. *IEEE Transactions on Reliability, 71,* 747-762. https://doi.org/10.1109/TR.2022.3158949.

Rekha, G., Tyagi, A. K., Sreenath, N., & Mishra, S. (2021). Class Imbalanced Data: Open Issues and Future Research Directions. *International Conference on Computer Communication and Informatics (ICCCI),* 1-6. https://doi.org/10.1109/ICCCI50826.2021.9402272

Samatas, G. G., Moumgiakmas, S. S., & Papakostas, G. A. (2021). Predictive Maintenance - Bridging Artificial Intelligence and IoT. *IEEE World AI IoT Congress (AIIoT)*, 0413-0419. https://doi.org/10.1109/AIIoT52608.2021.9454173

Singh, A. (2020, March 17). Calculating Accuracy of an ML Model. *Analytics Vidhya*. https://medium.com/analytics-vidhya/calculating-accuracy-of-an-ml-model-8ae7894802e

Solberg, A., & Solberg, R. (1996). A Large-Scale Evaluation of Features for Automatic Detection of Oil Spills in ERS SAR Images. *International Geoscience and Remote Sensing Symposium*, 1484–1486, Lincoln, NE.

Tantithamthavorn, C., Hassan, A. E., & Matsumoto, K. (2020). The Impact of Class Rebalancing Techniques on the Performance and Interpretation of Defect Prediction Models, *IEEE Transactions on Software Engineering, vol. 46, no. 11*, pp. 1200-1219. https://doi.org/10.1109/TSE.2018.2876537

Vafaei, N., Ribeiro, R. A., and Camarinha-Matos, L. M. (2019). Fuzzy Early Warning Systems for Condition-Based Maintenance. *Computers & Industrial Engineering, 128*, 736–746. https://api.semanticscholar.org/CorpusID:68236990

Wei, G., Zhao, X., He, S., & He, Z. (2019). Reliability Modeling with Condition-Based Maintenance for Binary-State Deteriorating Systems Considering Zoned Shock Effects. *Computers & Industrial Engineering, 130*, 282-297. https://doi.org/10.1016/j.cie.2019.02.034

Xu, Z., Shen, D., Nie, T., & Kou, Y. (2020). A hybrid sampling algorithm combining M-SMOTE and ENN based on Random Forest for medical imbalanced data. *Journal of Biomedical Informatics, 107*, 103465. https://doi.org/10.1016/j.jbi.2020.103465

APPENDICES

APPENDIX 1. DATASETS FOR PREDICTIVE MAINTENANCE

Algorithm	Data type	Year	Reference	Description of the data	Availability	Data-source
Logistic Regression	*RD*	*2017*	*T.L Wu et al.*	*From the actual machine*	*NA*	-
	SD	2015	H. K. Li et al.	Dongyu Machine and Tool CMV-850A Center	A	Mill DataSet 2007
	RD	2005	J.H. Yan et al.	From the doorcontrol board	NA	-
	RD	2017	J. J. A. Costello et al.	Vibration Data	NA	From EDF Energy
	SD	2010	W. Caesarendra et al.	Bearing Data	A	From simulation by MATLAB and test rig
	RD	2020	Cho et al.	Data collected from the extrusion process	NA	Environmental data of plastic extrusion process from a manufacturing company
Support Vector Machine	SD	2011	M. Saimurugan et al.	Vibration signals	NA	From a simulator
	SD	2015	Machado and Mota	Electrical signals	NA	Simulated data
	RD	2015	Susto et al.	Benchmark of semiconductor manufacturing maintenance	NA	-
	RD	2017	Mathew et al.	Turbo fan engine data from a prognostics data repository of NASA	A	Prognostics data repository of NASA
	RD	2018	Lasisi and Attoh-Okine	Track geometry data	NA	-
	RD	2018	U. Shafi et al.	From sensors on Toyota Corolla cars	NA	Provided by Toyota Corolla
Decision Tree	SD	2016	A. Krishnakumari	Data were collected using a commercial four-channel data acquisition module	NA	From the fault simulator
	RD	2017	Mathew et al.	Turbo fan engine data from a prognostics data repository of NASA	A	Prognostics data repository of NASA
	RD	2020	Toma et al.	Stator current as input data	A	Drive Technology (KAt) Research datacenter of Paderborn University
	RD	2017	Canizo et al.	Status data and operational data from the performance of wind turbines	NA	Status data (alarms activations and deactivations) and operational data
	RD	2018	Paolanti et al.	Data from sensors, PLCs and communication protocols	NA	Data set was from 14 Rover machines: 5 ES
	RD	2018	Kolokas et al.	Process sensor data from operation periods	NA	-
	RD	2018	Syafrudin et al.	IoT-generated sensor data	NA	Automotive manufacturing assembly line in Korea
Deep Neural Network	RD	2018	Luo et al.	Vibration signal	NA	CNC machining center in an automobile factory
	RD	2018	Amihai et al.	Vibration data	NA	-
	RD	2019	Wen et al.	Motor bearing dataset	A	Motor bearing dataset from CWRU
	RD	2019	Cipollini et al.	Data froman inverter-fed motor mounting	NA	Inhouse (3 faults, 4 loads, 1400 samples)
	RD	2020	Yasutomi and Enoki	Belt conveyor data	NA	-
	RD	2020	Huang et al.	Droplet jetting process video data	NA	-
	RD	2020	ArellanoEspitia et al.	Electrical motor driven system data	NA	-
	RD	2015	Biswal and Sabareesh	Accelerometer data	NA	-
	RD	2014	C. Jin et al.	Vibration and pressure data	NA	From the test bed
	RD	2015	D. Y. You et al.	Optical and visual data	NA	From a high-power disk laser welding system
	RD	2010	A. K. Mahamad et al.	Vibration data	A	Provided by the Center of IMS
	RD	2015	R. Ahmed et al.	Vibration data	NA	From Ford's Powertrain Engineering Research and Development Center
	RD	2019	Imoto et al.	Semiconductor fabrication data	NA	Real semiconductor fabrication data sets
K-nearest neighbors	RD	2017	Mathew et al.	Turbo fan engine data from a prognostics data repository of NASA	A	Prognostics data repository of NASA
	RD	2020	Toma et al.	Stator current as input data	A	Drive Technology (KAt) Research datacenter of Paderborn University
K-means	RD	2017	Eke et al.	Dissolved gases concentrations	NA	Dissolved gas data
	RD	2018	Amruthnath and Gupta	Vibration data	NA	Data collected from an exhaust fan
	RD	2018	Uhlmann et al.	Machine tool sensor data	NA	Sensor data from 206 manufacturing processes
Linear Regression	RD	2009	Onanena et al.	Electrochemical impedance	NA	-
	RD	2012	Susto et al.	Ion Beam Etching process	A	Infineon Technologies Austria AG, real industrial production dataset