UNISINOS

Programa de Pós-Graduação em

Computação Aplicada

Mestrado/Doutorado Acadêmico

Vitor Werner de Vargas

Heimdall: an architecture for online Machine Learning through imbalanced data

São Leopoldo, 2023

UNIVERSIDADE DO VALE DO RIO DOS SINOS — UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA
NÍVEL MESTRADO

VITOR WERNER DE VARGAS

HEIMDALL:
AN ARCHITECTURE FOR ONLINE MACHINE LEARNING THROUGH IMBALANCED DATA

São Leopoldo
2023

V297h Vargas, Vitor Werner de.
Heimdall : an architecture for online machine learning through imbalanced data / by Vitor Werner de Vargas. – 2023.
100 p. : il. ; 30 cm.

Dissertation (master's degree) — Universidade do Vale do Rio dos Sinos, Applied Computing Graduate Program, São Leopoldo, RS, 2023.
Advisor: Dr. Jorge Luis Victória Barbosa.
Co-advisor: Dr. Paulo Ricardo da Silva Pereira.

1. Imbalanced data. 2. Preprocessing. 3. Sampling. 4. Machine learning. 5. Software architecture. 6. Reactive agents. I. Title.

UDC: 004.85

Cataloging in Publication (CIP):
Librarian Alessandro Dietrich - CRB 10/2338

ATA DE BANCA EXAMINADORA DE DISSERTAÇÃO DE MESTRADO Nº 12/2023

Aluno: Vitor Werner de Vargas

Título da Dissertação: “HEIMDALL: AN ARCHITECTURE FOR ONLINE MACHINE LEARNING THROUGH IMBALANCED DATA”.

Banca: Prof. Dr. Jorge Luis Victoria Barbosa (Orientador) – Unisinos
Prof. Dr. Paulo Ricardo Da Silva Pereira (Coorientador) – Unisinos
Prof. Dr. Gabriel De Oliveira Ramos (Avaliador) – Unisinos
Prof. Dr. Marilton Sanchotene De Aguiar (Avaliador) – UFPel

Ao vinte e seis dias do mês de setembro do ano de 2023, às 14h a Comissão Examinadora de Defesa de Dissertação composta pelos professores: Prof. Dr. Jorge Luis Victoria Barbosa (Orientador) – Unisinos (participação por webconferência); Prof. Dr. Paulo Ricardo Da Silva Pereira (Coorientador) – Unisinos (participação por webconferência); Prof. Dr. Gabriel De Oliveira Ramos (Avaliador) – Unisinos (participação por webconferência) e Prof. Dr. Marilton Sanchotene De Aguiar (Avaliador) – UFPel (participação por webconferência) para analisar e avaliar a Dissertação apresentada pelo Vitor Werner de Vargas (participação por webconferência).

Considerações da Banca:
Após a apresentação realizada pelo aluno, os professores se reuniram para avaliação do trabalho. Os professores confirmaram os aperfeiçoamentos indicados durante a banca, as quais estão gravados no vídeo e deverão constar na versão final da dissertação.

Ocorreu alteração do título? (X) Não ( ) Sim
Indicar o novo título: .....
A Banca Examinadora, em cumprimento ao requisito exigido para a obtenção do Título de Mestre em Computação Aplicada, julga esta dissertação:

(X) APROVADA ( ) REPROVADA

Conforme Artigo 67 do Regimento do Programa o texto definitivo, com aprovação do Orientador, deverá ser entregue no prazo máximo de sessenta (60) dias após a defesa. O resultado da banca é de consenso entre os avaliadores. A emissão do Diploma está condicionada a entrega da versão final da Dissertação.

São Leopoldo, 26 de setembro de 2023.

Prof. Dr. Prof. Dr. Jorge Luis Victoria Barbosa – Orientador

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 / O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Código de Financiamento 001.

ACKNOWLEDGEMENTS

First of all, I would like to thank my family. If I have grown to enjoy learning, it is because my parents, Eunice and Volmir, have nourished my inquisitive mind since I was a little boy. If I made wise choices, it is because my girlfriend, Bruna, advised me after patiently hearing my predicaments. If I allowed myself to do so much, it is because my sister, Raquel, was a role model. If I had the confidence to follow this path, it is because I knew I could always count on the unconditional love and support from my family. Thank you. You are the most important people in my life.

Additionally, I would like to thank my advisor, professor Dr. Jorge Luis Victoria Barbosa, and co-advisor, professor Dr. Paulo Ricardo da Silva Pereira, who guided me forward in every step of my master's studies. Thank you for not only presenting opportunities but also investing effort into making them achievements. This dissertation has also been improved with the help of professors Dr. Gabriel de Oliveira Ramos, Dr. Marilton Sanchotene de Aguiar, and Dr. Sandro José Rigo. Thank you for your detailed reviews and suggestions.

I would also like to thank the professors at the UNISINOS' PPGCA, who always focused on connecting their content to the students' research, and the secretary staff for their support during a challenging period of my life.

Finally, I would like to thank CAPES for the financial support through the first 16 months of my master's studies.

"Our main business is not to see what lies dimly at a distance,
but to do what lies clearly at hand".
(Thomas Carlyle)

ABSTRACT

Machine Learning (ML) algorithms have been increasingly applied to domain areas where data is available for process automation. However, in the case of imbalanced data applications, the training process is challenging since ML algorithms intrinsically learn from balanced distributions. This research proposes Heimdall, a resourceful architecture for online ML through imbalanced data. Designed as a service for prediction and analysis requests, Heimdall serves existing applications from external systems, extending artificial intelligence capabilities and automated processes to traditional applications supervised by experts. The architecture focuses on efficiently solving imbalance and improving performance through a set of good practices compiled from mapped studies – such as probability threshold optimization, high-performance sampling, and ensemble learning. Furthermore, Heimdall proposes and evaluates the efficiency of novel functionalities. Firstly, a new performance metric corrects precision-recall balance according to the application's needs, enhancing probability threshold optimization. Secondly, the architecture independently automates data management and training pipelines through two rule-based reactive agents constantly monitoring data changes and model degradation to trigger processes. These reactive agents compose a strategy for adaptive efficiency, enabling better and more stable performance by sacrificing efficiency in warm-up conditions, and maintaining excellent performance and efficiency in hot conditions. To adequately evaluate the architecture, this study implemented a prototype for one well-studied and severely imbalanced application – Credit Card Fraud Detection (CCFD). Isolating the improvement of each proposed functionality, the analysis evaluated performance over time and overall performance against related works through five scenarios. Namely, the results indicated that the prototype achieved excellent performance even with few anomalies, and improved systemic efficiency over time. Finally, the overall performance achieved comparable results to the best-performing related works.

Keywords: Imbalanced data. Preprocessing. Sampling. Machine Learning. Software architecture. Reactive agents.

RESUMO

Algoritmos de aprendizado de máquina têm sido crescentemente utilizado em áreas de aplicação que possuem dados disponíveis para automação de processos. No entanto, no caso de aplicações com dados desbalanceados, o processo de treinamento é desafiador, visto que algoritmos de aprendizado de máquina são desenvolvidos para aprender, intrinsicamente, de distribuições balanceadas. Esta pesquisa propõe Heimdall, uma arquitetura com diversos recursos para aprendizagem de máquina ativa através de dados desbalanceados. Projetado como um serviço para atendimento de requisições de previsões e análises, Heimdall serve aplicações existentes de sistemas externos, estendendo recursos de inteligência artificial e automatização de processos a aplicações tradicionais supervisionadas por especialistas. A arquitetura soluciona o desbalanceamento através de uma série de boas práticas compiladas em mapeamentos de trabalhos relacionados – como otimização do limiar de probabilidade, amostragem de alto desempenho e aprendizado em conjunto. Adicionalmente, Heimdall propõe e avalia a eficiência de funcionalidades inovadoras. Primeiramente, uma nova métrica de performance corrige o equilíbrio entre *precision-recall* de acordo com as necessidades da aplicação, aprimorando a otimização do limiar de probabilidade. Segundamente, a arquitetura automatiza processos de gerenciamento de dados e aprendizado de máquina, de forma independente, através de dois agentes reativos baseados em regras, os quais monitoram constantemente as mudanças de dados e degradação de performance do modelo para acionar processos. Esses agentes reativos compõem uma estratégia para eficiência adaptativa, habilitando uma performance melhor e mais estável ao sacrificar eficiência em condições iniciais de implantação, e mantendo excelentes performance e eficiência em condições normais da aplicação. Para avaliar a arquitetura de forma adequada, o presente estudo implementou um protótipo para uma aplicação conhecida contendo dados severamente desbalanceados – detecção de fraudes em cartões de crédito. Isolando a melhoria de cada funcionalidade proposta, a análise avaliou a performance no decorrer do tempo e performance global versus trabalhos relacionados através de cinco cenários. Especificamente, os resultados indicam que o protótipo alcançou performance excelente mesmo com poucas anomalias e melhorou a eficiência sistêmica no decorrer do tempo. Por fim, a performance global obteve resultados similares aos melhores resultados em trabalhos relacionados.

Palavras-chave: Dados desbalanceados. Pré-processamento. Amostragem. Aprendizado de máquina. Arquitetura de software. Agentes reativos.

LIST OF FIGURES

1 Imbalanced Ratios in a binary classification problem . . . 25
2 Absolute rarity in imbalanced datasets: (a) small sample; (b) adequate sample; (c); adequate sample with noise . . . 26
3 Sampling types for imbalanced data . . . 26
4 Supervised Machine Learning: (a) example of training dataset; (b) generalization performance for different model complexities . . . 27
5 Five-fold cross validation . . . 28
6 Probability threshold optimization visual representation . . . 30
7 Precision-Recall curves . . . 30
8 Grid search results for SVM with cross-validation . . . 31
9 Basic automated Machine Learning pipeline . . . 31
10 Filtering process . . . 39
11 Taxonomy of sampling techniques proposed or compared in reviewed papers by ID . . . 47
12 Quantitative analysis of the reviewed papers proposing, comparing, and selecting: (a) sampling techniques; (b) Machine Learning models . . . 49
13 Taxonomy of Machine Learning models proposed or compared in reviewed papers by ID . . . 50
14 Development tools of reviewed applications . . . 52
15 Quantitative analysis in different domain areas for the selection of: (a) sampling techniques; (b) Machine Learning models . . . 53
16 Reviewed studies per year by digital library and type of venue . . . 54
17 Architecture for Machine Learning with imbalanced data: (a) block diagram; (b) preprocessing pipeline . . . 58
18 Architecture for Machine Learning as an online service . . . 61
19 Process monitoring applications: (a) architecture diagram; (b) components . . . 62
20 Measuring model degradation: (a) framework; (b) experimental results . . . 63
21 Heimdall's Technical Architecture Module diagram . . . 65
22 Machine Learning model evaluation delay caused by supervised labels . . . 67
23 Time series anomaly labeling: (a) chart; (b) data . . . 67
24 Sampler processes flowcharts: (a) training data generation; (b) minority update 68
25 Heimdall's pipeline and data flow . . . 71
26 Balanced Performance Indicator's flexibility: (a) probability threshold curve; (b) standard Precision-Recall curve . . . 72
27 Training control . . . 73
28 Box plot of PCA features, Scaled Amount (SA) and Scaled Time (ST) for each class of the ULB-CCFD dataset . . . 78
29 Daily samples throughout the evaluation of performance over time . . . 79
30 Testing data procedure for performance over time and overall . . . 80
31 Performance metrics and agents' triggers over time for tested scenarios . . . 82
32 Comparison of BPI over time for tested scenarios . . . 83
33 Data characteristics over time: (a) Imbalance Ratio; (b) fraudulent transactions 83

LIST OF TABLES

1	Basic Machine Learning performance metrics for classification problems	29
2	Details of topics from related works	34
3	Research Questions	37
4	Search string	37
5	Quality scores for the answers of Research Questions	39
6	Quality Assessment	40
7	Domain areas of reviewed applications	45
8	Lessons learned by answering the Research Questions	56
9	Search string for imbalanced data applications	57
10	Search string for general applications	60
11	General rules for Data Manager triggers	69
12	Heimdall databases' essential features	70
13	General rules for Training Controller triggers	73
14	Comparison between Heimdall's and related works' solutions	75
15	Evaluation Scenarios: S1-S5	79
16	Analyses of performance metrics and processing times for S1-S5	84
17	Performance comparison between tested configurations and related works	86

CONTENTS

1 INTRODUCTION **21**
1.1 Motivation 21
1.2 Research question 21
1.3 Objectives 22
1.4 Methodology 22
1.5 Structure 23

2 BACKGROUND **25**
2.1 Imbalanced data 25
2.2 Supervised Machine Learning 27
2.2.1 Model evaluation and optimization 28
2.2.2 Pipelines 30
2.3 Final considerations 31

3 RELATED WORKS **33**
3.1 Imbalanced data preprocessing for Machine Learning 33
3.1.1 Related works 33
3.1.2 Research method 36
3.1.3 Results 39
3.1.4 Conclusion 55
3.2 Real-Time Machine Learning architectures 55
3.2.1 Imbalanced data 56
3.2.2 General applications 60
3.3 Final considerations 63

4 HEIMDALL **65**
4.1 Architecture overview 65
4.2 Data management 66
4.3 Storage 69
4.4 Machine Learning and evaluation 69
4.5 Interface 74
4.6 Applications 74
4.7 Final considerations 75

5 EXPERIMENTAL EVALUATION **77**
5.1 Application and dataset 77
5.2 Testing methodology 78
5.3 Results and discussion 81
5.4 Final considerations 87

6 CONCLUSION **89**
6.1 Contributions 89
6.2 Future works 90
6.3 Publications 90

REFERENCES **93**

1 INTRODUCTION

Machine Learning (ML) has been increasingly applied to domain areas in which data is available for process automation. However, the training process is challenging since ML algorithms conceptually learn from balanced distributions (ZHANG; ZHOU; DENG, 2019). Therefore, learning from unevenly distributed samples can decrease both accuracy and reliability from the trained model. This characteristic is called imbalance or unbalance (FOTOUHI; ASADI; KATTAN, 2019).

Imbalanced data occur naturally in the majority of real-world classification problems. Nevertheless, when the ratio between the minority and majority classes is low, the minority classes tend to be ignored as noise (REKHA; REDDY; TYAGI, 2020). Consequently, the ML model becomes biased towards the majority classes (WONG; LEUNG; LING, 2014).

The solution for learning from imbalanced data applications can be implemented in two levels: data, through sampling techniques; and algorithmic, through ML algorithms optimized for imbalanced data, such as cost-sensitive and ensemble (ZHANG et al., 2017).

1.1 Motivation

Cost-sensitive learning optimizes the models for the training dataset characteristics, being hard to reapply models to new datasets. Conversely, data level solutions fix the imbalance and allow the use of standard ML models with multiple datasets (DONG; WANG, 2011). Additionally, data level solutions enable implementations in conjunction with ensemble ML models, further improving learning (ZHAO; WANG; YUE, 2020).

In this sense, studies with applications from various domain areas implement sampling techniques to correct imbalanced datasets before learning. This topic displays a growing research interest, specially since 2019 (VARGAS et al., 2022).

According to results from related works' reviews (Section 3.2), the research community has also directed efforts to advance software architectures for online ML, streamlining predictions in real-time – also known as real-time ML. Hence, online prediction models have been developed as a service for specific applications, returning predictions when accessed by clients. However, few works have proposed architectures addressing the imbalance problem.

1.2 Research question

The scenario presented in Section 1.1 leverages opportunities for exploring existing research on sampling techniques for improving ML, establishing favorable technologies, and defining new paths for further research. Consequently, the findings from such analysis can outline requirements for a new software architecture, generalized for imbalanced data applications. Therefore, this study adheres to the following research question:

"How should a Machine Learning application be implemented to efficiently solve data imbalance through sampling techniques and enable online operation for real-time predictions?"

1.3 Objectives

This research work has the main objective of proposing a software architecture which efficiently solves data imbalance through sampling and ML optimization, and extends external systems without artificial intelligence capabilities by providing an online interface. The study segments this goal into the following targets:

- Identify the most commonly used and best-performing sampling techniques and ML models for imbalanced data applications;
- Analyze and compare online ML architectures, and productive functionalities for automating and optimizing both general and imbalanced data applications;
- Specify a software architecture (Heimdall) applying a set of good practices from the state of the art in ML, and proposing novel solutions focused on efficiently solving data imbalance;
- Implement an experimental strategy focused on evaluating performance and efficiency improvements of the proposed functionalities in a severely imbalanced dataset through a prototype based on Heimdall. The evaluation must compare results against related works, analyze active learning and improvements over time.

Specifically, the proposed software architecture limits the scope to supervised ML, depending on access to a local database of an external system – with labels maintained by the system's experts. In addition, the *online* aspect refers to providing an online server for prediction requests, maintained by an automated pipeline.

1.4 Methodology

The research started by laying the foundations for the subjects needed to understand automated ML with imbalanced data. Hence, this step studied and detailed imbalanced data, resulting issues, solutions with sampling techniques, supervised ML, model evaluation, and optimization.

The second step of this study reviewed related works. This step focused on a systematic literature review of sampling techniques and ML models for imbalanced data applications. Additionally, this step reviewed software topologies for real-time ML in imbalanced data and general applications. Finally, this investigation highlighted takeaways and gaps from the reviewed domains, presenting directions for developing software architectures solving imbalance through sampling techniques.

The next step applied directions from the reviews of related works and foundations from the background to propose Heimdall – a software architecture for online ML with imbalanced data. This step consisted of the architecture design, defining essential functionalities and resources, modules, and technologies to build ML systems.

Lastly, the final step implemented a test environment with a prototype based on Heimdall and evaluated its performance and efficiency, with different configurations, against baseline architectures. Further analyses compared the results of these configurations against other studies applying the same dataset, defined whether the study achieved the objectives, and proposed paths for future studies.

1.5 Structure

This study is divided into six chapters. Chapter 1 introduces the study, contextualizing the problem and describing the adopted methodology. Later, Chapter 2 provides a background for understanding imbalanced data and supervised ML. After laying the foundations, Chapter 3 describes related works and lists highlights for developing a ML architecture and improving imbalanced data applications. Then, Chapter 4 specifies the software architecture Heimdall through the takeaways and research gaps from Chapter 3. Chapter 5 details the selected dataset, testing methodology and results. Finally, Chapter 6 summarizes the research and findings, and presents directions for future research on online ML through imbalanced data.

2 BACKGROUND

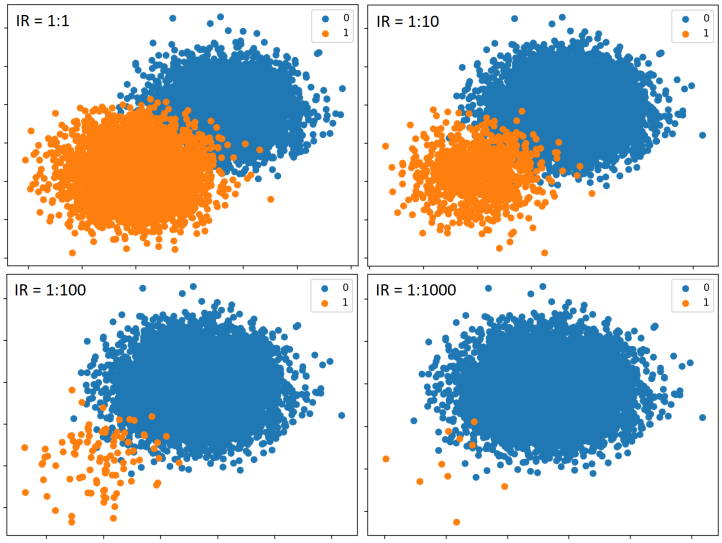
This chapter presents a brief description of the main concepts supporting the theoretical foundations of this work. Section 2.1 introduces imbalanced data and means for balancing datasets. Section 2.2 details supervised ML algorithms and their categories, performance metrics, optimization, and pipelines for automation.

2.1 Imbalanced data

Fernández et al. (2018) state that a dataset is imbalanced when there is a considerable disproportion among the number of instances of each class within the dataset. Consequently, statistical analyses, such as classification and regression, tend to under represent the minority classes.

The degree to which a dataset suffers from the imbalance problem is commonly estimated by the ratio between the minority and majority classes, called the Imbalance Ratio (IR). While most studies with imbalanced data fixate on IRs ranging between 1:4 and 1:100, several real-life applications – such as fraud and fault detection – suffer from IRs between 1:1000 and 1:5000. Figure 1 illustrates scatter plots for different IRs in a binary classification problem (BROWNLEE, 2021).

Figure 1: Imbalanced Ratios in a binary classification problem



Source: Adapted from Brownlee (2021).

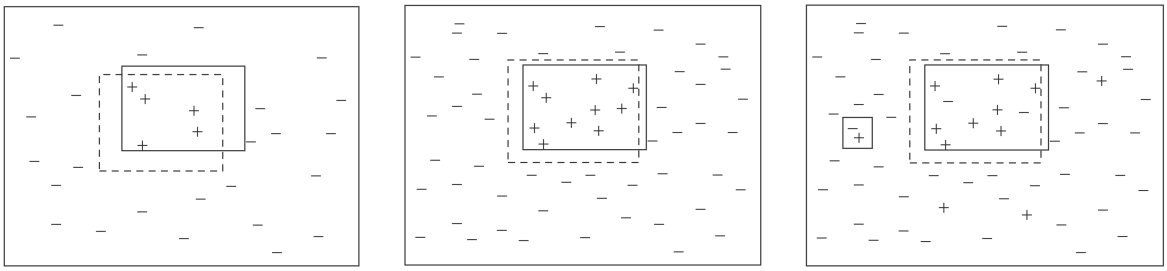
According to He and Ma (2013), there are two main issues when analyzing imbalanced data:

- *Absolute rarity – Lack of data*: when the IR is low and the application recently started collecting data, the rarity of occurrences from the minority classes deteriorates the real instance space. Consequently, more data is needed to recognize the real distribution and dimensionality of the dataset. Applications presenting noise aggravate this problem since

few instances may cause great spacial changes in the minority class. Figure 2 shows these characteristics;

- *Relative rarity – Inability to learn*: most learning algorithms are designed to learn by optimizing evaluation metrics such as accuracy. Hence, since the majority classes have proportionally higher instances, the learning process becomes biased – favoring the majority classes.

Figure 2: Absolute rarity in imbalanced datasets: (a) small sample; (b) adequate sample; (c); adequate sample with noise

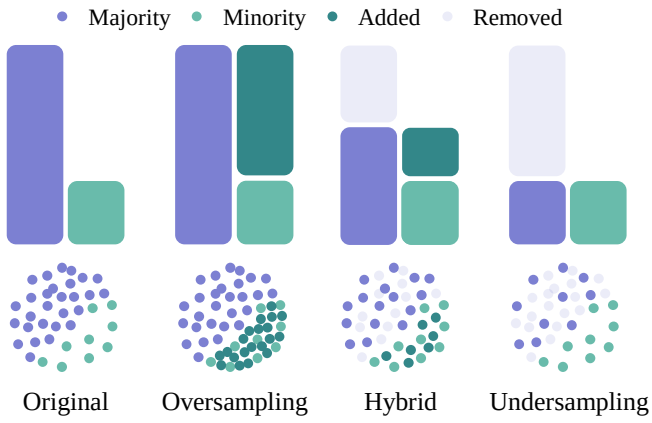


(a) (b) (c)

Source: He and Ma (2013).

There are different methods to counter the imbalance problem for enabling learning algorithms. One of the most popular and efficient is called sampling. As the name suggests, the method creates a new dataset, relatively more balanced, by sampling the original imbalanced dataset. The solution can apply three types of techniques: oversampling the minority classes; undersampling the majority classes; or both, called hybrid sampling (HE; MA, 2013). Figure 3 illustrates these techniques.

Figure 3: Sampling types for imbalanced data



Source: Created by the author.

While sampling does not solve the issue of absolute rarity, it does alleviate relative rarity. Consequently, learning algorithms can better capture decision boundaries between majority

and minority classes (HE; MA, 2013). Incidentally, applications with learning tend to apply sampling techniques in preprocessing, feeding learning algorithms with balanced datasets.

2.2 Supervised Machine Learning

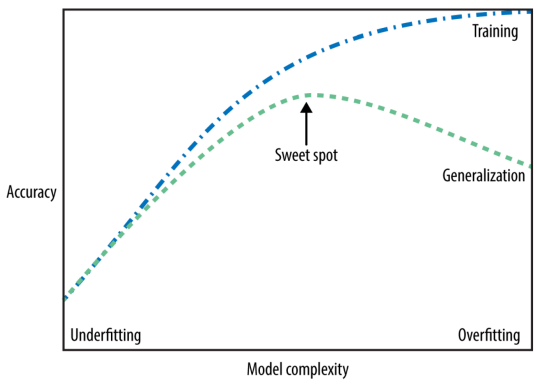
According to Géron (2017), "ML is the science of programming computers so they can learn from data". Specifically, supervised ML encompasses learning algorithms trained with examples of desired solutions to the learned task. The two most typical tasks are grouped in classification, for categorical identification, and regression, for numerical value prediction. Hence, ML enables humans to automate these tasks and make accurate predictions for new data using computer applications.

Technically, supervised ML algorithms are said to extract knowledge from a training dataset composed of a set of input features and the expected output, commonly called label. These algorithms build a ML model after learning from the training dataset, which then can be used to predict the output of unseen input data – called test datasets. A good model can generalize the training dataset in order to accurately predict the outputs of test datasets (MüLLER; GUIDO, 2017). Figure 4 illustrates an example of a training dataset for potential boat customer prediction, as well as a generalization performance for different model complexities.

Figure 4: Supervised Machine Learning: (a) example of training dataset; (b) generalization performance for different model complexities

Age	Number of cars owned	Owns house	Number of children	Marital status	Owns a dog	Bought a boat
66	1	yes	2	widowed	no	yes
52	2	yes	3	married	no	yes
22	0	no	0	married	yes	no
25	1	no	1	single	no	no
44	0	no	2	divorced	yes	no
39	1	yes	2	married	yes	no
26	1	no	2	single	no	no
40	3	yes	1	married	yes	no
53	2	yes	2	divorced	no	yes
64	2	yes	3	divorced	no	no
58	2	yes	2	married	yes	yes
33	1	no	1	single	no	no

(a)



(b)

Source: Müller and Guido (2017).

ML models can be grouped in different categories, correlated according to their learning algorithms. Classical models, such as K-Nearest Neighbors (KNN) and Decision Trees (DTs), are widely used and extensively studied over the last decades. In addition, ensemble algorithms combine multiple classical models to create more powerful and complex models. The most common and effective ensemble models derive from multiple DTs, like Random Forest (RF) and AdaBoost. Finally, Neural Networks (NN) models create probability-weighted associations based on the mechanics of the human brain. Their structure is usually customized to specific

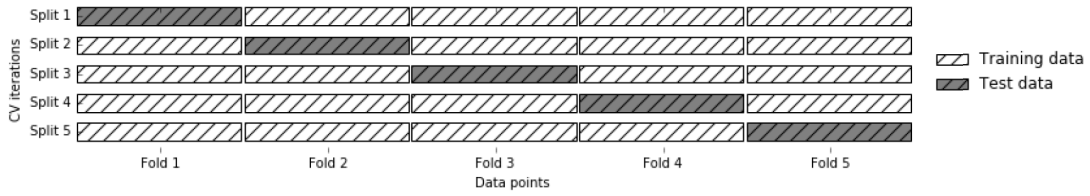
applications (MüLLER; GUIDO, 2017).

2.2.1 Model evaluation and optimization

A good model generalizes to new data. As illustrated in Figure 4b, the model needs not to perfectly fit the training dataset, but to make accurate predictions of data unobserved during training. Therefore, to estimate the performance with only training data, evaluation strategies usually partition the training dataset into two sets: *train*, used for training; and *test*, used for evaluating performance (MüLLER; GUIDO, 2017).

However, there are more robust and commonly accepted strategies for assessing the model performance, such as the statistical method of cross-validation. For instance, k-fold, one of the most commonly used algorithms, splits the training dataset into k parts of equal size – usually between five to ten. Then, the algorithm evaluates the model with the k folds, achieving a better representation of the model's performance. Figure 5 shows a visual representation of a five-fold for model evaluation (MüLLER; GUIDO, 2017).

Figure 5: Five-fold cross validation



Source: Müller and Guido (2017).

Model performance varies according to the application since each application has different priorities. In a binary classification problem, the predicted output can be True Positive (TP), True Negative (TN), False Positive (FP), or False Negative (FN). While some applications may prioritize detecting positive instances and ignoring false instances, others may prioritize avoiding false instances and catching remaining positive instances. Thus, the existing literature developed different metrics for evaluating ML models. Table 1 presents some of the most common metrics for classification problems – where every metric ranges between 0 and 1.

In addition to the metrics in Table 1, ML classification models can be better evaluated and optimized by analyzing probability curves. For instance, when a ML model predicts the output of a binary classification problem, the assigned value corresponds to the probability for belonging to the positive class – ranging between 0 and 1. ML models default to defining a probability threshold of 0.5, classifying any probability higher than the threshold as positive. Nonetheless, the ML model can be optimized by varying the probability threshold, achieving favorable results (BROWNLEE, 2021).

One method for estimating the ML model capability is calculating the Area Under the Curve (AUC) score of probability curves – such as Receiver Operating Characteristic (ROC) and

Table 1: Basic Machine Learning performance metrics for classification problems

Metric	Definition	Description
Accuracy	$acc = \frac{TP + TN}{TP + TN + FP + FN}$	Correct prediction rate. Measures the proportion of correct predictions in all instances.
Recall	$rec = \frac{TP}{TP + FN}$	True positive rate. Measures the proportion of positive instances actually assigned as positive. Also called sensitivity.
Specificity	$spe = \frac{TN}{TN + FP}$	True negative rate. Measures the proportion of negative instances actually assigned as negative. Complements recall.
Precision	$pre = \frac{TP}{TP + FP}$	Measures the proportion of assigned positive instances that are truly positive.
G-Mean	$gm = \sqrt{rec \cdot spe}$	Geometric mean. Measures the relative balance between the performances of both positive and negative classes.
F-Measure	$fm = \frac{(1+\alpha) \cdot rec \cdot pre}{\alpha \cdot pre + rec}$	Weighted harmonic mean of precision and sensitivity for any weighting factor $\alpha > 0, \alpha \in \mathbb{R}$. Also called F1-score.
IBA	$iba = [1 + \alpha \cdot (rec - spe)] \cdot M$	Index of Balanced Accuracy. Measures a metric M favoring the positive class by a weighting factor $\alpha > 0, \alpha \in \mathbb{R}$.

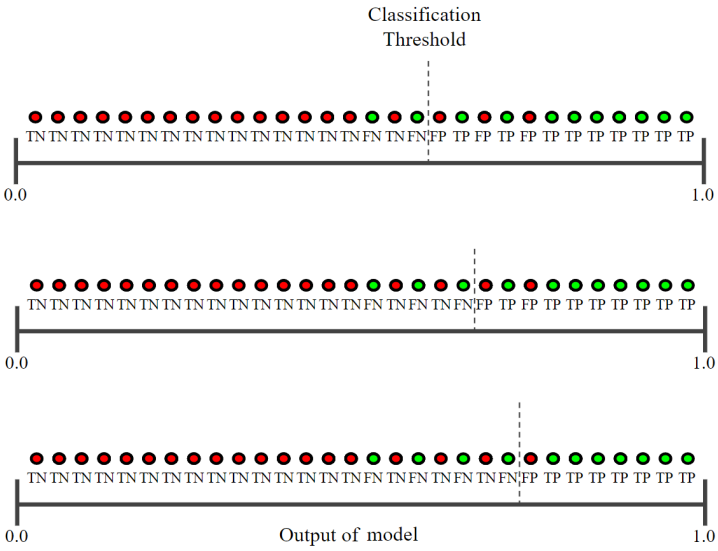
Source: Created by the author.

Precision-Recall (PR). ROC and PR curves explore the trade-off between correctly classified positive instances and misclassified negative instances for different probability thresholds (HE; MA, 2013). Figure 6 illustrates the effect of changes in the probability threshold for precision and recall. Additionally, Figure 7 shows PR curves of classifiers by varying the threshold.

The curves in Figure 7 demonstrate that the classifier C1 has a smaller AUC than C2. Consequently, the latter has better capabilities for classification, enabling optimal combinations of PR – according to the application's priorities. This resource is particularly useful for imbalanced data applications, where the ML model needs sensitivity to the minority classes without overfitting (HE; MA, 2013).

Finally, in addition to the probability threshold, every ML algorithm has different parameters for learning. For instance, Support Vector Machine (SVM) has a kernel function, a kernel bandwidth (gamma), and a regularization parameter (C). The literature presents different methods for optimizing these parameters, ranging from exhaustive search to heuristic strategies. Grid search, the most commonly used, tests a predefined set of parameters to find the best overall performance (MüLLER; GUIDO, 2017). Figure 8 shows results from a grid search for a SVM model with cross-validation.

Figure 6: Probability threshold optimization visual representation



$$\text{Precision} = \frac{TP}{TP+FP} = \frac{9}{9+3} = 0.75$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{9}{9+2} = 0.82$$

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{8}{8+2} = 0.8$$

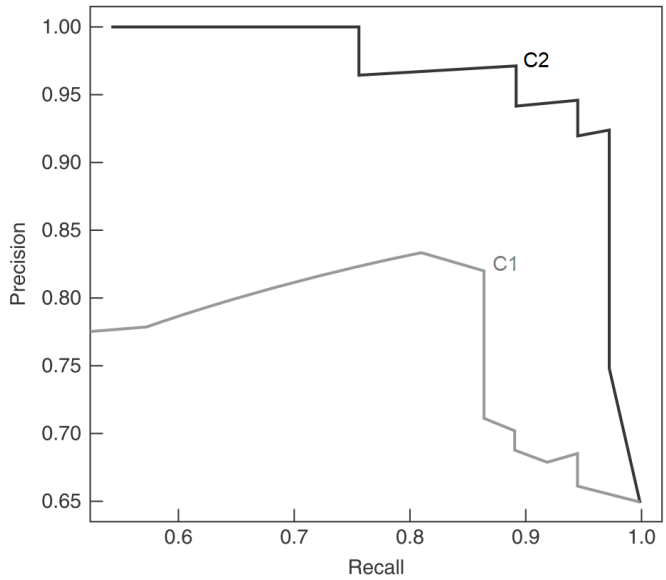
$$\text{Recall} = \frac{TP}{TP+FN} = \frac{8}{8+3} = 0.73$$

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{7}{7+1} = 0.88$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{7}{7+4} = 0.64$$

Source: Adapted from Google (2020).

Figure 7: Precision-Recall curves



Source: Adapted from He and Ma (2013).

2.2.2 Pipelines

The essential benefit of ML comes from the automation of real-life tasks. When new training data becomes available in an application, a workflow should automatically trigger the execution of processes, such as data validation, model training and deployment. Hence, current automated applications develop ML pipelines – responsible for maintaining ML models, recording historical model changes, and standardizing configurations (HAPKE; NELSON, 2020).

A pipeline is a recurring cycle. Computationally, a pipeline executes in a feedback loop,

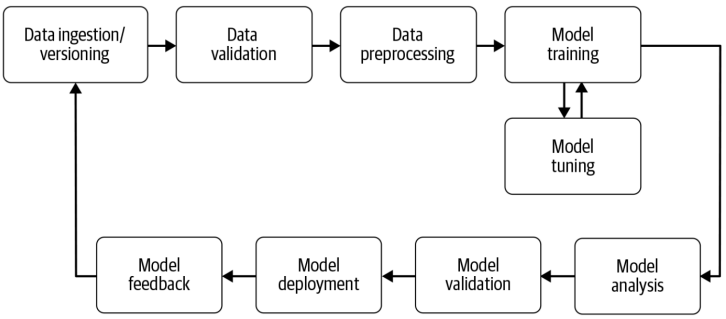
Figure 8: Grid search results for SVM with cross-validation

	0	1	2	3	...	38	39	40	41
param_C	0.001	0.001	0.001	0.001	...	0.1	1	10	100
param_gamma	0.001	0.01	0.1	1	...	NaN	NaN	NaN	NaN
param_kernel	rbf	rbf	rbf	rbf	...	linear	linear	linear	linear
mean_test_score	0.37	0.37	0.37	0.37	...	0.95	0.97	0.96	0.96
rank_test_score	27	27	27	27	...	11	1	3	3
split0_test_score	0.38	0.38	0.38	0.38	...	0.96	1	0.96	0.96
split1_test_score	0.35	0.35	0.35	0.35	...	0.91	0.96	1	1
split2_test_score	0.36	0.36	0.36	0.36	...	1	1	1	1
split3_test_score	0.36	0.36	0.36	0.36	...	0.91	0.95	0.91	0.91
split4_test_score	0.38	0.38	0.38	0.38	...	0.95	0.95	0.95	0.95
std_test_score	0.011	0.011	0.011	0.011	...	0.033	0.022	0.034	0.034

Source: Adapted from Müller and Guido (2017).

continuously creating triggers to run specific processes. A basic model cycle starts collecting, validating and preprocessing data for training. Then, the pipeline optimizes and validates the model for deployment. Finally, the model stays constantly analyzed to start a new cycle when the performance from the previous training data does not represent the current instance space, reducing the model's performance (HAPKE; NELSON, 2020). Figure 9 illustrates this cycle.

Figure 9: Basic automated Machine Learning pipeline



Source: Hapke and Nelson (2020).

Namely, the engineering design of a ML application has to break down the stages of the pipeline into manageable tasks – a ML life cycle (MCMAHON, 2021).

2.3 Final considerations

This chapter provided essential information for understanding imbalanced data, resulting issues, and how to solve them with sampling techniques. Additionally, the second section detailed supervised ML, how to evaluate and optimize models, and the foundation for developing automated ML systems.

3 RELATED WORKS

This chapter presents two studies on published papers applying ML through imbalanced data. Section 3.1 presents a systematic mapping of sampling techniques for ML applications with imbalanced data, analyzing both techniques and ML models. In addition, Section 3.2 shows two reviews of supervised ML software architectures – dividing into ones focused on imbalanced data and others for general applications. Finally, Section 3.3 discusses key aspects from these studies, highlighting research gaps and relevant features for the architecture proposed in Chapter 4.

3.1 Imbalanced data preprocessing for Machine Learning

This study, published in the journal "Knowledge And Information Systems" (VARGAS et al., 2022), has the main objective of reviewing papers solving ML in imbalanced data applications through data level preprocessing techniques. Additionally, this work details the analyzed works' domain areas and solutions – specifying current and effective sampling techniques and ML models, thus serving as a basis for future works. Structured as a systematic mapping study, the search process found 9,927 papers through seven digital libraries. From these, an eight-step filtering process selected 35 papers for analyses and discussions.

The section is organized as follows: Section 3.1.1 describes related literature reviews and this study's contribution; Section 3.1.2 details the materials and methods used in this literature review; Section 3.1.3 answers research questions, discusses results, and presents taxonomies and illustrations of the findings; and, finally, Section 3.1.4 provides conclusions and lessons learned from the study.

3.1.1 Related works

The research method described in Section 3.1.2 yielded 5 reviews and surveys addressing techniques for dealing with the imbalance problem generally (KAUR; PANNU; MALHI, 2019; FELIX; LEE, 2019; SPELMEN; PORKODI, 2018; SUSAN; KUMAR, 2020; SHAKEEL; SABHITHA; SHARMA, 2017). Additionally, 19 reviews analyzed solutions limited to specific applications (JOHNSON; KHOSHGOFTAAR, 2019; LI; MAO, 2014; BUDA; MAKI; MAZUROWSKI, 2018; BHATORE; MOHAN; REDDY, 2020; SIRSAT; FERMÉ; CÂMARA, 2020; THANOUN; YASEEN, 2020; CHUGH; KUMAR; SINGH, 2021; ISHTIAQ et al., 2020; HU et al., 2018; BENHAR; IDRI; FERNÁNDEZ-ALEMÁN, 2020; IDRI et al., 2018; LEI et al., 2020; ZHANG et al., 2021; AMARASINGHE; APONSO; KRISHNARAJAH, 2018; PRISCILLA; PRABHA, 2019; LI; JING; ZHU, 2018; PANDEY; MISHRA; TRIPATHI, 2021; MALHOTRA, 2015; GUZELLA; CAMINHAS, 2009).

This section describes general and application-limited reviews in Sections 3.1.1.1 and 3.1.1.2,

respectively. Moreover, Section 3.1.1.3 details this study's contribution.

3.1.1.1 General reviews and surveys

Kaur, Pannu and Malhi (2019) presented an in-depth literature review on the imbalanced data challenges for ML. The paper extensively details solution methods in ML, exploring preprocessing techniques, cost-sensitive learning, algorithm-centered and hybrid methods. The authors structured and analyzed works through domain areas and corresponding applications. Additionally, the authors described and compared ML algorithms applied to metrics obtained in the selected studies.

Felix and Lee (2019) reviewed published studies on preprocessing techniques for general ML applications. The work focuses on evaluating the quality of published papers, highlighting the score per data-related issues and preprocessing techniques – hence directing future works.

Spelmen and Porkodi (2018) detailed solutions from papers handling imbalanced data on both data and algorithmic levels – including hybrid models. The study describes the proposed solution and results for each work through a discussion organized by solution methods.

Susan and Kumar (2020) surveyed studies on preprocessing techniques for ML applications. The paper thoroughly describes sampling methods and how each analyzed work implemented the proposed solutions. Finally, the survey also summarizes experimental procedures, details, and reported results.

Shakeel, Sabhitha and Sharma (2017) reviewed works on preprocessing techniques for ML binary and multiclass classification. The authors briefly described classification algorithms, preprocessing, and ensemble methods.

Furthermore, the reviewed papers (KAUR; PANNU; MALHI, 2019; FELIX; LEE, 2019; SPELMEN; PORKODI, 2018; SUSAN; KUMAR, 2020; SHAKEEL; SABHITHA; SHARMA, 2017) discuss strengths, weaknesses, applications, and opportunities for future works. Table 2 outlines relevant topics of these papers: reference, data level preprocessing as the only solution method, ML-only applications, Quality Assessment (QA), and primary focus. The topic is classified as *partially* when the study covers balancing solutions different than sampling for preprocessing – such as cost-sensitive and ensemble learning, or applications without ML.

Table 2: Details of topics from related works

Work	Preproc.	ML	QA	Primary focus
Kaur, Pannu and Malhi (2019)	Part.	Yes	No	Applications and results
Felix and Lee (2019)	Yes	Part.	Yes	Quality assessment
Spelmen and Porkodi (2018)	Part.	Part.	No	Solution description
Susan and Kumar (2020)	Yes	Yes	No	Solution description and results
Shakeel, Sabhitha and Sharma (2017)	Part.	Yes	No	Classification applications

Source: Created by the author.

3.1.1.2 Application-focused reviews and surveys

The research method also found 19 reviews addressing solutions for specific imbalanced data and ML applications. These papers explore: classification algorithms (JOHNSON; KHOSHGOFTAAR, 2019; LI; MAO, 2014; BUDA; MAKI; MAZUROWSKI, 2018), credit risk evaluation (BHATORE; MOHAN; REDDY, 2020), disease diagnosis (SIRSAT; FERMÉ; CÂMARA, 2020; THANOUN; YASEEN, 2020; CHUGH; KUMAR; SINGH, 2021; ISHTIAQ et al., 2020; HU et al., 2018; BENHAR; IDRI; FERNÁNDEZ-ALEMÁN, 2020; IDRI et al., 2018), fault diagnosis (LEI et al., 2020; ZHANG et al., 2021), transaction fraud detection (AMARASINGHE; APONSO; KRISHNARAJAH, 2018; PRISCILLA; PRABHA, 2019), software defect prediction (LI; JING; ZHU, 2018; PANDEY; MISHRA; TRIPATHI, 2021; MALHOTRA, 2015), and spam filtering (GUZELLA; CAMINHAS, 2009). Furthermore, some of these papers also limit reviewed studies by the ML algorithms, covering only boosting (LI; MAO, 2014), Convolutional Neural Networks (CNN) (BUDA; MAKI; MAZUROWSKI, 2018), and Deep Learning (DL) (JOHNSON; KHOSHGOFTAAR, 2019; HU et al., 2018).

3.1.1.3 Contribution

Although there is a large number of studies addressing imbalanced data through preprocessing, Felix and Lee (2019) affirm that there is a lack of literature reviews in order to assert the reliability of the proposed techniques.

Related works mainly focus on specific applications (JOHNSON; KHOSHGOFTAAR, 2019; LI; MAO, 2014; BUDA; MAKI; MAZUROWSKI, 2018; BHATORE; MOHAN; REDDY, 2020; SIRSAT; FERMÉ; CÂMARA, 2020; THANOUN; YASEEN, 2020; CHUGH; KUMAR; SINGH, 2021; ISHTIAQ et al., 2020; HU et al., 2018; BENHAR; IDRI; FERNÁNDEZ-ALEMÁN, 2020; IDRI et al., 2018; LEI et al., 2020; ZHANG et al., 2021; AMARASINGHE; APONSO; KRISHNARAJAH, 2018; PRISCILLA; PRABHA, 2019; LI; JING; ZHU, 2018; PANDEY; MISHRA; TRIPATHI, 2021; MALHOTRA, 2015; GUZELLA; CAMINHAS, 2009). Additionally, other works focus on describing the solutions proposed by the reviewed papers (KAUR; PANNU; MALHI, 2019; SPELMEN; PORKODI, 2018; SUSAN; KUMAR, 2020; SHAKEEL; SABHITHA; SHARMA, 2017), or assessing their quality (FELIX; LEE, 2019).

Conversely, this review aims to quantitatively detail sampling techniques and ML models in imbalanced data applications. This approach centers on structuring and analyzing publication data from different domains. In this sense, the study enables the creation of two taxonomies of sampling techniques and ML models tested in the reviewed studies. Additionally, this analysis may outline novel findings on performance and correlation with domain areas.

The quantitative analysis evaluates the reliability of sampling techniques and ML models through the number and relative performance by comparing the ratio between selected and tested methods in the reviewed studies. Moreover, this study searches for simulation-based

solutions as support for future implementations.

Expanding related reviews from Table 2, this study covers both preprocessing and ML, assessing the studies' quality through answers for the Research Questions (RQs). Finally, the publication date gap may also contribute by including more recent studies. Related works covered their most recent papers from 2017 (SPELMEN; PORKODI, 2018; SHAKEEL; SABHITHA; SHARMA, 2017), 2018 (KAUR; PANNU; MALHI, 2019; FELIX; LEE, 2019), and 2020 (SUSAN; KUMAR, 2020).

3.1.2 Research method

This work applied a systematic mapping methodology for conducting an evidence-based literature review of research publications addressing preprocessing techniques for imbalanced data in ML applications. Generally used to identify, aggregate, and classify studies on the research topic, the methodology aims to be unbiased and replicable (KITCHENHAM et al., 2010; COOPER, 2016).

Oriented by the guidelines proposed by Petersen, Vakkalanka and Kuzniarz (2015), this systematic mapping defined the following procedures: (1) Research Questions; (2) Search strategy; (3) Papers filtering; and (4) Quality Assessment.

3.1.2.1 Research Questions

Accurate RQs are the key to finding a good sample of articles on a domain area (De Almeida et al., 2020). Hence, a preliminary research and analyses of the resulting articles defined this study's questions. These questions guided the discovery and characterization of studies applying sampling techniques for improving ML applications with imbalanced datasets.

This study divided RQs into three sets, shown in Table 3:

- *General Question (GQ)*: it states the main research focus;
- *Focused Questions (FQs)*: these five questions detail existing solutions in order to structure models, identify patterns, limitations, and gaps for future research;
- *Statistical Questions (SQs)*: these two questions comprise bibliography information for chronological analysis and QA.

3.1.2.2 Search strategy

The study defined three steps for the search strategy: (1) specify search string; (2) select databases; and (3) collect results. The first step identified the major terms and their most relevant synonyms – based on preliminary research and related works. Subsequently, the search string merged the major terms with their synonyms with Boolean operators. Table 4 presents the specified string.

Table 3: Research Questions

RQ#	Description
GQ1	How have preprocessing techniques been used to optimize Machine Learning from imbalanced datasets?
FQ1	What are the domain areas of Machine Learning applications with imbalanced datasets?
FQ2	Which preprocessing techniques are used to balance imbalanced datasets for Machine Learning training?
FQ3	Are there any studies that use simulation data for preprocessing imbalanced datasets?
FQ4	Which Machine Learning models are used in imbalanced data applications?
FQ5	Which development tools are used for implementing the proposed solutions? (programming language, package, or software)
FQ6	Are there any correlations between domain areas and preprocessing techniques or Machine Learning models?
SQ1	How has the quantity of studies evolved? (publications per year)
SQ2	Where have the studies been published? (type of venue and digital library)

Source: Created by the author.

The preliminary research found other combinations of search terms yielding numerous results – such as “filtering” for “preprocessing”, and “class imbalance” for “imbalanced data”. However, these synonyms created negative effects. For instance, “filtering” resulted in too many irrelevant signal noise reduction works, and “class imbalance” biased results towards general classification problems.

Table 4: Search string

Major term	Search terms
Imbalanced data	((“imbalanced data” OR “imbalanced dataset” OR “imbalanced data set” OR “unbalanced data” OR “unbalanced dataset” OR “unbalanced data set”)
	AND
Preprocessing	(preprocessing OR pre-processing OR preparation)
	AND
Machine Learning	(“machine learning” OR “deep learning” OR “artificial intelligence”))

Source: Created by the author.

Secondly, the search strategy encompassed seven digital libraries: Association for Computing Machinery (ACM), IEEE Xplore, Institution of Engineering and Technology (IET), Science Direct, Scopus, Springer Link, and Wiley. The selection of these libraries prioritized well-known research sources with multidisciplinary fields, which is essential to finding applications in various areas of knowledge – as suggested by Silva and Braga (2020).

Finally, in addition to the search string as the search query, the research applied filters for language and type of venue according to the filtering process – when available in the digital library.

3.1.2.3 Papers filtering

The collected papers went through a filtering process, removing studies unrelated to sampling techniques for ML applications. The following Exclusion Criteria (EC) supported the filtering process:

- *EC1*: the study is not written in English;
- *EC2*: the study venue is neither conference nor journal;
- *EC3*: the study matches the keywords defined in the search string, but the context is different from the research purposes;
- *EC4*: the study is a literature review (Section 3.1.1);
- *EC5*: the study is not accessible in full-text;
- *EC6*: the study is a short paper (4 pages or less);
- *EC7*: the solution focuses on algorithmic level techniques for imbalanced data;
- *EC8*: the study does not detail the sampling techniques or ML models implemented in the solution – answering FQ2 and FQ4;
- *EC9*: the study validates the proposed solution through datasets from multiple applications.

Papers filtering started at the initial search from each digital library, removing results complying with EC1 and EC2. This process did not have any date restraint, therefore collecting all results published in conferences or journals, and written in English. Then, one filter by title and one filter by abstract extracted studies meeting EC3 and EC4. After that, a combination of the remaining papers removed repeated works.

Subsequently, the original papers went through a filter based on the three-pass method (Keshav S., 2007), excluding papers complying with EC5, EC6, and EC7. Finally, a careful full-text read selected the most representative works for the research purposes. The final step rejected algorithmic level solutions, low-quality papers, and papers without a single predetermined application – meeting EC7, EC8, or EC9.

3.1.2.4 Quality Assessment

Following the scoring system proposed by Kitchenham et al. (2010), this study evaluates the selected papers' quality applying FQs 1 to 5 – since they inherently structure the research. Table 5 presents the QA scores, attributing better values for more satisfactory answers through a classification between Yes (Y), Partially (P), and No (N). Additionally, this study also presents the H-Index, year of publication, and type of venue of each paper.

Table 5: Quality scores for the answers of Research Questions

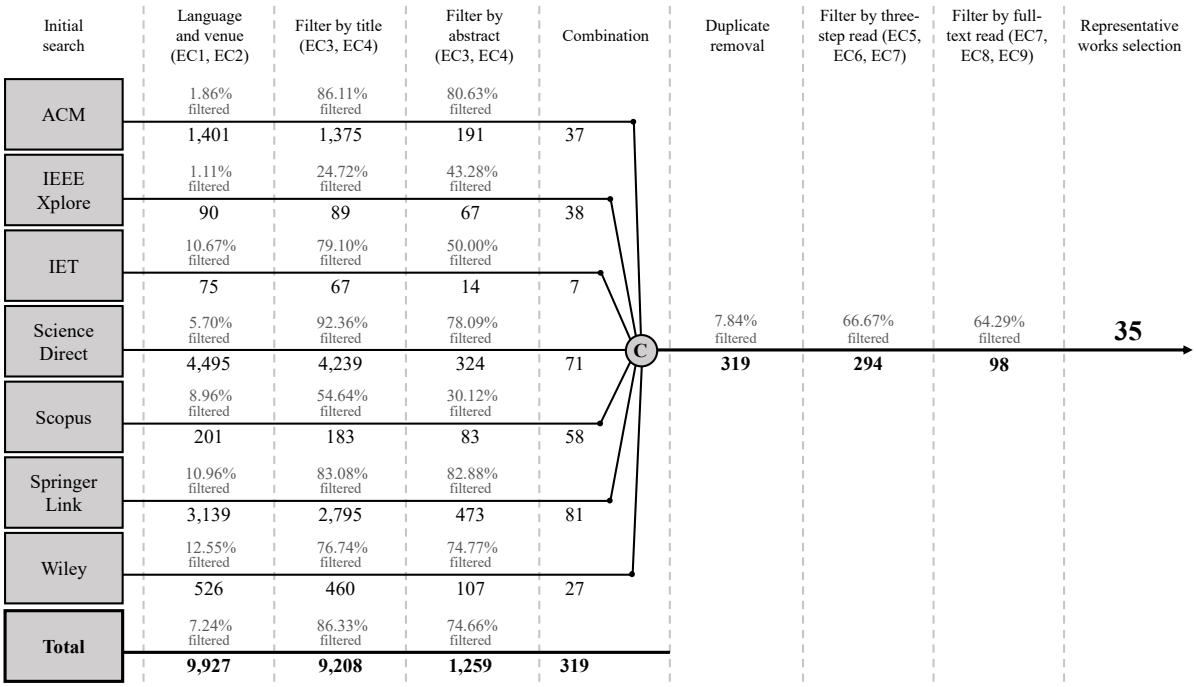
Answer	Score	Criterion
Y	1.0	The paper entirely answers the question
P	0.5	The paper partially answers the question
N	0.0	The paper does not address the topic

Source: Created by the author.

3.1.3 Results

The collection of results in all seven digital libraries integrated 9,927 studies. After an eight-step filtering process, the selection of the representative works resulted in the 35 papers indicated in the QA (Table 6). Figure 10 details the filtering process.

Figure 10: Filtering process



Source: Created by the author.

The selected studies completely answer FQs 1 to 4, so Table 6 merges their quality score in the column FQ1-FQ4. The only majorly unanswered question details development tools (FQ5). Therefore, all works have their QA between 4 and 5. This result indicates good quality papers – detailing application, sampling techniques, and ML models.

Table 6: Quality Assessment

ID	Work	Venue	FQ1-FQ4	FQ5	QA	H-Index
1	Wang and Ye (2020)	Journal	1.0	0.0	4.0	110
2	Liu, Ma and Cheng (2020)	Journal	1.0	0.0	4.0	127
3	Tra, Duong and Kim (2019)	Journal	1.0	0.0	4.0	119
4	Shamsudin et al. (2020)	Conference	1.0	0.0	4.0	20
5	Haldar et al. (2019)	Conference	1.0	0.0	4.0	-
6	Lee and Kim (2020)	Journal	1.0	1.0	5.0	68
7	Gicić and Subasi (2018)	Journal	1.0	0.0	4.0	38
8	Yan et al. (2016)	Journal	1.0	1.0	5.0	180
9	Tashkandi and Wiese (2019)	Conference	1.0	1.0	5.0	-
10	Vu et al. (2016)	Conference	1.0	0.0	4.0	-
11	Purnami and Trapsilasiwi (2017)	Conference	1.0	0.0	4.0	-
12	Chang, Lin and Liu (2019)	Conference	1.0	0.0	4.0	-
13	Dewi et al. (2020)	Conference	1.0	0.0	4.0	-
14	Zhou et al. (2020)	Conference	1.0	0.5	4.5	-
15	Zhang et al. (2019)	Journal	1.0	0.0	4.0	71
16	Rustam et al. (2019)	Journal	1.0	0.0	4.0	22
17	Santos et al. (2015)	Journal	1.0	0.5	4.5	103
18	Smiti and Soui (2020)	Journal	1.0	0.5	4.5	66
19	Mahadevan and Arock (2021)	Journal	1.0	0.5	4.5	70
20	Malhotra and Lata (2020)	Journal	1.0	1.0	5.0	43
21	Faris et al. (2020)	Journal	1.0	0.0	4.0	18
22	Han et al. (2019)	Journal	1.0	0.5	4.5	44
23	Marqués, García and Sánchez (2013)	Journal	1.0	0.0	4.0	108
24	Ma et al. (2019)	Journal	1.0	0.0	4.0	52
25	Gangwar and Ravi (2019)	Conference	1.0	1.0	5.0	-
26	Jiang and Li (2021)	Journal	1.0	1.0	5.0	130
27	Malhotra and Kamal (2019)	Journal	1.0	1.0	5.0	143
28	Cohen et al. (2006)	Journal	1.0	0.0	4.0	87
29	Filho et al. (2019)	Conference	1.0	0.5	4.5	76
30	Liu et al. (2017)	Journal	1.0	0.5	4.5	92
31	Yan et al. (2020)	Journal	1.0	0.0	4.0	184
32	Jiang et al. (2019)	Journal	1.0	0.0	4.0	4
33	Pereira et al. (2020)	Journal	1.0	0.0	4.0	102
34	Zhou (2013)	Journal	1.0	1.0	5.0	121
35	Nnamoko and Korkontzelos (2020)	Journal	1.0	0.0	4.0	87

Source: Created by the author.

3.1.3.1 GQ1: How have preprocessing techniques been used to optimize Machine Learning from imbalanced datasets?

Data preparation is fundamental for ML. Hence, several preprocessing techniques can be applied to improve the learning process in applications with imbalanced datasets.

Cohen et al. (2006) published the first study filtered in the search process. The authors proposed the use of two clustering techniques in a hybrid model: Agglomerative Hierarchi-

cal Clustering (AHC)-based oversampling and K-Means-based undersampling. Tested against Random UnderSampling (RUS) and Random OverSampling (ROS), the hybrid model achieved the most effective results with five different ML models – improving hospital-acquired (nosocomial) infection prediction.

Lee and Kim (2020) also compared RUS, ROS, and a hybrid approach (RUS+ROS) with different sampling probabilities for DL-based toxicity classification in nuclear receptor compounds. The hybrid model enhanced specificity and sensitivity without compromising accuracy for two models – SCFP and FP2VEC.

Other works also create hybrid models combining RUS and oversampling through synthetic sample generation. Mahadevan and Arock (2021) advanced ensemble learning by using RUS and Synthetic Minority Oversampling TEchnique (SMOTE). The system achieved the best results for review rating prediction in e-commerce – compared to other models. RUS+SMOTE avoided induced bias and loss of useful information.

Complementary hybrid models applied clustering techniques for undersampling with synthetic oversampling. Rustam et al. (2019) applied Edited Nearest Neighbour (ENN) and SMOTE for improving the performance of cerebral infarction detection in hospital patients through SVM. The experimental results show that the performance of SVM classifiers is improved by using these techniques – which produce better accuracy as a hybrid algorithm rather than individually.

Similarly, Chang, Lin and Liu (2019) implemented hybrid sampling with ENN and ADAptive SYNthetic sampling (ADASYN) for enhancing the sensitivity of fraud identification in telephones through Stacked-SVM. Han et al. (2019) developed a credit scoring solution preprocessed by a Gaussian Mixture Model (GMM)-based majority undersampling and SMOTE. Based on tested ML metrics with both Logistic Regression (LR) and Decision Trees (DT), the authors assessed that the proposed algorithm generally performs better than eleven standard sampling algorithms.

Marqués, García and Sánchez (2013) also proposed credit scoring solutions by testing eight undersampling and oversampling techniques with LR and SVM. The authors concluded that oversampling generally outperforms undersampling for both ML models. Following a congruent path, Pereira et al. (2020) compared eight well-known sampling techniques in order to identify COVID-19 from a record of chest X-Ray images. The most effective combination results from ENN with a Multi-Layer Perceptron (MLP) model.

Vu et al. (2016) tested different techniques for encrypted network traffic identification. The study shows that ConDensed Nearest Neighbour (CDNN) and SVM-based SMOTE (SVM-SMOTE) performed the best as undersampling and oversampling techniques, respectively. However, both techniques proved to be slow compared to simpler algorithms, such as RUS, ROS, and SMOTE. Correspondingly, Shamsudin et al. (2020) also achieved one of the highest precision and recall with a hybrid model between SVM-SMOTE and RUS – for credit card fraud detection with Random Forest (RF).

Haldar et al. (2019) addressed epilepsy detection by applying the hybrid sampling technique Selective Preprocessing of Imbalanced Data, also known as SPIDER, with three different ML models. The results showed that SPIDER with the K-Nearest Neighbours (KNN) classifier achieved the best performance.

Malhotra published two studies on software source code problems (MALHOTRA; KAMAL, 2019; MALHOTRA; LATA, 2020). The first, with Kamal, implements a modified version of the SPIDER2 algorithm, called SPIDER3. The proposed solution for software defect prediction performed better than SPIDER2 and the original SPIDER. However, ADASYN achieved the best average results in combination with five ML models (MALHOTRA; KAMAL, 2019)

In addition, Malhotra and Lata (2020) performed an empirical study for selecting the best well-known sampling techniques and ML models for software maintainability prediction. After conducting tests with fourteen techniques and eight models, the authors found that Safe Level SMOTE (SL-SMOTE) significantly outperformed other techniques. The study also achieved relevant results with hybrid sampling between ENN and SMOTE, as well as Tomek Links (TL) and SMOTE.

Ma et al. (2019) improved SL-SMOTE through an evolutionary optimization process for the algorithm's parametrization. The solution, named Evolutionary SL-SMOTE (ESL-SMOTE), achieved the highest metrics for seminal quality prediction with AdaBoost against related works. Additionally, the results indicate that the preprocessing technique achieves good recall for other models – such as Back Propagation Neural Networks (BPNN) and SVM.

Five works applied only SMOTE for improving ML and retaining superior overall results (YAN et al., 2016; PURNAMI; TRAPSILASIWI, 2017; DEWI et al., 2020; ZHANG et al., 2019; GICIĆ; SUBASI, 2018). Yan et al. (2016) achieved good results for lung cancer recurrence prediction with Gaussian Radial Basis Function Network (GRBFN). Moreover, Purnami and Trapsilasiwi (2017) advanced breast cancer malignancy classification from biopsy records through Least Squares SVM (LS-SVM).

Another two SMOTE-focused studies used SMOTE in biology applications. Dewi et al. (2020) improved stability of patchouli (flowering plants) classification with Extreme Learning Machine (ELM). Additionally, Zhang et al. (2019) achieved higher accuracy for Protein-Protein Interactions (PPI) hot spots identification than related works through SMOTE and RF.

Gicić and Subasi (2018) applied SMOTE in order to improve credit scoring for micro-enterprises of the minority class (poor). After preprocessing at 100% and 200% of the minority sample and testing with fifteen classical and ensemble ML models, the authors concluded that the minority classification improved significantly and retained superior results overall.

Tra, Duong and Kim (2019) introduced a solution for diagnosing fault symptoms in the insulation oil of power transformers. The authors implemented an algorithm for improving SMOTE by estimating a local reachability distance of the majority and minority samples with two clusters. The Adaptive SMOTE (ASMOTE) algorithm achieved a higher classification

accuracy than ROS and SMOTE with the proposed MLP model.

Comparably, Jiang and Li (2021) improved fault detection in wind turbines by combining Dependent Wild Boostrap (DWB) with SMOTE (DWB-SMOTE). Since wind buffers have multivariate time series of sensors from several subsystems, the proposed CNN model generated better temporal-dependent synthetic samples and, consequently, better results.

Faris et al. (2020) tested various oversampling techniques and ML models in order to predict companies' financial bankruptcy through financial and non-financial records. After analyzing the results, the authors concluded that SMOTE with AdaBoost achieved promising and reliable predictions.

A modified version of SMOTE, called BorderLine SMOTE (BL-SMOTE), focuses on synthetic sample generation at the boundary between classes. Smiti and Soui (2020) proposed this technique for companies' financial bankruptcy prediction through DL. Jiang et al. (2019) also applied BL-SMOTE for heartbeat classification through electrocardiograms with CNN. Both works achieved the best results with BL-SMOTE.

Santos et al. (2015) implemented a clustering-based oversampling approach through K-Means++ and SMOTE for hepatocellular carcinoma survival prediction. ML with Artificial Neural Networks (ANN) and LR presented significantly better results than without clustering or oversampling. Alternatively, Tashkandi and Wiese (2019) applied K-means++ for undersampling. The results indicated an improvement in the prediction accuracy of mortality risk prediction in Intensive Care Units (ICUs) through different classical and ensemble ML models.

Zhou et al. (2020) undersampled standard features for lower back pain early diagnosis through K-Means clustering – testing both stratified sampling and Manhattan distance. In general, these techniques improved the performance of all tested models for different k values.

Three papers proposed synthetic oversampling through a modern technique based on ML, called Generative Adversarial Networks (GAN) (LIU; MA; CHENG, 2020; GANGWAR; RAVI, 2019; YAN et al., 2020). Liu, Ma and Cheng (2020) developed GAN for balancing individual and fused sensor data of rotating machinery, such as bearing and gearbox. After learning with a multi-class CNN, the proposed techniques showed effective results in a wide range of IRs.

In addition, Gangwar and Ravi (2019) applied GAN and Wasserstein GAN (WGAN) oversampling for a highly imbalanced dataset of credit card transactions. According to the authors, the results against ROS, SMOTE, and ADASYN indicate that GAN-based methods control FP spectacularly without affecting TP – which is essential for imbalanced data applications.

Yan et al. (2020) implemented a Conditional WGAN (CWGAN) framework for multi-class air handling units' fault detection. Combined with quality control of the synthetic samples, the solution improved results from different ML classifiers – reaching an accuracy of almost 1 for every model.

Data spatial distribution is important for optimized classification. Therefore, Wang and Ye (2020) implemented a spatial distribution-based sample generation for balancing historical and simulated power system stability data. The solution classifies distance intervals through KNN

and creates properly distributed synthetic data through SMOTE – which feeds a Deep Neural Network (DNN) for evaluating transient stability.

Nnamoko and Korkontzelos (2020) also created an optimized version of SMOTE for enhancing diabetes prediction. The algorithm uses the InterQuartile Range (IQR) technique for oversampling dispersed/extreme data before SMOTE, improving the training sample distribution. According to the authors, IQR+SMOTE consistently produced the best accuracy for different models and maintained the best overall metrics.

Liu et al. (2017) introduced a Fuzzy-based OverSampling (FOS) algorithm for balancing tweets' data in spam detection – optimizing the distribution in synthetic sampling. The method improved precision for different ensemble learning models. However, ROS and RUS achieved better accuracy.

Filho et al. (2019) studied automated essay scoring through ML regression and classification for Brazil's National High School Examination (ENEM). After testing SMOTE, ADASYN, ROS, and RUS, the authors concluded that random sampling performs better because the employed vectorization for feature extraction has unusual spatial characteristics.

Lastly, Zhou (2013) tested different preprocessing techniques in order to enhance corporate bankruptcy prediction through ML. The authors concluded that there is no significant difference between the results of oversampling and undersampling with large amounts of data – for instance, in a dataset of USA companies from 1981 to 2009. However, the computational time is better in undersampling. When there is not much data, SMOTE performs the best overall. Additionally, GMM-based undersampling and RUS are better than Cluster Centroid (CC).

3.1.3.2 FQ1: What are the domain areas of Machine Learning applications with imbalanced datasets?

There are 5 central domain areas for 31 of the reviewed works: health, finance, engineering, software, and biology. Additionally, 4 works are from other areas – classified as *others*. Table 7 summarizes the domain areas and corresponding applications.

Health is the most prevalent domain, accounting for 12 studies. These studies differ in their application and type of classification. For instance, the 3 cancer-related works classify breast cancer malignancy (PURNAMI; TRAPSILASIWI, 2017), predict hepatocellular carcinoma survival (SANTOS et al., 2015), and predict lung cancer recurrence (YAN et al., 2016).

Nosocomial studies spread even more, proposing solutions for predicting risk of mortality in ICUs (TASHKANDI; WIESE, 2019) and nosocomial infections (COHEN et al., 2006), classifying heartbeats (JIANG et al., 2019), detecting cerebral infarction (RUSTAM et al., 2019) and COVID-19 (PEREIRA et al., 2020). Other health domain works introduce solutions such as detection of epileptic seizure (HALDAR et al., 2019) and lower back pain (ZHOU et al., 2020), as well as prediction of semen quality (MA et al., 2019) and diabetes (NNAMOKO; KORKONTZELOS, 2020).

Table 7: Domain areas of reviewed applications

Domain	Subdomain	Application	ID
Health (34.3%)	Cancer (8.6%)	Lung cancer recurrence	8
		Breast cancer malignancy	11
		Hepatocellular carcinoma survival	17
	Hospital (14.3%)	Risk of mortality in ICUs	9
		Cerebral infarction	16
		Nosocomial infections	28
		Heartbeats	32
		COVID-19	33
	Others (11.4%)	Epileptic seizure	5
		Lower back pain	14
		Seminal fluids quality	24
		Diabetes	35
Finance (22.9%)		Companies bankruptcy	18, 21, 34
		Credit cards fraud	4, 25
		Credit risk	7, 22, 23
Engineering (14.3%)	Fault (14.3%)	Power systems stability	1
		Rotating machinery	2
		Power transformers	3
		Wind turbines	26
		Air handling units	31
Biology (8.6%)		Nuclear receptor compounds toxicity	6
		Flowering plants species	13
		PPI hot spot	15
Software (8.6%)	Source code (5.7%)	Maintainability	20
		Defect	27
	Others (2.9%)	Network traffic data	10
Others (11.4%)		Telephone fraud	12
		E-commerce products rating	19
		Essay score	29
		Spam in tweets	30

Source: Created by the author.

Finance, on the other hand, deals with cost-effective correlated problems. Representing 8 works, they predict companies bankruptcy (SMITI; SOUI, 2020; FARIS et al., 2020; ZHOU, 2013), credit risk (GICIĆ; SUBASI, 2018; HAN et al., 2019; MARQUÉS; GARCÍA; SÁNCHEZ, 2013), and credit card fraud (SHAMSUDIN et al., 2020; GANGWAR; RAVI, 2019).

Similarly, engineering studies propose solutions for fault diagnosis in different electrical and mechanical engineering applications. Accounting for 5 works, the solutions improve stability in power systems (WANG; YE, 2020), wind turbines (JIANG; LI, 2021), power transformers (TRA; DUONG; KIM, 2019), rotating machinery (LIU; MA; CHENG, 2020), and air handling units (YAN et al., 2020).

Furthermore, there are 3 papers related to software. These works improve ML for source code maintainability (MALHOTRA; LATA, 2020) and defect prediction (MALHOTRA; KA-

MAL, 2019), as well as network traffic data classification (VU et al., 2016).

Biology also accounts for 3 studies. These studies introduce nuclear receptor compounds toxicity prediction (LEE; KIM, 2020), flowering plants species (DEWI et al., 2020) and PPI hot spot classification (ZHANG et al., 2019).

Finally, 4 papers from other areas deal with telephone fraud detection (CHANG; LIN; LIU, 2019), e-commerce rating prediction (MAHADEVAN; AROCK, 2021), essay score classification (FILHO et al., 2019), and spam detection in tweets (LIU et al., 2017).

Dataset characteristics – such as features and IR – differ for each subdomain according to its applications. The reviewed studies do not always explore these characteristics, difficulting a comparative analysis. Specifically, some applications do not have enough data to infer the exact IR. Therefore, 4 works overcame this problem and generalized their solution by manually testing different IRs (WANG; YE, 2020; LIU; MA; CHENG, 2020; LEE; KIM, 2020; MARQUÉS; GARCÍA; SÁNCHEZ, 2013).

Moreover, every domain area has particularities in its applications, demanding specialized preprocessing procedures before sampling. For instance: time series data in engineering (JIANG; LI, 2021; LIU; MA; CHENG, 2020; YAN et al., 2020; WANG; YE, 2020) and health (JIANG et al., 2019); image processing in health (PEREIRA et al., 2020; COHEN et al., 2006; YAN et al., 2016) and biology (DEWI et al., 2020); text processing in other areas (MAHADEVAN; AROCK, 2021).

3.1.3.3 FQ2: Which preprocessing techniques are used to balance imbalanced datasets for Machine Learning training?

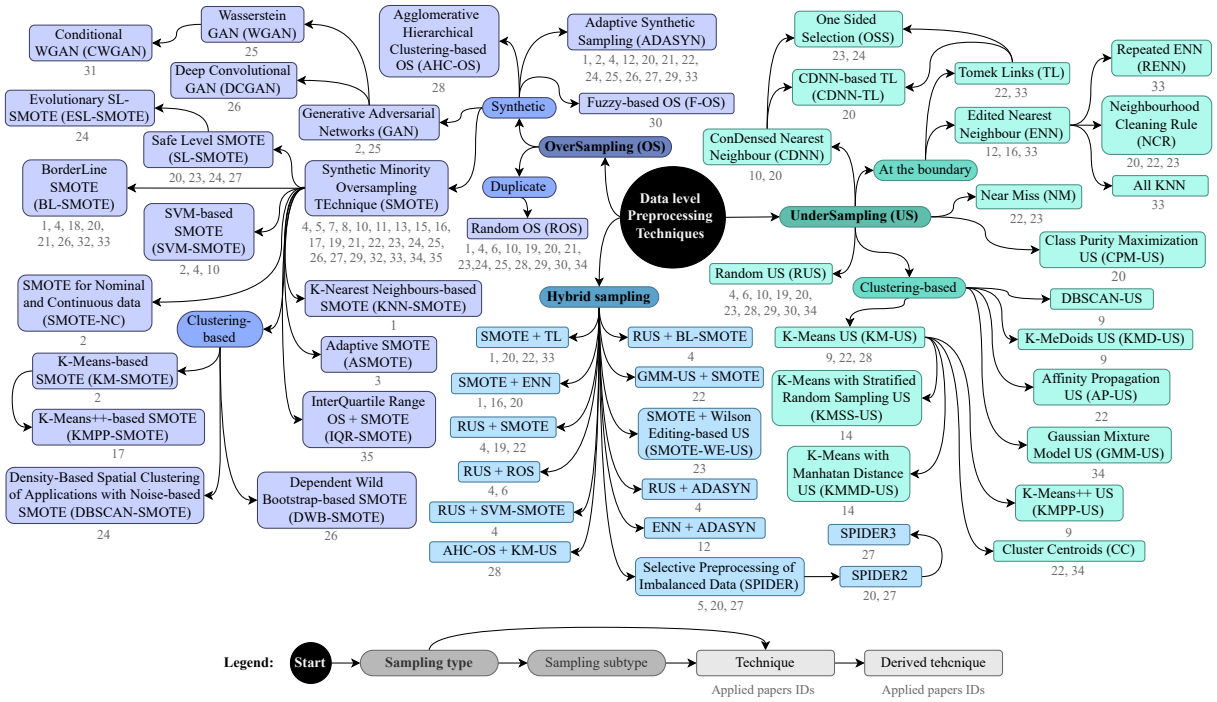
The literature covers a wide variety of preprocessing techniques for ML applications with specific characteristics and applications. This question focuses on sampling techniques for balancing datasets before ML training. Consequently, preprocessing techniques for other purposes, such as feature extraction, image, and natural language processing are not answered in this section.

Some of the reviewed studies propose a sampling technique and compare them with alternatives. Conversely, other reviews implement empirical analyses comparing several techniques to discuss results and select the best one(s). Therefore, this systematic mapping classified the techniques applied in each paper between *proposed*, *compared* and *selected*.

Figure 11 shows a taxonomy of all sampling techniques, either proposed or compared in the reviewed papers – indicated by ID below the corresponding box. The taxonomy divides these algorithms into three types: *oversampling*, *undersampling*, and *hybrid sampling*. Each algorithm is distributed according to its parent technique or type.

Additionally, Figure 12a details the number of papers applying each technique in three columns: proposed, compared, and selected. The figure presents the most used techniques as darker, while less used as lighter – based on a grayscale. These techniques are grouped by

Figure 11: Taxonomy of sampling techniques proposed or compared in reviewed papers by ID



Source: Created by the author.

their type and subtipes, following the taxonomy in Figure 11. The rightmost column indicates the percentage ratio of selected to proposed and compared techniques (relative performance).

The distribution of techniques in Figure 12a shows a more significant interest in oversampling and hybrid sampling for the proposed solutions. The studies frequently compare results with standard oversampling and undersampling techniques – such as SMOTE, ADASYN, ROS, and RUS. Namely, each has at least 10 implementations.

Techniques focused at the boundary between classes are also popular (BL-SMOTE, ENN, and TL). Additionally, clustering-based algorithms are common in both oversampling and undersampling. For instance, AHC, KM, and DBSCAN have implementations in both. Nevertheless, clustering is more frequent in undersampling due to the grouping behavior.

In addition, the distribution of selected methods indicates growth in hybrid sampling and a decrease in oversampling and undersampling – relative to the proposed methods. Specifically, 13 of 35 papers tested hybrid sampling, out of which 9 (69.2%) were the best performing sampling type (SHAMSUDIN et al., 2020; HALDAR et al., 2019; LEE; KIM, 2020; CHANG; LIN; LIU, 2019; RUSTAM et al., 2019; MAHADEVAN; AROCK, 2021; HAN et al., 2019; MARQUÉS; GARCÍA; SÁNCHEZ, 2013; COHEN et al., 2006). In the remaining 4 papers, hybrid sampling is outperformed by oversampling with KNN-SMOTE (WANG; YE, 2020), SL-SMOTE (MALHOTRA; LATA, 2020), and ADASYN (MALHOTRA; KAMAL, 2019), and by undersampling with ENN (PEREIRA et al., 2020).

However, oversampling remains the most selected sampling method, proportionally. Overall, synthetic sample generation techniques have the best performance, either individually or

in hybrid models. The selected methods are composed of modified SMOTE algorithms in 12 (34.3%), standard SMOTE in 10 (28.6%), ADASYN in 2 (5.7%), and AHC-OS in 1 (2.9%). Finally, GAN-based oversampling performed as the best techniques in 3 out of 4 papers (75%) – with GAN (LIU; MA; CHENG, 2020), WGAN (GANGWAR; RAVI, 2019), and CWGAN (YAN et al., 2020).

Three works indicate the need for testing different sampling techniques (VU et al., 2016; MALHOTRA; KAMAL, 2019; LIU et al., 2017). More specifically, 3 other works name the need for testing GAN-based oversampling (WANG; YE, 2020; LEE; KIM, 2020; JIANG; LI, 2021), since related studies achieved good results. Reviewed studies also support GAN-based approaches in their conclusions. Liu, Ma and Cheng (2020) claim that GAN improves experimental accuracy as the IR increases when compared to other sampling techniques. Gangwar and Ravi (2019) assert that WGAN outperforms GAN due to having a better objective function, as well as envision investigating different generator architectures for improving results even more.

Concerning the importance of hybrid sampling, 2 studies affirm that this is the best sampling type for improving the classification of imbalanced data (SHAMSUDIN et al., 2020; MAHADEVAN; AROCK, 2021). According to both studies, undersampling alone causes loss of information, while oversampling alone might cause induced bias or overfitting – especially in highly imbalanced datasets.

3.1.3.4 FQ3: Are there any studies that use simulation data for preprocessing imbalanced datasets?

There is only one study using simulation data – on power system stability (WANG; YE, 2020). However, the authors used simulated data for training and testing, not as preprocessing support for real-world data tests.

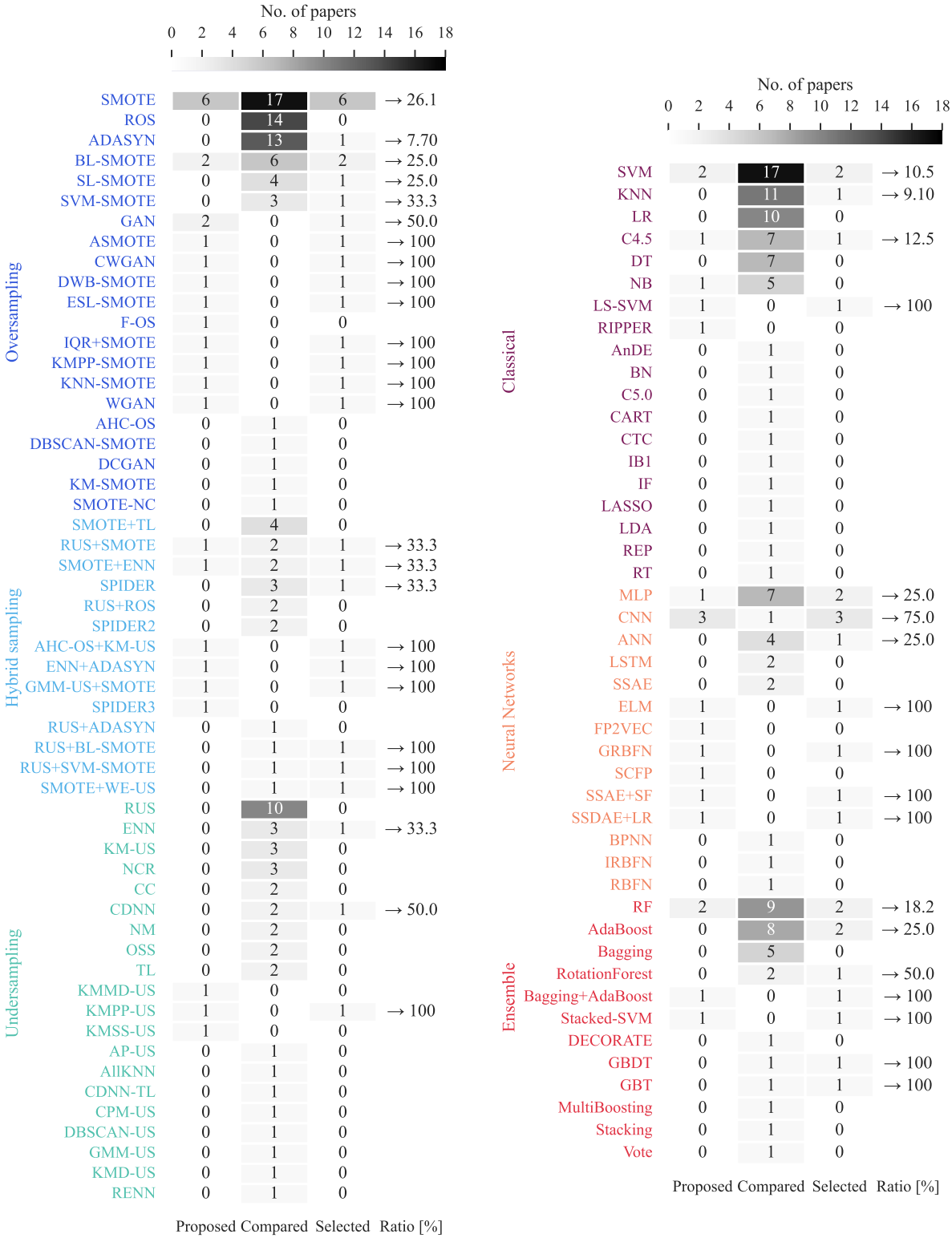
Even so, some of the domain areas have potentially applicable simulators for synthetic data generation. For instance, electrical and mechanical engineering have fault simulators, and health has exam simulators to this end.

From the 35 studies, 28 (80%) selected solutions based on synthetic oversampling (SMOTE, ADASYN, AHC-OS, and GAN). Thus, using simulation data in suitable domain areas can represent a means for optimizing results and accelerating training time. This acceleration is essential due to the high computational cost for synthetic data generation.

3.1.3.5 FQ4: Which Machine Learning models are used in imbalanced data applications?

Similar to preprocessing techniques, the studied works test a wide variety of ML models to improve predictions. From the 35 works, 15 (42.9%) propose a specific model and compare it against alternatives. Conversely, 20 works (57.1%) implement empirical analyses comparing

Figure 12: Quantitative analysis of the reviewed papers proposing, comparing, and selecting: (a) sampling techniques; (b) Machine Learning models

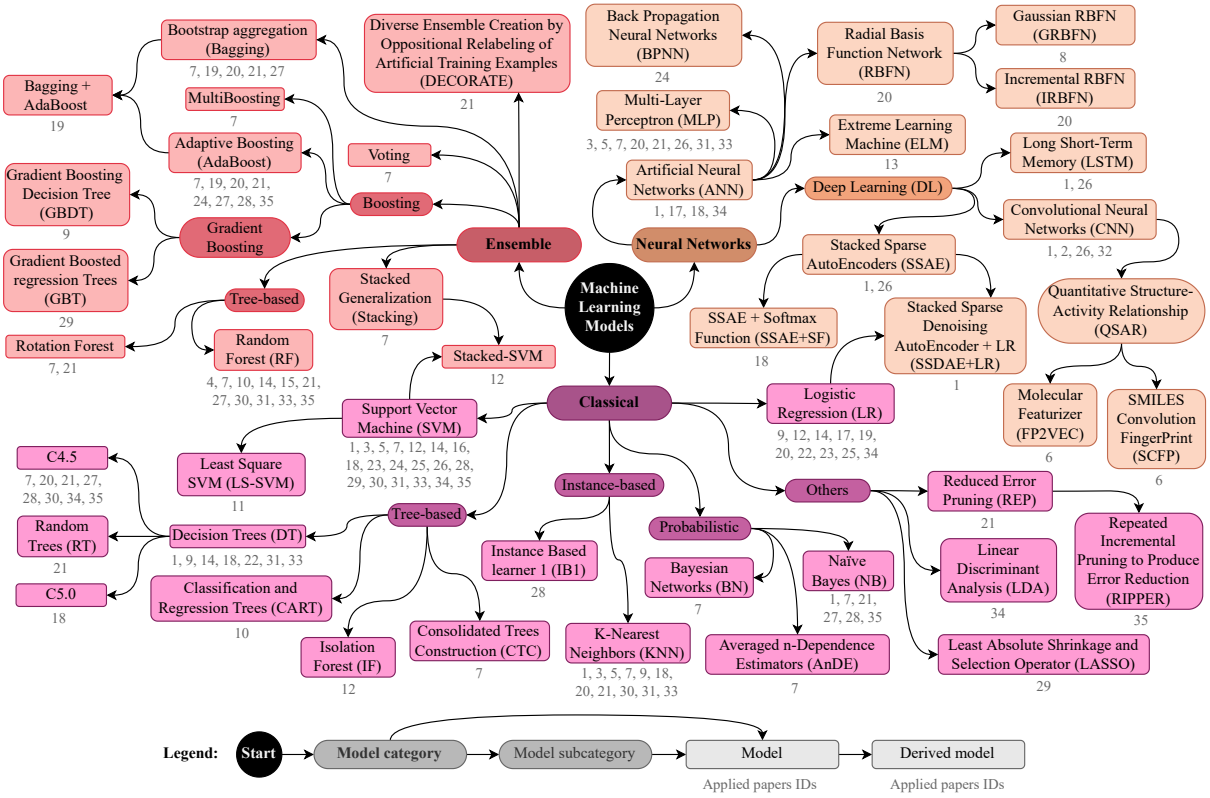


Source: Created by the author.

multiple ML models to discuss results and select the best one.

Hence, following the method applied to sampling techniques, this review proposes a taxonomy and a quantitative description of all ML models from the reviewed studies. The taxonomy in Figure 13 allocates models by their category, dividing into *classical*, *NN*, and *ensemble*. The figure also indicates the ID of papers applying each model below the corresponding box. Moreover, Figure 12b details the number of papers for each ML model – grouping by the corresponding category, classifying between *proposed*, *compared*, and *selected*, as well as showing the relative performance.

Figure 13: Taxonomy of Machine Learning models proposed or compared in reviewed papers by ID



Source: Created by the author.

The distribution of models in Figure 12b indicates a substantial interest in classical supervised learning models for empirical studies – such as SVM, KNN, LR, DT, and NB. Even so, most of the proposed ML models involve ANN and CNN optimally configured for the application (LIU; MA; CHENG, 2020; TRA; DUONG; KIM, 2019; JIANG; LI, 2021; JIANG et al., 2019). This is specially noticeable in works of singularly used models, such as SSDAE+LR (WANG; YE, 2020), SCFP and FP2VEC (LEE; KIM, 2020), GRBFN (YAN et al., 2016), ELM (DEWI et al., 2020), and SSAE+SF (SMITI; SOUI, 2020).

Comparatively, 6 out of 7 studies (85.7%) testing both NN and classical models achieved better performance with NN models (WANG; YE, 2020; TRA; DUONG; KIM, 2019; SANTOS et al., 2015; SMITI; SOUI, 2020; JIANG; LI, 2021; PEREIRA et al., 2020). Additionally, 5

NN models achieved the best performance when not compared with classical models (LIU; MA; CHENG, 2020; LEE; KIM, 2020; YAN et al., 2016; DEWI et al., 2020; JIANG et al., 2019). Conversely, 4 classical models achieved the best performance when not tested against NN models (PURNAMI; TRAPSILASIWI, 2017; RUSTAM et al., 2019; COHEN et al., 2006; NNAMOKO; KORKONTZELOS, 2020) – besides the 1 out of 7 studies that did and performed better (HALDAR et al., 2019).

Ultimately, ensemble models correspond to 9 (25.7%) of the best-performing out of 35 papers. The results in Figure 12b indicate that RF, AdaBoost, and Bagging are frequently applied – even with preprocessed imbalanced data.

The superiority of NN and ensemble models is noticeable in the studies' conclusions, mentioning the lack of these model categories as a limitation. Incidentally, 4 works expect to apply NN models in future implementations (FILHO et al., 2019; YAN et al., 2020; JIANG et al., 2019; PEREIRA et al., 2020). Additionally, 4 works want to apply ensemble models (SHAMSUDIN et al., 2020; FARIS et al., 2020; MALHOTRA; KAMAL, 2019; FILHO et al., 2019).

Finally, Jiang et al. (2019) argue the importance of evaluating the most meaningful metrics for improving imbalanced datasets – since many studies only consider the system's accuracy. Different applications have different priorities. For instance, Haldar et al. (2019) focus on improving the sensitivity of the minority class while sufficiently preserving the accuracy in epileptic seizure detection (health). In applications such as disease detection, it is better to guarantee all TPs (diagnoses) possible, even though this creates more FPs.

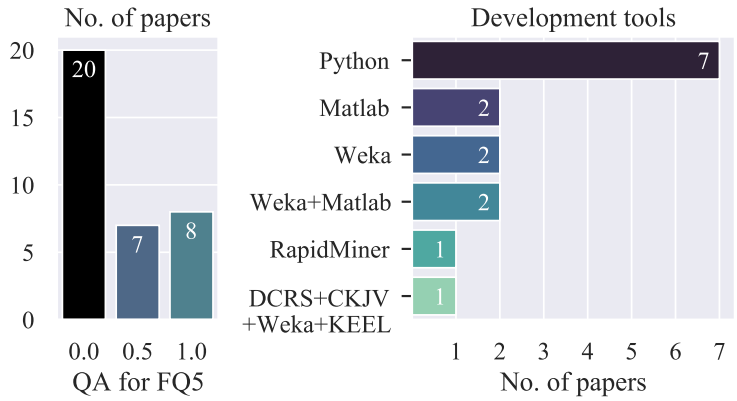
3.1.3.6 FQ5: Which development tools are used for implementing the proposed solutions?

Figure 14 shows the development tools applied for implementing the studies' solutions. The difficulty in answering this FQ is that most papers do not report any tools used for data processing and ML. These papers account for 20 works (57.1%). The remaining 15 papers report using at least one tool for data preprocessing, ML training, and testing. The completion of this answer – such as programming language, package, and software – corresponds to the QA score for FQ5 in Table 6.

The programming language Python is the most used tool, with ML models through the packages Scikit-learn (GANGWAR; RAVI, 2019; FILHO et al., 2019), Keras (JIANG; LI, 2021), Tensorflow and Chainer (LEE; KIM, 2020). Additionally, text data applications use natural language processing packages, such as SpaCy (MAHADEVAN; AROCK, 2021), NLPNET and NLTK (FILHO et al., 2019). Other use cases implement sampling techniques through Imbalanced-learn (GANGWAR; RAVI, 2019) and user-developed scripts (ZHOU et al., 2020; HAN et al., 2019).

Another programming language applied in the studies is Matlab. Two studies employ the language for implementing both preprocessing and ML models (SANTOS et al., 2015; SMITI; SOUI, 2020). In contrast, 2 studies create a test system with Matlab in conjunction with standard

Figure 14: Development tools of reviewed applications



Source: Created by the author.

ML models from the software Weka (MALHOTRA; KAMAL, 2019; ZHOU, 2013). Moreover, other solutions use only Weka for all experiments – such as feature selection, sampling, ML training, and testing (YAN et al., 2016; LIU; MA; CHENG, 2020).

Tashkandi and Wiese (2019) compared solutions with the software RapidMiner Studio – combining preprocessing, modeling, training, and testing. Finally, Malhotra and Lata (2020) created a testing system with the following tools: Data Collection and Reporting System (DCRS) tool for data extraction through GIT repositories; Chidamber and Kemerer Java Metrics (CKJV) tool for object-oriented metrics in Java source codes; Weka for outlier analysis through IQR; Knowledge Extraction based on Evolutionary Learning (KEEL) tool for sampling techniques and ML.

3.1.3.7 FQ6: Are there any correlations between domain areas and preprocessing techniques or Machine Learning models?

Generally, the 5 central domain areas and *others* – segmented in Section 3.1.3.2 – applied distinctive sampling techniques and ML models in their solutions. Figure 15 details the number of sampling techniques and ML models selected by the authors of at least one paper within the corresponding domain areas. Additionally, studies which did not select and clearly indicate at least one best performing method for the application have not been accounted for – such as ML models in software (Figure 15b).

Studies on health applied the most diverse methods, potentially due to the substantial proportion of works (34.3%). This domain is the only one applying classical ML models. Additionally, health is the only domain selecting pure undersampling techniques – apart from one study on software (VU et al., 2016).

Finance, the second most prevalent domain (22.9%), splits between using oversampling and hybrid sampling techniques. However, for ML categories, 75% of the works indicate a preference for ensemble models. In contrast, one work implements a specialized DL model

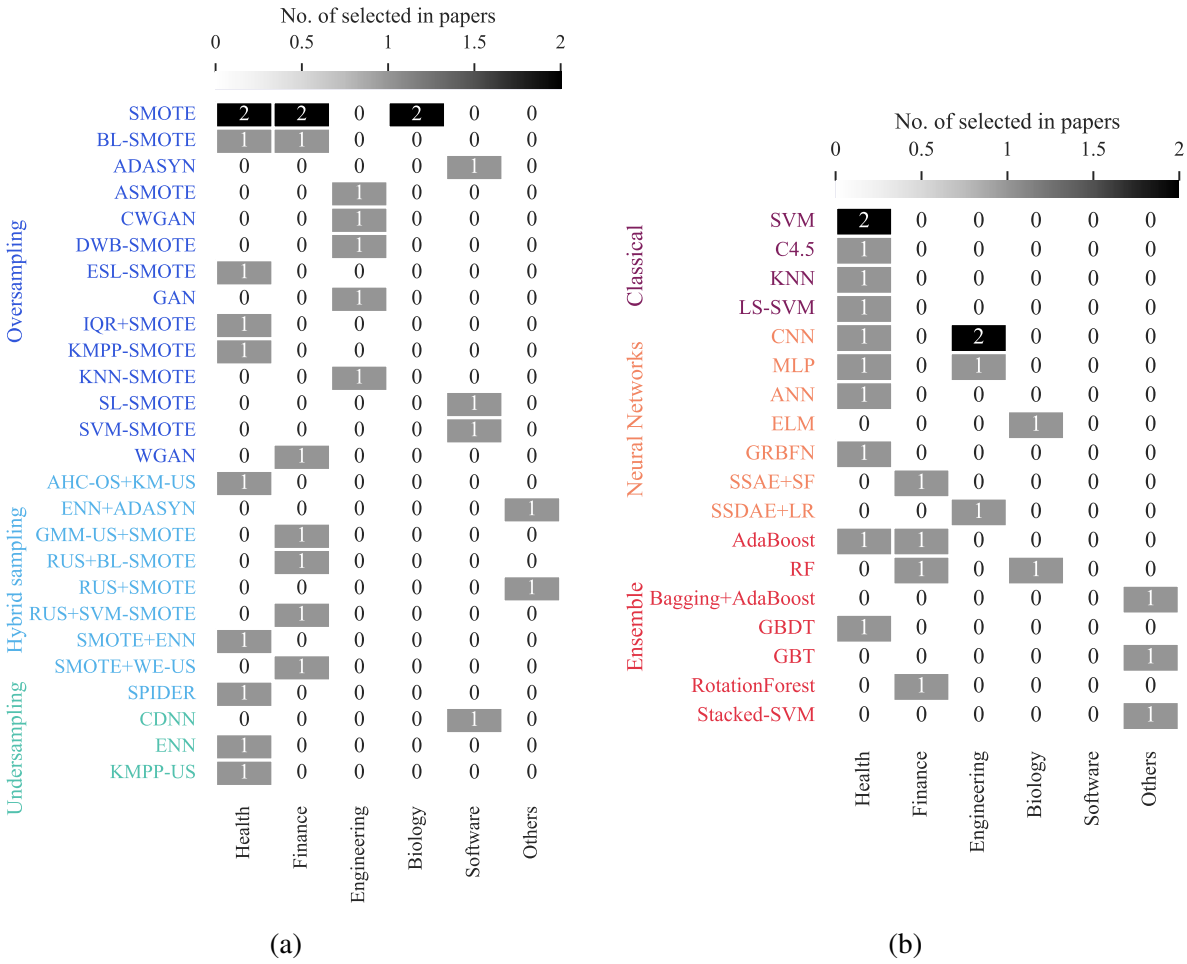
for bankruptcy prediction (SSDAE +SF) – although it does not compare results with ensemble models (SMITI; SOUI, 2020).

Engineering studies (14.3%) selected an unanimous combination of methods: oversampling and NN models. This domain has all applications related to fault detection – generally suffering from high IR and benefiting from oversampling techniques.

Similarly to engineering, biology studies (8.6%) also selected only oversampling – through SMOTE. Additionally, two studies split ML between NN, with ELM, and ensemble, with RF.

Software studies did not select any best performing ML model. However, the three studies (8.6%) achieved their best results through oversampling. One implementation, by Vu et al. (2016), points similar performance between SMOTE-SVM and CDNN (undersampling) – in less processing time with the latter. Finally, all studies in other areas (11.4%) indicate better performance through hybrid sampling and ensemble models.

Figure 15: Quantitative analysis in different domain areas for the selection of: (a) sampling techniques; (b) Machine Learning models



Source: Created by the author.

As pointed in Sections 3.1.3.3 and 3.1.3.5 (FQ2 and FQ4), undersampling techniques and

classical models obtain generally worse results than other methods. These studies have only been selected in the bigger sample of studies from the health domain. Therefore, better results should be expected in all domain areas by implementing solutions with combinations of oversampling or hybrid sampling with NN or ensemble models.

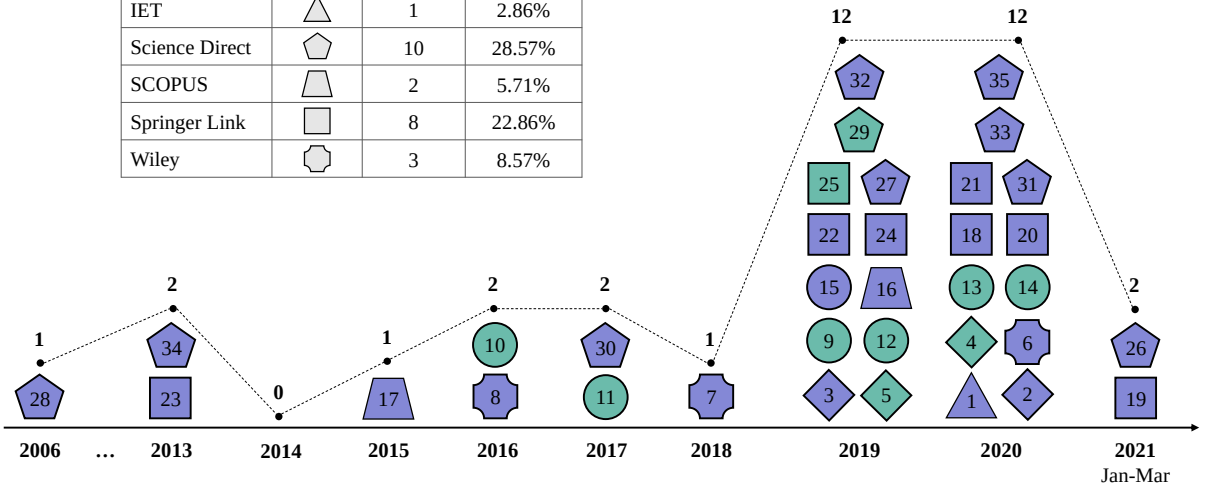
3.1.3.8 SQ1: How has the quantity of studies evolved?

Figure 16 shows the yearly publication of selected papers by the originated digital library and type of venue. The research was performed at the beginning of April 2021 without any date restraint.

Figure 16: Reviewed studies per year by digital library and type of venue

Digital library	Symbol	Quantity	Percentage
ACM		7	20.00%
IEEE Xplore		4	11.43%
IET		1	2.86%
Science Direct		10	28.57%
SCOPUS		2	5.71%
Springer Link		8	22.86%
Wiley		3	8.57%

Type of venue	Color	Quantity	Percentage
Journal		25	71.43%
Conference		10	28.57%



Source: Created by the author.

The results indicate a growing interest in data level preprocessing techniques for ML in imbalanced data applications, especially since 2019. Worth noting that EC7 and EC9 filtered out some algorithmic level techniques and solutions for multiple applications – creating a gap of representative works between 2007 and 2012. However, papers using these solutions followed a similar pattern of growing interest, presented in Figure 16.

3.1.3.9 SQ2: Where have the studies been published?

The representative works selection integrates 35 publications. This selection shows that 25 journal publications correspond to 71.4%, and 10 conference publications account for 28.6% of the studies reviewed in this work. Figure 16 indicates the type of venue of these studies by color.

The search process collected most selected papers through the digital libraries ACM, Science Direct, and Springer Link, where each accounts for at least 20% of the results. Additionally, only the journal "Artificial Intelligence in Medicine" has 2 works – one from 2006 (COHEN et al., 2006), and the other from 2020 (NNAMOKO; KORKONTZELOS, 2020).

3.1.4 Conclusion

This study applied a systematic mapping study to review current and effective data level preprocessing techniques and ML models in imbalanced data applications. After an eight-step filtering process, the selection of the representative works culminated in 35 papers. The results section presents two taxonomies and quantitative classifications of proposed, compared, and selected preprocessing techniques and ML models.

Overall, research studies mainly focus on applying standard or modified clustering-based sampling techniques for balancing data. Specifically, oversampling is the most common and also the best performing type of sampling, proportionally. Relatively, however, hybrid sampling techniques can potentially surpass oversampling if future studies implement them.

Classical ML models such as SVM, KNN, and LR still are the most frequent. Nevertheless, the most recent mapped studies show an increase in NN models – from simple ANNs, like MLP, to complex DL models. The results indicate that well configured NN models tend to achieve better results than classical models. Additionally, ensemble learning models also show promising results.

Ultimately, the results found in this systematic mapping study indicate that future works may explore the usage of simulation-based oversampling for balancing data in ML applications. Moreover, a solution with hybrid sampling mixed with NN or ensemble learning models can potentially achieve favorable results. Table 8 compiles the highlights from RQs' answers.

The lack of analyzable dataset characteristics is a limiting factor for this study. In future literature reviews, the authors suggest the addition of an EC if studies do not present the information of interest. An alternate study could be performed by reviewing papers with well-known prefixed datasets from different domain areas.

3.2 Real-Time Machine Learning architectures

This study's main objective is to analyze the topology of software architectures for supervised ML applications proposed in published papers. The study is divided into two reviews. The first one, on Section 3.2.1, performs an in-depth search for finding every paper proposing architectures for imbalanced data applications. The last one, on Section 3.2.2, scans papers proposing ingenious implementations in their architectures for general applications.

Both reviews apply a search strategy similar to the systematic mapping in Section 3.1: (1) specify search string; (2) select databases; and (3) collect results. However, this study samples

Table 8: Lessons learned by answering the Research Questions

RQ#	Lessons learned
FQ1	There are 5 central domain areas in imbalanced data applications: health (34.3%), finance (22.9%), engineering (14.3%), biology (8.6%), and software (8.6%). These areas have good references for new applications. New domains have the potential to be explored.
FQ2	The studies applied 55 different sampling techniques – oversampling (55.5%), undersampling (27.4%), and hybrid sampling (17.1%). Oversampling techniques achieved the best performance among the existing types, whereas hybrid sampling techniques performed better relatively (ratio of selected within tested studies).
FQ3	None of the studies used simulation as a means for optimizing synthetic data generation and accelerating training time in oversampling. This technology could optimize results and reduce computational costs in domains such as engineering and health.
FQ4	The studies applied 45 different ML models – classical (54%), ensemble (24.8%), and NN (21.2%). NN models achieved the best performance overall and relative to tested studies, with ensemble models as a close second.
FQ5	There are 3 recurrent development tools within the studies: Python, Matlab, and Weka. These tools have both sampling techniques and ML models already implemented as resources.
FQ6	Domain areas selected distinctive sampling techniques and ML models – especially in health. However, there is a clear preference for oversampling in engineering, biology, and software, while finance splits between oversampling and hybrid sampling. For ML, engineering selected only NN models, and finance selected mostly ensemble models. Other domains did not have a clear categorical preference.
SQ1	There is a growing research interest in the subject, especially since 2019.
SQ2	The 35 reviewed studies show a prevalence of journal publications, with 25 works (71.4%), while the remaining 10 are from conferences. The digital libraries ACM, Science Direct, and Springer Link account for at least 20% of the results individually.

Source: Created by the author.

results by selecting 2 of the 6 previous digital libraries: ACM and IEEE Xplore. This configuration concentrates a reasonable sample of software engineering works, since other digital libraries consist of broader domains. Finally, each review has an individual filtering process for selecting collected results – detailed in Sections 3.2.1 and 3.2.2.

3.2.1 Imbalanced data

This review applied a systematic mapping methodology for revising research publications proposing software architectures for supervised ML in imbalanced data applications. Hence, this work seeks to answer the following guiding GQ: “How have software architectures been solving the imbalance problem in ML?”.

3.2.1.1 Search strategy

Following the search strategy in Section 3.1, the first step determined major terms and their most relevant synonyms based on preliminary research. Afterwards, the search string in Table 9 merged major terms with their synonyms with Boolean operators. Finally, this string defined the query applied in both digital libraries – ACM and IEEE Xplore.

Table 9: Search string for imbalanced data applications

Major term	Search terms
Architecture	((architecture OR service OR framework OR system)
	AND
Real-time	(“real-time” OR “real time”)
	AND
Machine Learning	(“machine learning” OR “deep learning” OR “artificial intelligence”)
	AND
Imbalanced data	(“imbalanced data” OR “imbalanced dataset” OR “imbalanced data set” OR “unbalanced data” OR “unbalanced dataset” OR “unbalanced data set”))

Source: Created by the author.

3.2.1.2 Papers filtering

The collected papers went through a filtering process, removing ones that did not propose a software architecture for real-time ML with imbalanced data. The following EC supported the filtering process:

- *EC1*: the study is written before 2012;
- *EC2*: the study is not written in English;
- *EC3*: the study venue is neither conference nor journal;
- *EC4*: the study matches the keywords defined in the search string, but the context is different from the research purposes;
- *EC5*: the study is a literature review;
- *EC6*: the study is not accessible in full-text;
- *EC7*: the study is a short paper (4 pages or less);
- *EC8*: the solution focuses on a procedure for experiments;
- *EC9*: the study does not detail the proposed architecture or framework.

Papers filtering started at the initial search from each digital library, removing results complying with EC1, EC2, and EC3 – resulting in journal and conference publications, written in English over the last 10 years (up to the review date). Then, one filter by title extracted studies meeting EC4 and EC5.

The remaining papers went through a filter based on the three-pass method (Keshav S., 2007), excluding works complying with EC6, EC7, and EC8. Finally, a careful full-text read

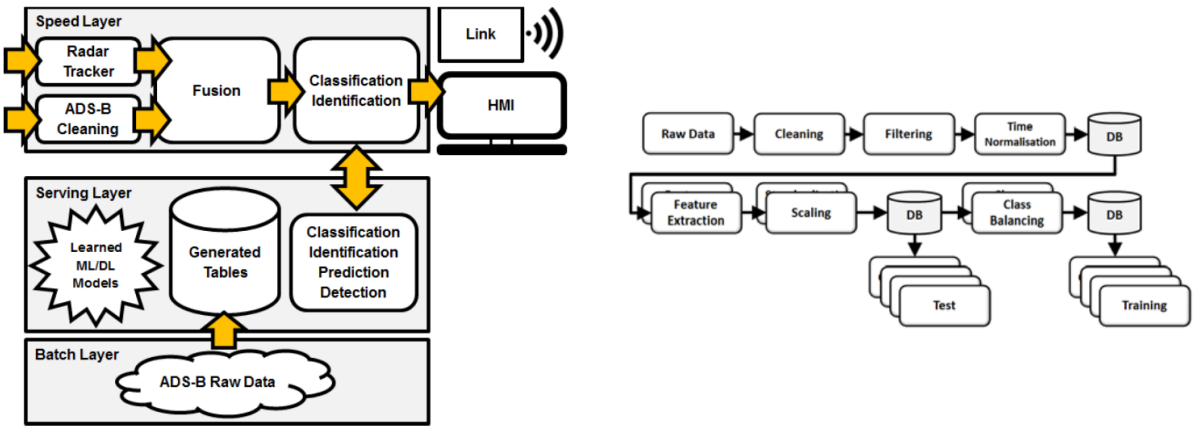
selected all papers detailing a software architecture for imbalanced data ML. The final step rejected experimental procedures for validating one scenario and low-quality papers – meeting EC8 or EC9, respectively.

3.2.1.3 Results

The combination of results from the 2 digital libraries integrated 472 works, reduced to 397 after applying EC1, EC2, and EC3. During the filtering process, a considerable amount of papers within the context of imbalanced data applications only propose an experiment procedure (EC8) and do not propose a software architecture (EC9). Incidentally, the selection of the representative works resulted in 5 papers.

Dästner et al. (2018) proposed the most complete work: a lambda architecture for classifying military aircraft in real-time radar systems using radar tracker and automatic surveillance data. The architecture consists of a batch layer for storing data, a serving layer for training and supplying classifiers, and a speed layer for applying the classifiers to identify aircraft through streaming data. The software has a preprocessing pipeline using oversampling to overcome the imbalance problem – elementary configured for an initially optimized and fixed IR after sampling. Figure 17 shows a block diagram of the architecture and the preprocessing pipeline.

Figure 17: Architecture for Machine Learning with imbalanced data: (a) block diagram; (b) preprocessing pipeline



(a) (b)

Source: Dästner et al. (2018).

Li et al. (2019) built an online defect detection system for large-scale photovoltaic plants. The system inspects photovoltaic modules with unmanned aerial vehicles and offloads captured images for expert acknowledgment of newly found defects by the ML model. The model consists of a CNN trained by previously labeled defects – acknowledged by experts. The system treats imbalanced data by generating synthetic images to train the model, as well as analyzing the model's PR curve to measure performance. Additionally, the system applies transfer learn-

ing to fine-tune the CNN model without having to retrain too frequently, reducing training time and resource consumption.

Correspondingly, Ruan et al. (2022) developed a cyber-physical system for fault prediction in industrial processes. The proposed model gathers real-time data from wireless sensors of a chemical plant and applies a novel DL network (CURNet) for multi-class classification. This DL network handles imbalanced time series data intrinsically.

Choi and Jeon (2021) proposed a real-time framework for detecting spam tweets. Similarly to Li et al. (2019), the framework uses experts to evaluate tweets and label spams to improve automated spam detection through supervised learning. A module from the framework is responsible for retraining the ML model when new diagnoses from experts reconstruct the training dataset. The solution handles imbalanced data with a cost-based ML model and evaluates results with the following metrics: precision, recall, and F-measure.

Finally, Barata et al. (2021) proposed a framework for designing ML solutions in applications under cold start, specifically with imbalanced data. The authors compared results from different proposed solutions with credit card fraud datasets. The study's main contribution comes from policies directing algorithms for fully unlabeled data (cold), few labeled data (warm-up), and sufficiently labeled data (hot). Whereas only unsupervised learning is able to handle cold conditions, warm-up and hot conditions may apply supervised learning algorithms. Barata et al. (2021) developed different probabilistic approaches for both warm-up and hot states. Specifically, the authors suggest a higher efficiency when applying only labeled data – as opposed to also using unlabeled data –, consuming less computing power to achieve better results. Additionally, the authors measured the compared models by running 35 simulations and evaluating performance over time.

In addition to the representative selection of works, Ahmed, Raman and Mathur (2020) presented challenges and suggestions for anomaly detection in real-time applications – focusing on industrial control systems. The authors make and discuss numerous assertions for supervised learning, noting the importance of updating the database with new labeled data in regular intervals and retraining the ML models – catering for environmental effects and process variations. Additionally, the authors highlight the difficulty of detecting distribution shifts and learning from imbalanced data, since they easily degrade the currently trained model.

3.2.1.4 Conclusion

This review applied a systematic mapping methodology for conducting a revision of research publications proposing software architectures for supervised ML in imbalanced data applications. After a filtering process, the selection of representative works resulted in 5 papers.

From these papers, only Dästner et al. (2018) detailed a software architecture with minor module management, segregating between 3 layers: speed, serving, and batch. However, the authors did not apply the proposed architecture in a real-case scenario, where the software

should automate data management and ML model retraining – testing performance over time. The study only collects results by changing dataset variables, such as oversampling rate and time series window size, as well as configurations with different ML models.

The remaining four studies show an overview of frameworks and systems for specific applications (LI et al., 2019; RUAN et al., 2022; CHOI; JEON, 2021; BARATA et al., 2021). From them, Li et al. (2019) apply oversampling indirectly by generating synthetic images. Other works do not use sampling techniques for preprocessing data (RUAN et al., 2022; CHOI; JEON, 2021; BARATA et al., 2021).

In conclusion, the gaps left in existing research present opportunities for further research. Challenges discussed by Ahmed, Raman and Mathur (2020) need more solutions and data. The complexity for active learning with imbalanced data, studied by Barata et al. (2021), expresses the need for analyses of performance over time. Finally, a fully automated architecture using current design and tools could improve upon the work of Dästner et al. (2018).

3.2.2 General applications

This review searched for efficient designs and tools in research publications proposing software architectures for supervised ML. Therefore, this revision seeks to answer the following guiding GQ: “How can ML software architectures be efficiently implemented?”.

3.2.2.1 Search strategy and Paper filtering

Following the search strategy in Section 3.1 and Section 3.2.1, the first step determined major terms and their most relevant synonyms based on preliminary research. Afterwards, the search string in Table 10 merged major terms with their synonyms with Boolean operators. Finally, this string defined the query applied in both digital libraries – ACM and IEEE Xplore.

Table 10: Search string for general applications

Major term	Search terms
Architecture	((architecture OR service OR framework OR system)
	AND
Real-time	(“real-time” OR “real time”)
	AND
Machine Learning	(“machine learning” OR “deep learning” OR “artificial intelligence”))

Source: Created by the author.

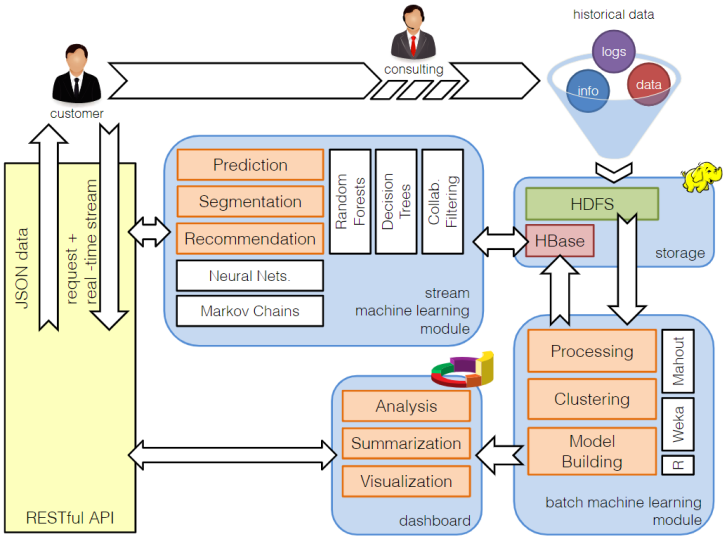
The search string in Table 10 expands the results obtained in Section 3.2.1 by removing terms related to imbalanced data. Consequently, the collected papers have a broader range of applications and solutions. Hence, to narrow the reviewed scope and improve research quality, the initial search from each digital library filtered only journal and conference publications written in English over the last 10 years (up to the review date).

Later, a brief read-through of titles selected works ingenious solutions for ML software architectures. Then, the remaining papers went through a filter based on the three-pass method (Keshav S., 2007). This step selected studies that detailed the proposed architectures, presenting visual representations of the topology design and processes, as well as charts validating proposed tools. Finally, a careful full-text read singled out a sample of papers with complete and efficient design and tools – avoiding duplicated ideas.

3.2.2.2 Results

Baldominos et al. (2015) proposed an architecture for real-time analysis in big data, arranged as an online service. The design resembles a lambda architecture – such as the one from Dästner et al. (2018) in Figure 17 –, dividing ML processes in two modules: batch and stream. The batch module is responsible for accessing and processing historical data, and training the ML model. The stream module is responsible for processing new data and returning predictions through the current ML model. The architecture conditions the batch module to execute periodically while the stream module runs in real-time.

Figure 18: Architecture for Machine Learning as an online service

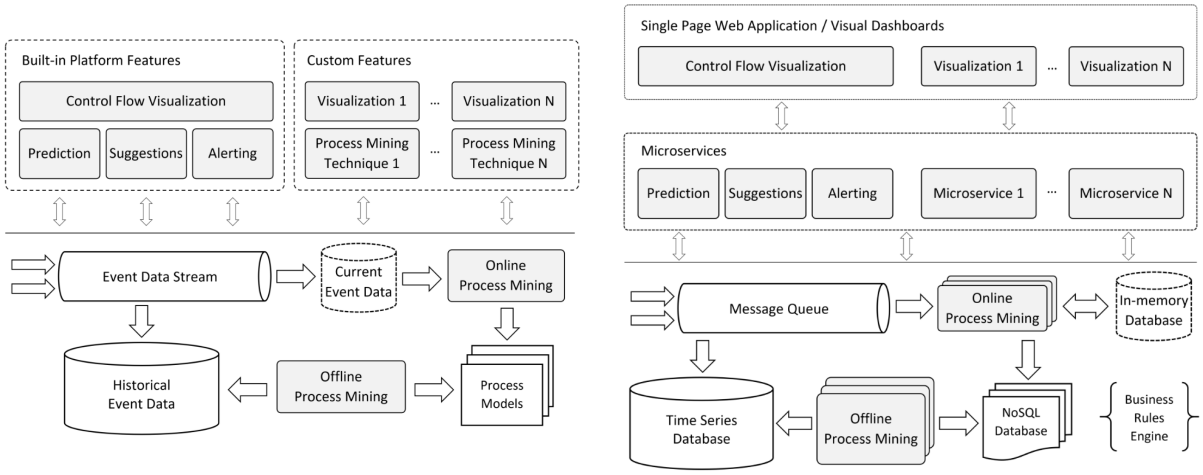


Source: Baldominos et al. (2015).

Bundling the ML modules, a REpresentational State Transfer (REST) Application Programming Interface (API) enables users to perform consultations for predictions and analyses. Predictions operate as immediate requests for the stream module. Analyses occur on-demand as an individual module (dashboard). Figure 18 shows the architecture's diagram. Although the paper does not develop new tools or algorithms, the authors conclude that the proposed topology simplifies data analysis for end users, extracting value from data while abstracting complexities. Additionally, preliminary evaluations showed encouraging results (BALDOMINOS et al., 2015).

Also based on a lambda topology, Batyuk, Voityshyn and Verhun (2018) designed a software architecture for monitoring processes in real-time. Aiming at applications that do not require big data, this architecture creates a local cache persisting recent data, minimizing latency and reducing complexity. Similarly to Baldominos et al. (2015), the authors propose serving a REST API for supplying a fast and flexible means of integrating the system to external services. Finally, the authors conclude that this design can efficiently extend features of third-party applications with processes on an operator control level.

Figure 19: Process monitoring applications: (a) architecture diagram; (b) components



Source: Batyuk, Voityshyn and Verhun (2018).

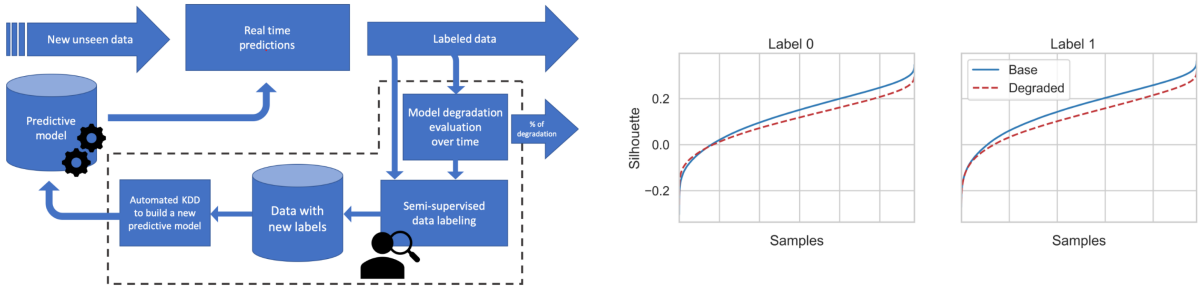
Cerquitelli et al. (2019) discussed the concept of automated triggers for updating ML models by continuously evaluating their performance with newly labeled data to detect distribution drifts. In the context of online models, predicting in real-time, model degradation self-evaluation enables unsupervised maintenance. The alternative is to retrain the model periodically, which is computationally intensive and less efficient (CERQUITELLI et al., 2019).

To fully automate the process, Cerquitelli et al. (2019) proposed using a quantitave metric for measuring model degradation. This metric triggers a retraining process after reaching a predefined threshold – within the risks of possible predicting errors. Figure 20 shows building blocks of the proposed framework and the results from an experiment conducted by the authors. In their conclusions, the authors direct future research to propose metrics and improve triggering mechanisms.

3.2.2.3 Conclusion

This review searched for efficient designs and tools in research publications proposing software architectures for supervised ML. A filtering process selected solutions with efficient implementations and clear directions for future research, resulting in 3 papers.

Figure 20: Measuring model degradation: (a) framework; (b) experimental results



(a) (b)

Source: Cerquitelli et al. (2019).

Considering solutions from filtered works, especially the selected papers, a lambda-based architecture emerges as an intrinsically efficient topology. Decoupling ML modules between batch and speed naturally organizes non-recurring model maintenance and time-constrained predictions. Additionally, wrapping ML processes with a REST API shows promising results. Existing papers classify the solution as a flexible way of integrating ML with external systems and returning predictions – while suppressing complexities from users (BALDOMINOS et al., 2015; BATYUK; VOITYSHYN; VERHUN, 2018).

Ultimately, ML processes – such as preprocessing, training, and optimization – need guiding rules to achieve a fully automated ML architecture. Cerquitelli et al. (2019) propose triggering these processes after the model's performance decreased to a maximum threshold of model degradation – measured with new data. Nonetheless, there is space for further research by employing metrics for specific applications, such as imbalanced data.

3.3 Final considerations

After the thorough examination of studies with sampling techniques and ML models, the results from the systematic mapping in Section 3.1 direct new works to efficient solutions and design considerations. These aspects are essential for creating a software architecture with a specific purpose, such as active ML with imbalanced data. In this sense, pairing the specialized points from imbalanced data solutions with the software topology references from Section 3.2, the following takeaways and research gaps can be concluded:

- Imbalanced data applications using ML have been increasingly studied, especially since 2019, presenting relevant references in broad domain areas – particularly with sampling techniques. However, the majority of software development solutions focus on experimental procedures and frameworks, opening a path for new software architectures;
- Lambda-based topologies are an efficient design for ML software architectures since the three layers have straightforward purposes in ML: batch for processing historical data, training and optimizing the model; speed (or stream) for processing new data and return-

ing predictions; and serving for enabling access to the ML capabilities;

- REST API is a flexible tool for serving ML and extending data analysis capabilities to external applications while suppressing complexities from users;
- The development of fully automated and computationally efficient ML systems depends on retraining criteria for maintaining a high-performance by measuring its degradation. In this context, there is space for further research on new metrics and triggers for dealing with specific problems – such as data imbalance;
- The lack of data from the minority classes is one of the main challenges for imbalanced data applications. Consequently, cold and warm-up conditions in developing applications delay reliable model deployment. To overcome this problem, new works can apply modern high-performance algorithms, such as GAN-based oversampling and ensemble learning, to evaluate improvements in model reliability and reduce ML systems' time to deployment;
- Creating a cache and using local storage is a clear path for applications that do not require big data, minimizing latency and reducing complexity. In this context, undersampling may enable this feature by reducing required storage;
- Oversampling and hybrid sampling for ML achieves encouraging results over different domain areas;
- NN and ensemble models achieve the best performance overall on imbalanced data applications;
- Python, Matlab, and Weka are favorable development tools in imbalanced data applications since they supply fully implemented resources for preprocessing data and training ML models.

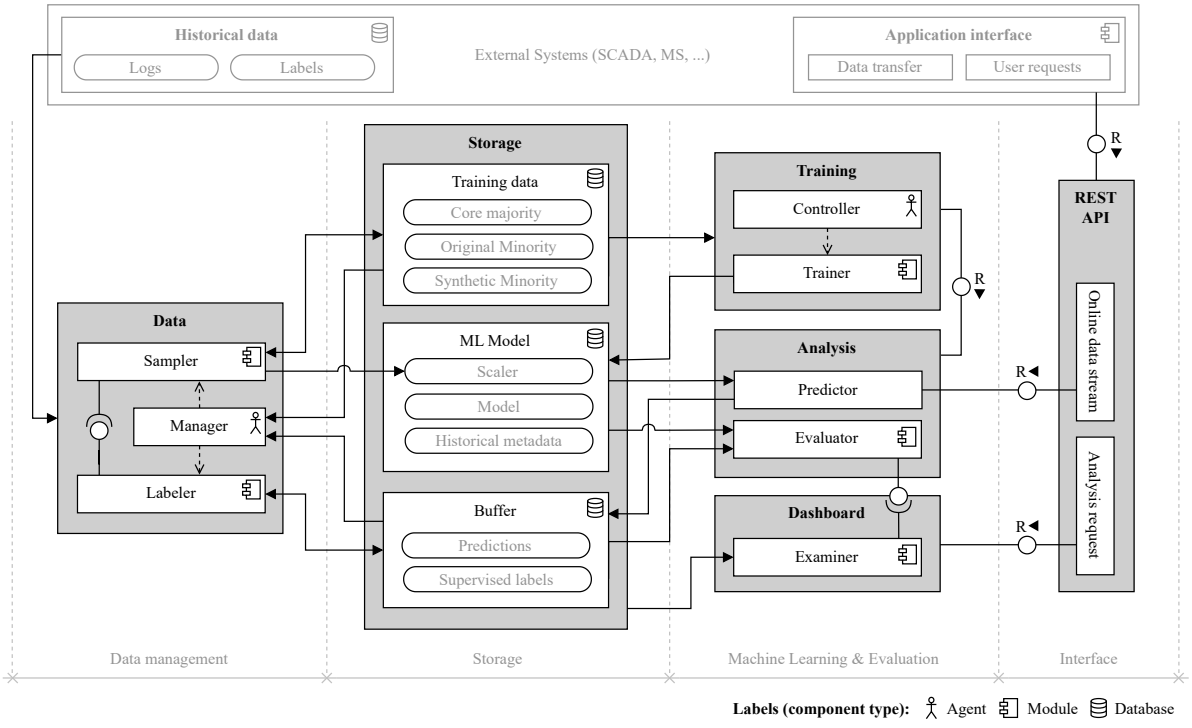
4 HEIMDALL

This chapter presents Heimdall, an architecture for online ML through imbalanced data. The chapter is organized in seven sections. The first one introduces the architecture of the proposed model and overviews its four layers. Sections 4.2 to 4.5 detail each layer, modules, and processes. Finally, Section 4.6 outlines potential applications, and Section 4.7 discusses final considerations on the topology.

4.1 Architecture overview

Heimdall is designed as a service for predictions and analyses requests, based on a lambda topology – organized with stream and batch processes. The service receives these requests through an interface with existing external systems, and accesses their databases for training and maintaining the ML model. Figure 21 presents the proposed architecture based on SAP's Technical Architecture Module (TAM) diagram (SAP, 2007). In summary, Heimdall is composed of four layers: data management; storage; ML and evaluation; and interface.

Figure 21: Heimdall's Technical Architecture Module diagram



Source: Created by the author.

The service interface encapsulates all ML capabilities and serves external systems, providing predictions and analyses on-demand through REST API requests. The service performs best by processing predictions in streams and examinations in batches – since evaluations have considerable changes only after enough new predictions. However, this rule is not mandatory,

and the service may stream both.

The architecture has two rule-based reactive agents independently monitoring data and deciding when to run processes: *Data Manager* and *Training Controller*. By definition, software agents are autonomous and aware of their environment, enabling decentralized decisions. Specifically, reactive agents respond to significant changes and events, hence causing immediate actions (PADGHAM; WINIKOFF, 2004). These characteristics can fully automate ML maintenance and allow asynchronous batch processing, potentially reducing CPU loads and improving computational efficiency.

Since synthetic data generation has a high-computational cost, the *Data Manager* is essential for deciding when and how to update training data in imbalanced data applications. This agent uses strict application-based criteria, enabling complex algorithms, such as GAN-based oversampling. Similarly, the *Training Controller* improves the ML model performance through metrics focused on imbalanced data.

Furthermore, the layers for data management, ML and evaluation, and interface contain modules with separate processing tasks, whereas storage symbolically keeps associated databases. Technically, both modules and reactive agents correspond to software classes. The next sections detail these components.

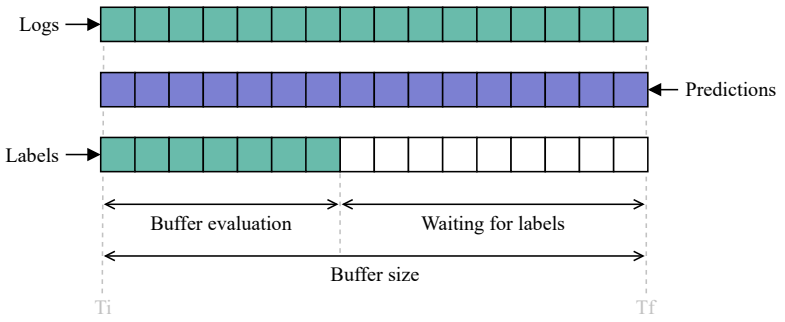
4.2 Data management

The data management layer is responsible for maintaining training data and supervised labels. To accomplish that, the *Data Manager* monitors historical data from databases and decides if and how both *Labeler* and *Sampler* act.

External systems usually segregate data through two processes: logging and labeling. While systems log measures and information automatically – periodically or through event-driven approaches –, labels are commonly classified or validated by human experts. This supervised learning characteristic creates a delay for performance evaluation since predictions synchronize with historical data logs (real-time), but data is only labeled afterward. Figure 22 demonstrates this characteristic, in which the buffer contains unlabeled predictions that are not taken into account for performance evaluation. Consequently, the buffer size is bigger than the evaluation sample.

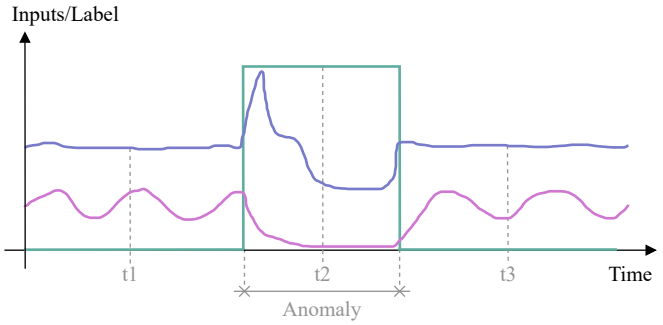
The relationship between logging and labeling databases varies according to the application. Applications sharing unique IDs (keys), such as shopping orders, need only direct connections for labeling. However, applications sharing only time series data, such as fault analysis, require anomaly detection algorithms for labeling data through input variables (logs) and reported occurrence time (labels) by experts. Hence, the algorithm has to find unusual disturbances within input data around the reported time and label the anomaly. Figure 23 illustrates this phenomenon. In Heimdall, the *Labeler* is responsible for performing this task – labeling both buffer and training data, inherited by the *Sampler*.

Figure 22: Machine Learning model evaluation delay caused by supervised labels



Source: Created by the author.

Figure 23: Time series anomaly labeling: (a) chart; (b) data



(a)

Time	t1	...	t2	...	t3
Input 1	0.58	...	0.35	...	0.59
Input 2	0.32	...	0.02	...	0.27
...	...	...	...	...	...
Label	0	...	1	...	0

Reported anomaly time: around t2

(b)

Source: Created by the author.

The *Sampler* has two main tasks: generating training data and updating the minority sample. The first one transforms historical data into training data through necessary preprocessing techniques. For instance, in the case of time series data, the module should label and slice data through predetermined periods, before and after anomalies.

Data preparation is an essential step for ML – particularly for imbalanced data (MURESAN et al., 2015). The techniques composing this step depend on the application. For instance, image preprocessing requires gray scale conversion and data augmentation, while text preprocessing requires tokenization, stemming and feature extraction. Generally, the fundamental techniques can be conventionalized as:

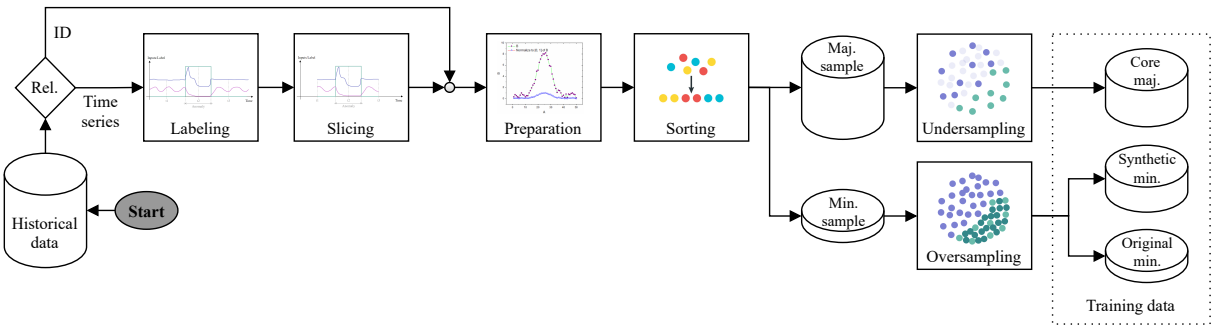
- *Null data handling*: solving raw data inconsistencies due to missing values throughout instances' features. Example: removal of instances or imputation of values through customized functions, such as mean;
- *Dimensionality reduction*: reducing the feature space (number of features) while retaining as much information as possible. Example: Principal Component Analysis (PCA);
- *Scaling*: scaling numerical attributes to fit within a specified range – usually $[0, 1]$ or $[-1, 1]$. Example: min-max scaler;
- *Encoding*: converting textual classes (labels) into scalar values. Example: label encoder.

In addition to these techniques, the *Sampler* sorts preprocessed data between majority and minority samples before applying sampling techniques.

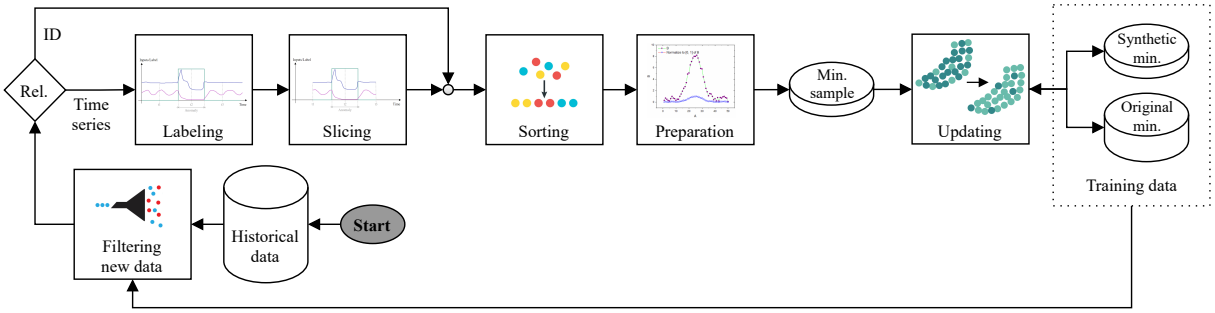
Following the takeaways concluded in Section 3.3, Heimdall should preferably apply hybrid sampling or oversampling with complex and CPU-demanding algorithms – such as GAN-based oversampling. This process results in a balanced relational database divided into: core majority sample (undersampled or raw); synthetic minority (oversampled); and original minority. Figure 24a illustrates this process with hybrid sampling.

The update process for new instances initiates filtering new data from existing training data. After that, it sorts only minority samples for preparation – configured identically to the current training data. Lastly, the task removes the closest synthetic minority data and adds new original minority data. Figure 24b shows this process.

Figure 24: Sampler processes flowcharts: (a) training data generation; (b) minority update



(a)



(b)

Source: Created by the author.

Both training data generation and minority sample update processes depend on triggers from the *Data Manager*. This agent's independence grants training consistency and computational efficiency – since high-performance oversampling generates high-quality training samples, but it is not overdone when not necessary. Furthermore, the delegation for labeling the predictions buffer (supervised labels) also maintains the ML model's evaluation efficiency.

Moreover, each application may have different rules for managing data. Table 11 generalizes essential rules to trigger processes from both *Sampler* and *Labeler*. Specifically, the *Sampler*'s rules can be applied in conjunction. For instance, although periodicity enables generation, the

Data Manager has to wait for an offline gap. This capability is advised for sampling generation since this process has high CPU usage. However, the *Data Manager* should prioritize high-frequency labeling – and update, if necessary for the application.

Table 11: General rules for Data Manager triggers

Module	Sub-process	Rule	Description
Sampler	Generation	Delay	Pre-defined number or percentage of new labeled anomalies since the last generation
		Periodicity	Pre-defined time difference between last generation and recent historical data
		Offline gap	External system offline for enough time (maintenance, weekend, ...) to generate training data – calculated through last generation time and data load
	Update	Delay	Pre-defined number or percentage of new labeled anomalies since the last update
		Periodicity	Pre-defined time difference between last update and recent labeled anomalies
		Offline gap	External system offline in order to update minority samples through new labeled anomalies
Labeler	Label	Delay	Pre-defined number or percentage of new labeled anomalies since last supervised label

Source: Created by the author.

Finally, the data management modules save data locally – minimizing latency and reducing complexity. These databases, proposed in Section 4.3, are accessed by components of the ML and evaluation layer, as detailed in Section 4.4.

4.3 Storage

The storage layer represents data saved within processes for maintaining the service. This data, displayed at each process of Heimdall's pipeline in Figure 25, is grouped in: training data, ML model, and buffer.

The features composing each database vary since every application has different inputs and parameters. However, in order to accomplish Heimdall's purpose and allow the processes detailed in Sections 4.2 and 4.4, these databases have minimally essential features. Table 12 describes these features by group, database and type – based on Python data types.

4.4 Machine Learning and evaluation

The ML and evaluation layer has four main tasks: training the ML model through data maintained by the data management layer; making predictions for new data; evaluating the model's performance; and creating analyses to validate predictions and training data. These

Table 12: Heimdall databases' essential features

Group	Database	Feature	Type	Description
	Training data	DT	datetime	Date and time of occurence
		ID	int, str	Identifier (key)
		Inputs*	float	Preprocessed input variables
		Label	int, str	Supervised label (or class)
		Synthetic	bool	If instance is artificially generated
		Train	bool	If instance is used in training
ML Model	Historical metadata	StartDT	datetime	Start date and time of training
		TrainTime	datetime	Training time from StartDT
		SampleSize	int	Number of instances
		LabelRatios*	dict	Proportion of each label
		Optimize	bool	If optimized, else retrained
		Model	str	Trained algorithm
		Hyperpar*	int, float	Algorithm's hyperparameters
		ProbThr	float	Optimized probability threshold
		CVPar*	int, float	Cross-validation parameters
		Metrics*	float	Trained model's CV results
		BPI	float	BPI (Equation 4.1)
	Scaler		object	Trained scaler object
	Model		object	Trained ML Model object
	Buffer	DT	datetime	Date and time of rep. occurence
		ID	int, str	Identifier (key)
		Prediction	int, str	Predicted label
		Label	int, str	Supervised label

* indicates multiple features, represented as one due to sharing the same description

Source: Created by the author.

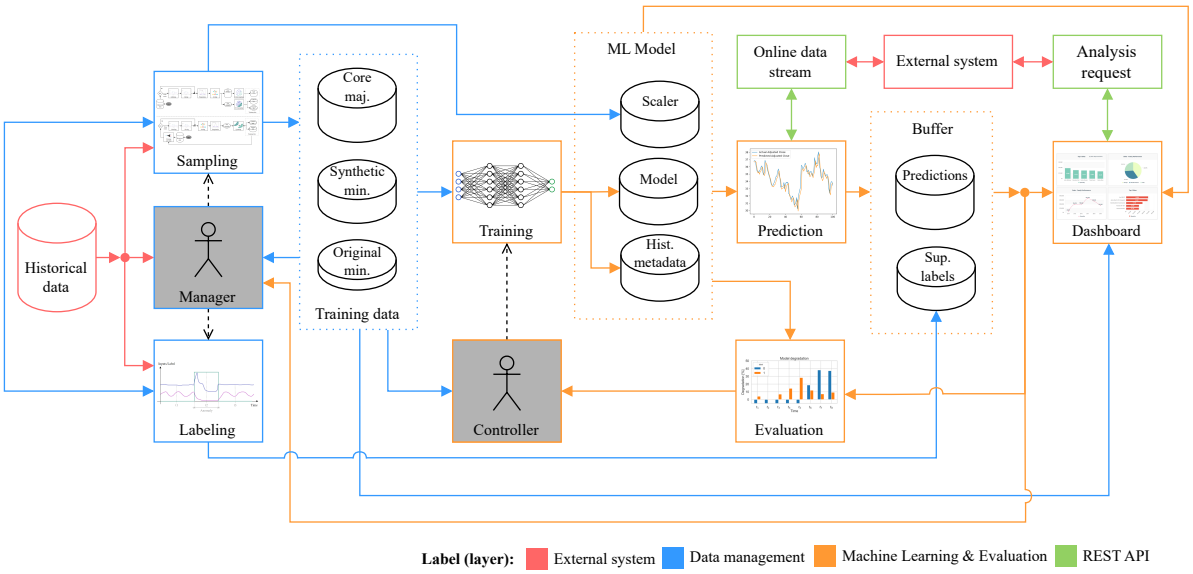
tasks are executed by the four corresponding modules from Figure 21 – *Trainer*, *Predictor*, *Evaluator*, and *Examiner*.

Similarly to the data management layer, ML and evaluation has an agent, named *Training Controller*. This agent monitors training data and the model's performance to decide if and how the *Trainer* acts. In contrast to the basic pipeline from Figure 9 and to the real-time ML architecture from Figure 18, the inclusion of these reactive agents optimizes processing efficiency through automated triggers – as suggested by Cerquitelli et al. (2019). Figure 25 illustrates Heimdall's pipeline for ML and evaluation through training data and supervised labels maintained by the data management layer. Additionally, this figure shows the interface with an external system – following the TAM diagram in Figure 21.

The learning process, performed by the *Trainer* and, uses local training data. Following the conclusions from the systematic mapping for ML in imbalanced data applications (Section 3.1), the architecture should preferably apply NN or ensemble models to achieve good results. Additionally, the *Trainer* can use different approaches and technologies:

- *Standardization*: selected model through Exploratory Data Analysis (EDA) and iterations

Figure 25: Heimdall's pipeline and data flow



Source: Created by the author.

between different models;

- *Application-specific*: model specifically developed for the application in previous or related works – such as the QSAR-based FP2VEC for learning molecular properties (LEE; KIM, 2020);
- *Optimization*: dynamic adjustment of the ML model hyperparameters to improve the objective function composed by the performance metrics of interest – such as grid search (Figure 8);
- *Auto ML*: automated iterations between different models and optimization of their hyperparameters.

Auto ML has growing research interest and potentially good results – even outperforming human data scientists within training time or performance in some cases (HANUSSEK; BLOHM; KINTZ, 2020). However, once the best model is selected, this iterative process is more time-consuming and may create unpredictable performances due to model variability. Nonetheless, developers with less modeling experience or data knowledge could benefit from prototyping with this technology before deploying it to Heimdall.

In contrast, standardized and application-specific models coupled with optimization algorithms favor the proposed architecture by producing stable performance metrics and training times. The *Trainer* saves these measurements for every trained model into the historical metadata database, allowing future analyses. Firstly, stable performance metrics enable the validation of the model's continual improvement at every training. Secondly, stable training times support more accurate estimations for the *Training Controller*.

When optimizing the model, the *Trainer* iterates through hyperparameters and cross-validates the model to calculate generalized performance metrics. Since imbalanced data has dispropor-

tionate samples for each class, the *Trainer* uses the stratified k-fold cross-validation – preserving the IR. Additionally, the cross-validation uses the same predefined parameters from previous models.

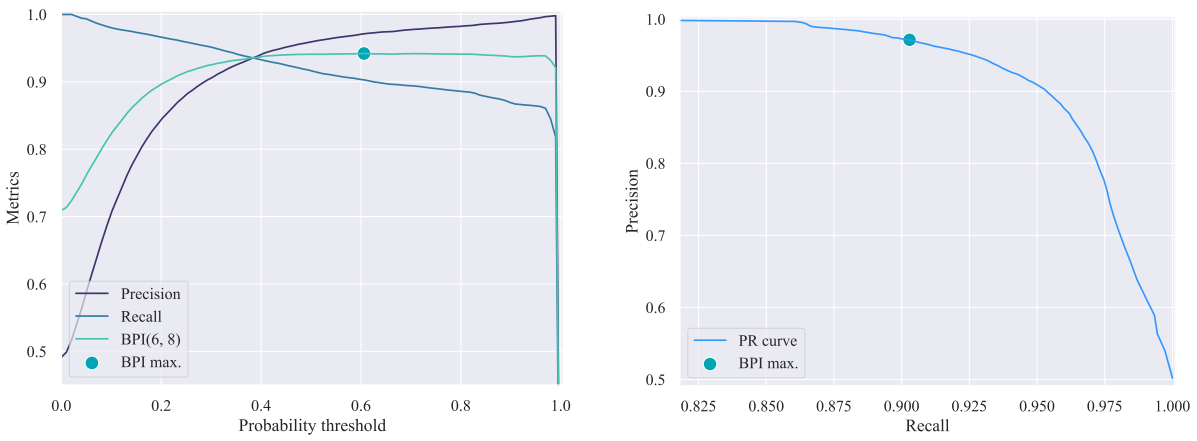
Subsequently, the training process searches for the best probability threshold – described in Section 2.2.1 – to maximize recall and precision within possible, according to the application priorities. Instead of using the F-Measure or standard AUC of probability curves, this study proposes a new metric for imbalanced data, called Balanced Performance Indicator (BPI), for the objective function. This metric calculates the weighted arithmetic mean between recall and precision. Thus,

$$BPI(\alpha, \beta) = \frac{\alpha \cdot rec + \beta \cdot pre}{\alpha + \beta}, \in [0, 1] \tag{4.1}$$

where: rec is the recall; pre is the precision; α is the recall weight; and β is the precision weight.

The BPI allows flexibility for solving the applications' priorities by analyzing PR trade-offs. For instance, if the application detects faults in power systems, there is a high cost for FNs – so it is preferable to weigh more on recall than on precision. Alternatively, if the application detects fraudulent transactions, it is best to avoid overloading and delaying clients by reducing FPs – weighing more on precision. If a fraud is not detected, the client reports it, and the model improves in future training data updates. Figure 26 illustrates the flexibility of BPI, prioritizing precision over recall and achieving better results than the standard probability of 0.5.

Figure 26: Balanced Performance Indicator's flexibility: (a) probability threshold curve; (b) standard Precision-Recall curve



(a) (b)

Source: Created by the author.

The *Training Controller* delegates the training process by monitoring three sources: training data, historical metadata, and buffer evaluation. In conjunction with analyses of these sources, this agent's independence guarantees training efficiency and reduces model degradation. Similarly to the *Data Manager*, each application may have different rules for training a model.

Table 13 generalizes essential rules to trigger the *Trainer*. Ideally, if the application has processing restraints, delay, degradation, and periodicity should wait for an offline gap.

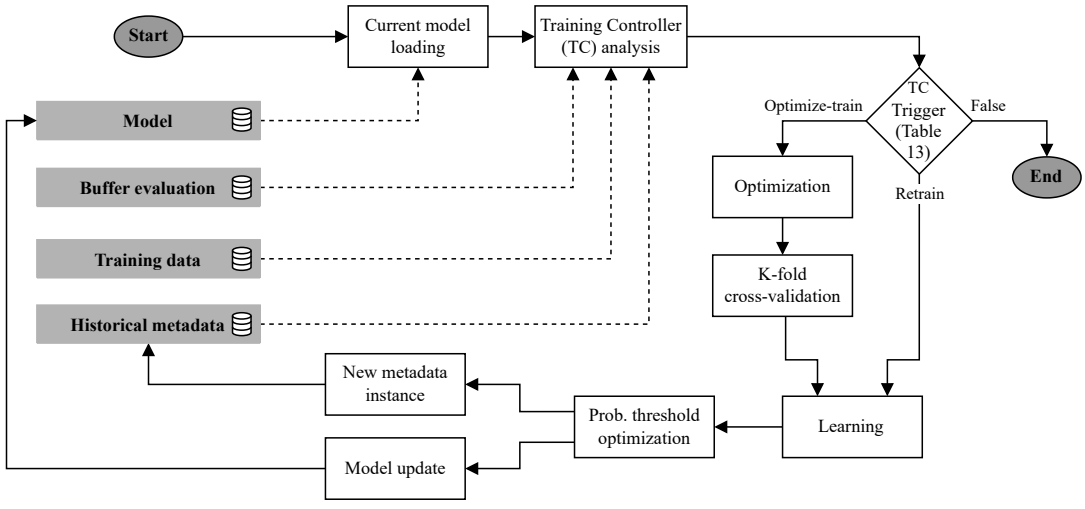
Table 13: General rules for Training Controller triggers

Module	Process	Rule	Description
Trainer	Training	Delay	Pre-defined number or percentage of new minority instances since last training
		Degradation	Pre-defined negative variation in the model's evaluation
		Periodicity	Pre-defined time difference between recent training data and the historical metadata time period
		Offline gap	External system offline for enough time (maintenance, weekend, ...) in order to train a new model

Source: Created by the author.

In general, the complete training process – with optimization – should only be commanded by the *Training Controller* when the model's performance degraded below a significant threshold. Delay and periodicity avoid model degradation by fitting (retraining) the current model with new training data generated by the *Data Manager*. Otherwise, no trigger should be generated since the current model retains acceptable performance. Figure 27 illustrates this process.

Figure 27: Training control



Source: Created by the author.

Training times generally vary according to the selected ML algorithm, number of features, and sample size. For instance, the time complexity for SVM regression is $\mathcal{O}(n_{ft} \cdot {n_{sp}}^{p})$, where $2 \leq p \leq 3$ (SCIKIT-LEARN, 2021). However, since the *Trainer* saves the trained models' historical metadata, detailed in Section 4.3, the training time can be estimated through regression from previous models – enabling the projection of adequate offline gap periods.

The *Predictor* loads the currently trained model in Random-Access Memory (RAM) for predicting the output of new streamlined data – returning REST API requests (Section 4.5).

Moreover, this module verifies the current model by accessing historical metadata and reloads if the *Trainer* updates the model. By default, this verification happens at every new prediction, but high-frequency applications can benefit from low-frequency accesses.

After the prediction for a new instance is processed, the *Predictor* returns it for the external system through the REST API. Additionally, the module appends all predictions to the buffer – subsequently labeled by the *Labeler*, maintained by the data management layer. As illustrated in Figure 22, only labeled predictions compose the evaluated buffer.

Similarly to the *Trainer*'s continual improvement method, the *Evaluator* applies stratified k-fold cross-validation or stratified train-test split to obtain performance metrics and calculate the buffer's BPI through Equation 4.1. This result feeds the *Training Controller*. Finally, the *Examiner* uses training data, historical metadata, and buffer to generate summaries and charts on the stored data via REST API requests.

4.5 Interface

The interface layer has the task of serving requests for predictions and analyses. This layer is composed of a REST API, enabling the external system (client) to access Heimdall's resources (server) on-demand. Consequently, these systems can extend their functionalities with artificial intelligence resources without any internal ML developments.

Specifically, this topology enables external systems to automatically stream newly logged data to Heimdall, benefiting from online prediction responses for real-time applications. Moreover, these systems can request the *Examiner*'s analyses in batches, allowing periodical validations – such as model degradation, current performance, and buffer delay. Finally, the REST API approach facilitates a flexible, fast, reliable, and widely-accepted interface (BALDOMINOS et al., 2015; BATYUK; VOITYSHYN; VERHUN, 2018).

4.6 Applications

Heimdall is an architecture with the potential to be applied in any application from the five domains in Table 7 – mapped in Section 3.1. Namely, these applications could add a client to the available management system for accessing Heimdall's resources. Some of the potential applications are:

- *Health*: disease detection and prognostic in hospital and exam management systems;
- *Finance*: fraud and risk evaluation in corporate and banking systems;
- *Engineering*: fault detection in machines, systems, and industrial plants;
- *Biology*: classification in research facilities and laboratories;
- *Software*: data and source code analyses in software companies.

Furthermore, other applications implemented in systems with enough development flexibility – to access Heimdall's REST API – and processing power could benefit from this service.

Ultimately, Heimdall encompasses rulesets and best practices described in previous sections. In this sense, new imbalanced data applications can ensure good performance and efficiency by employing the best sampling techniques and ML models from related domains – such as the ones analyzed in Section 3.1.3.7.

4.7 Final considerations

This chapter detailed Heimdall, an architecture for online ML through imbalanced data. The architecture stands out from related works by proposing new strategies centered on improving performance in imbalanced data applications – such as a new performance metric and minority sample data management. In addition, Heimdall applied research conclusions and efficient features from reviews of related works (Section 3.3). Table 14 compares key characteristics from related works (Sections 3.2.1 and 3.2.2) in relation to the proposed architecture.

Table 14: Comparison between Heimdall's and related works' solutions

Solution characteristics	Dästner et al. (2018)	Li et al. (2019)	Ruan et al. (2022)	Choi and Jeon (2021)	Barata et al. (2021)	Baldominos et al. (2015)	Batyuk, Voityshyn and Verhun (2018)	Cerquitelli et al. (2019)	**Heimdall**
Focused on imbalanced data	X	X	X	X	X				**X**
Designed as an architecture	X					X	X	X	**X**
Lambda-based architecture for ML	X					X	X		**X**
Local storage		X	X	X	X		X		**X**
Handles time series data	X	X	X				X		**X**
Preprocesses with sampling techniques	X	X							**X**
High-performance oversampling									**X**
Training data monitoring									**X**
Neural Networks or ensemble models		X	X						**X**
Analyzes probability curve		X							**X**
Online service with REST API						X	X		**X**
Model degradation monitoring					X			X	**X**
Rule-based active learning					X			X	**X**
Reactive agents for automation									**X**
Balanced Performance Indicator									**X**

Source: Created by the author.

To sum up, Heimdall stands out through the following key aspects:

- Architecture designed as an online service for imbalanced data applications, extending artificial intelligence functionalities to a wide range of existing systems;
- Local storage for fast data access and implementation simplicity;
- Processing capabilities for both ID and time series-only relational databases;
- Full automation through two rule-based reactive agents, responsible for independently monitoring data changes and model degradation, as well as triggering processing pipelines;
- High-performance in imbalanced data applications through complex sampling algorithms, potentially enabling earlier deployment – in warm-up conditions;
- Performance enhancement through probability threshold optimization and a novel metric (BPI) as the objective function.

5 EXPERIMENTAL EVALUATION

This chapter presents the experimental evaluation of a prototype based on Heimdall. The evaluation focuses on incrementally enabling proposed functionalities through five scenarios, validating their improvements to systemic efficiency and performance in one severely imbalanced data application. Consequently, analyzed results show the efficiency of the proposed architecture for active learning in imbalanced data.

Section 5.1 details the application and selected dataset. Section 5.2 presents the testing methodology, describing the developed prototype's functionalities, components extracted from Heimdall, and test configurations to obtain results. Subsequently, Section 5.3 analyzes test results, comparing them to baseline architectures and related works. Finally, Section 5.4 discusses final considerations on the experimental evaluation.

5.1 Application and dataset

As indicated in Section 3.1.3.2, finance is one of the most studied domains with imbalanced data applications. One of these applications is credit card fraud – a highly imbalanced binary classification problem where most transactions are legitimate (not fraudulent).

With the advancement of digitalization and utilization of internet technology in banking institutions, credit card transactions increase significantly from a wide range of payment channels – such as physical stores and e-commerce. Consequently, credit card fraud poses a grave financial issue (ITOO; MEENAKSHI; SINGH, 2020). The literature usually refers to solutions for this problem as Credit Card Fraud Detection (CCFD) or CCFD Systems (CCFDS).

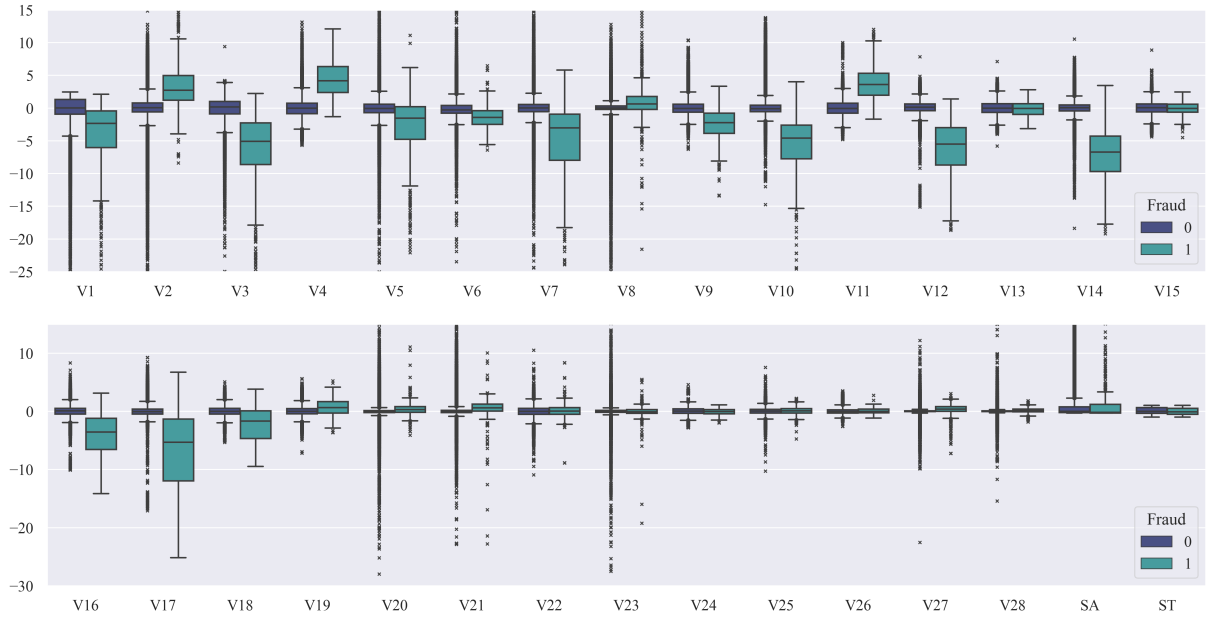
A systematic review of CCFD by Priscilla and Prabha (2019) analyzed evaluation metrics, ML models, tools, and datasets. The most used public dataset has been provided by a research collaboration within the Université Libre de Bruxelles (ULB), accounting for 37.5% of mapped works. The ULB-CCFD dataset compiles transactions made by European credit cardholders in September 2013 (ULB, 2018).

Highly imbalanced, the ULB-CCFD dataset contains 492 frauds in 284,807 transactions. Therefore, frauds account for 0.173% of all transactions, resulting in an IR of approximately 1:579. Whereas payment-processing companies like MasterCard© track multiple features – such as transaction size, location, time, device, and purchase data (ALTEXSOFT, 2021) –, ULB renamed the dataset's features due to confidentiality issues. In addition, the university preprocessed the dataset with a PCA transformation, reducing feature space and showing only numerical input (V1, V2, ..., V28). As a result, the dataset contains only the 28 transformed variables, the *time* elapsed between each and first transaction, and the *amount* paid (ULB, 2018). Figure 28 illustrates the numerical data distribution for PCA features, and scaled *time* and *amount*.

Thus, the implementation of the present study selected the ULB-CCFD dataset for the following reasons: real-life data; a reasonable amount of instances; extremely low IR; and a sub-

stantial quantity of related works with standardized preprocessing – due to the built-in PCA.

Figure 28: Box plot of PCA features, Scaled Amount (SA) and Scaled Time (ST) for each class of the ULB-CCFD dataset



Source: Created by the author.

5.2 Testing methodology

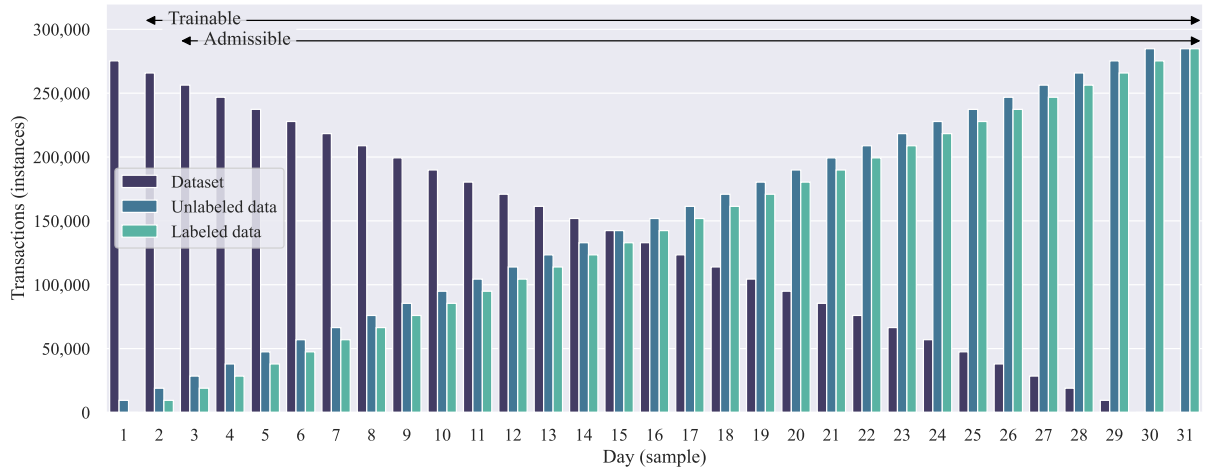
Following the takeaways from Section 3.3, a CCFDS prototype was developed in Python, applying resources from various third-party packages distributed by the *Anaconda platform*[1], such as *pandas*, *sklearn* and *tensorflow*. Each processing layer from Heimdall contains the corresponding modules and reactive agents as classes with necessary methods and attributes. Conversely, the storage layer consists of Python objects (ML model and scaler) and relational databases – such as training data, buffer, historical metadata and evaluations.

In addition to the CCFDS, a test environment developed in a Jupyter Notebook acts as an external system – connecting to Heimdall's interface and streaming new data to detect fraudulent transactions and requesting analyses focused on performance evaluation for this study's purposes.

The test algorithm extracts and streams 30 samples from the ULB-CCFD dataset to validate performance over time and deployment in warm-up conditions (BARATA et al., 2021). This number simulates the 30 days from the recorded data (September 2013), corresponding to an average of approximately 9493 instances and 16 frauds within daily transactions. Furthermore, the test creates a 1-day delay between the streaming and labeling to account for expert labeling or deficiencies in the process. Figure 29 illustrates data flow throughout the test.

[1] https://www.anaconda.com/

Figure 29: Daily samples throughout the evaluation of performance over time



Source: Created by the author.

Specifically, the test evaluates the effect of Heimdall's essential functionalities on CCFDS's performance by incrementally enabling them and creating different scenarios – shown in Table 15. These scenarios collect results for classification metrics and processing time throughout the daily data flow (Figure 29), evaluating performance through buffer data. Since preliminary results of scenarios 3 to 5 yielded mostly perfect performance scores through only cross-validation, the prototype's *Trainer* segregates currently labeled data into stratified (similar IR) train-test sets of the standard 70-30% size, respectively. The train set goes through preprocessing for learning, while the test set is saved as the initial buffer – which grows in size after each daily stream of transactions through the REST API. Figure 30 summarizes this procedure.

Table 15: Evaluation Scenarios: S1-S5

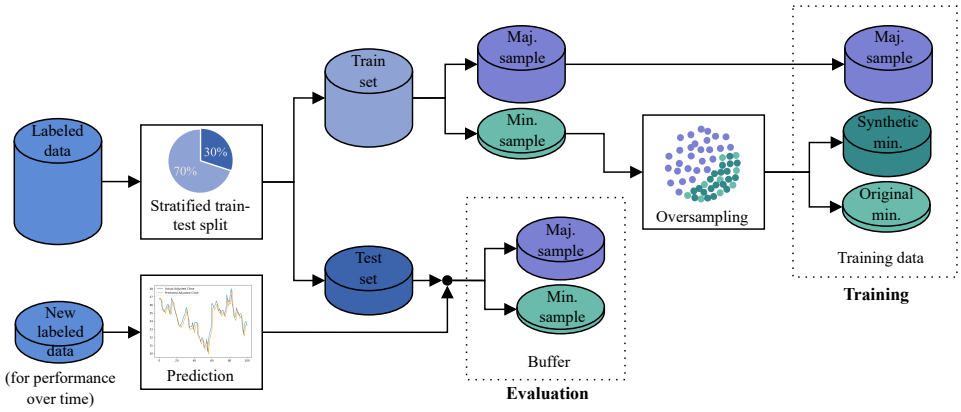
Scenario	1	2	3	4	5
Standard ML	X	X			
Hyperparameters optimization through BPI	X	X	X	X	X
Standard oversampling	X	X	X	X	
Probability threshold optimization through BPI		X	X	X	X
Ensemble ML			X	X	X
Data Manager & Training Controller				X	X
High-performance oversampling					X

Source: Created by the author.

In addition to the performance over time, a second script tests the overall performance for the configuration from each scenario applying to the entire ULB-CCFD dataset – using the same stratified train-test split characteristics from Figure 30. These results are compared between one another and against best-performing models in related works – mapped by Priscilla and Prabha (2019).

Complying with the preferences for sampling and ML in the finance domain (Section 3.1.3.7)

Figure 30: Testing data procedure for performance over time and overall



Source: Created by the author.

and in CCFDS (PRISCILLA; PRABHA, 2019), the tests apply the following techniques and models: SMOTE for standard oversampling; CGAN for high-performance oversampling; LR for standard ML; RF for high-performance ML (ensemble). A preliminary EDA defined the ML model's hyperparameters iterated in a grid search optimization.

Moreover, the CCFDS prototype respects the following premises to evaluate improvements in performance and efficiency:

- *Admissible results* (S1-S5): labeled data must contain at least 50 anomalies (frauds) to allow admissible results – achieved on day 3 due to the 1-day delay, as illustrated in Figure 29. Since training data is split into train-test (70-30%), this premise requires at least 15 frauds in the buffer to evaluate performance properly;
- *Baseline architectures* (S1-S3): periodical execution of basic ML pipelines (data preprocessing and ML model training), with hyperparameter optimization and standard probability threshold (fixed at 0.5), every 3 days (update);
- *Probability threshold optimization* (S2-S5): wrongful fraud predictions must not delay cardholders. Therefore, the CCFDS must prioritize the mitigation of FP, weighing more on precision without sacrificing recall – hence, varying the probability threshold for a BPI with $\beta > \alpha$ for Equation 4.1. Definition: $\beta = 8$ and $\alpha = 6$;
- *Data management* (S4-S5): the *Sampler* must generate new training data only if labeled data accumulated 7% more frauds than current training data (delay) – following Table 11;
- *Training control* (S4-S5): the *Trainer* must only retrain the model if the *Evaluator* measured a $1.5 - 3\%$ drop in BPI (degradation) – following Table 13 and Figure 27. Additionally, the *Trainer* must train and optimize the model if the *Evaluator* measured a drop higher than 3% in BPI (degradation) or if 7 days have passed since the last optimization (periodicity).

As a result, collected data from the testing environments, presented in Section 5.3, enable the answer to questions such as: does high-performance oversampling and ensemble learning enable deployment in warm-up conditions? Are reactive agents efficient for ML automation?

Compared to periodically trained ML models, what are the advantages of a fully automated architecture? Is Heimdall a perceptive solution to implement ML applications with imbalanced data for online prediction?

5.3 Results and discussion

Following the principles defined by He and Ma (2013), one of the main imbalanced data issues is absolute rarity. Therefore, historical analysis of performance over time enables conclusions on time to deployment – specifically at a stage of warm-up conditions, when there are few labeled data (BARATA et al., 2021). Hence, the test environment for performance over time, described in Section 5.2, registered analyses for each streamed sample.

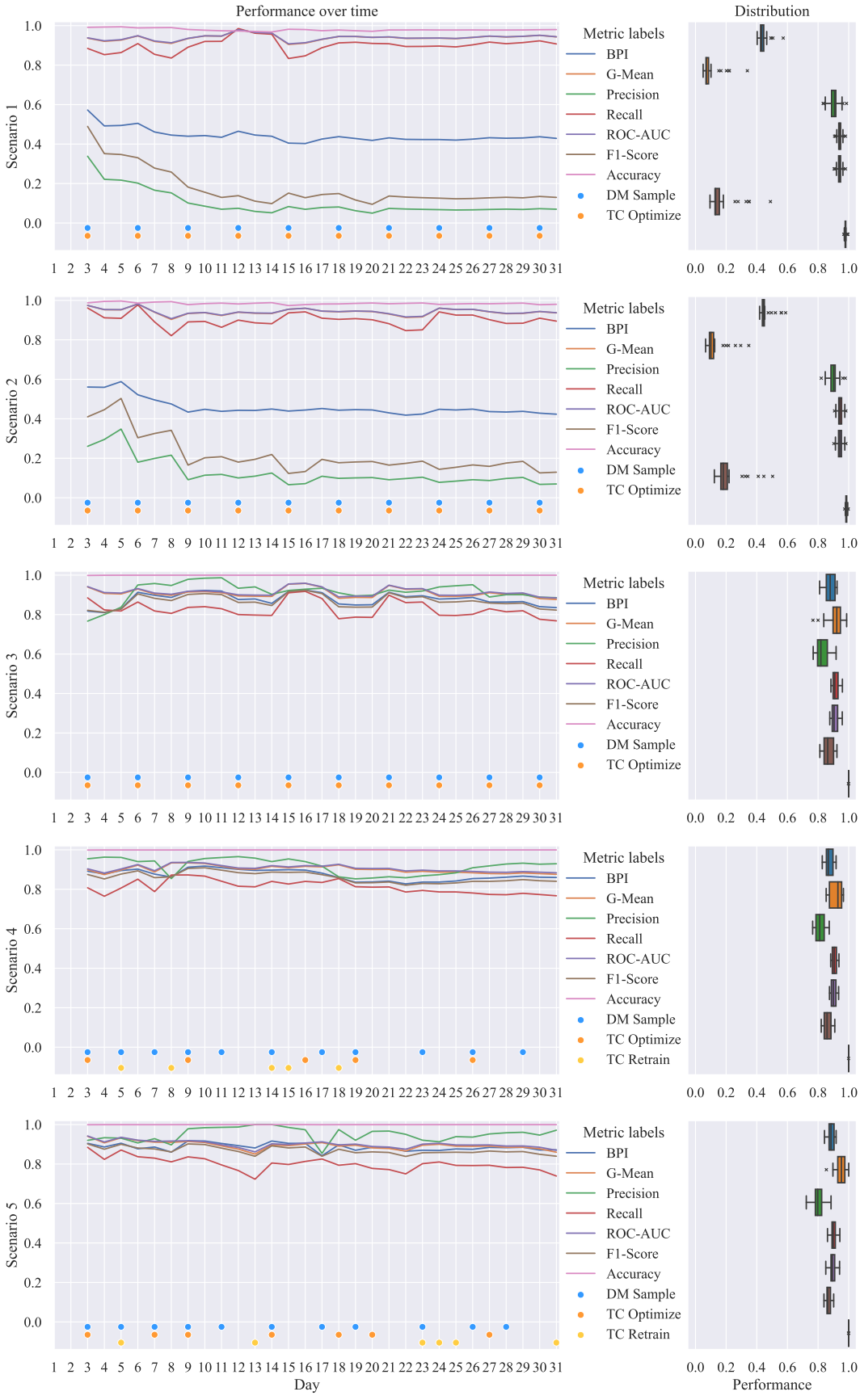
Accordingly, Figure 31 presents performance metrics recorded throughout the evaluation of each scenario, as well as triggers for the most computationally costly processes – training data generation by the *Data Manager* and model training by the *Training Controller*. The figure represents data generation as "DM Sample" and divides model training into "TC Optimize" or "TC Retrain", following the premises described at the end of Section 5.2 and decision from the flow chart in Figure 27. In addition, Figure 32 isolates BPI over time for tested scenarios.

The first 3 scenarios (S1-S3) simulated results with a baseline solution, applying standard oversampling and ML model with periodical updates every 3 days. Specifically, scenarios 1 and 2 were more susceptible to variations since they employed a simpler ML model (LR). Consequently, Figure 31 shows that these models started with performance metrics superior to those in later evaluations – which stabilized around the 11th day. These results occur due to a higher concentration of frauds within the first third of the ULB-CCFD dataset, illustrated through the IR in Figure 33a. The changes within dataset characteristics also validate the foundation of continuous performance monitoring in order to adapt models after instance space changes (HAPKE; NELSON, 2020).

While S2 presented only a slight advantage in performance compared to S1, this difference induces relevant improvement for the application. Particularly in the case of imbalanced data, the 3% variance in average BPI improved the balance between precision and recall – reducing total FP by 26%, consequently decreasing credit card disruptions and user friction, one of the main concerns for CCFD. Additionally, every metric improved their average performance. However, the improvement came at a cost in processing time. While S1 spent an average of 4 minutes in training, S2 spent 203% more time, at approximately 11 minutes. Table 16 presents processing time and average performance metrics throughout S1-S5.

Hence, S2 ascertains a cost-effective functionality for imbalanced data: probability threshold optimization. This technique achieved interesting results in improving a targeted performance metric – especially BPI, improving average precision by 25% while retaining other metrics. However, even though probability threshold optimization enhances solutions for imbalanced data, the standard ML model LR could not achieve acceptable performance ($BPI > 0.8$)

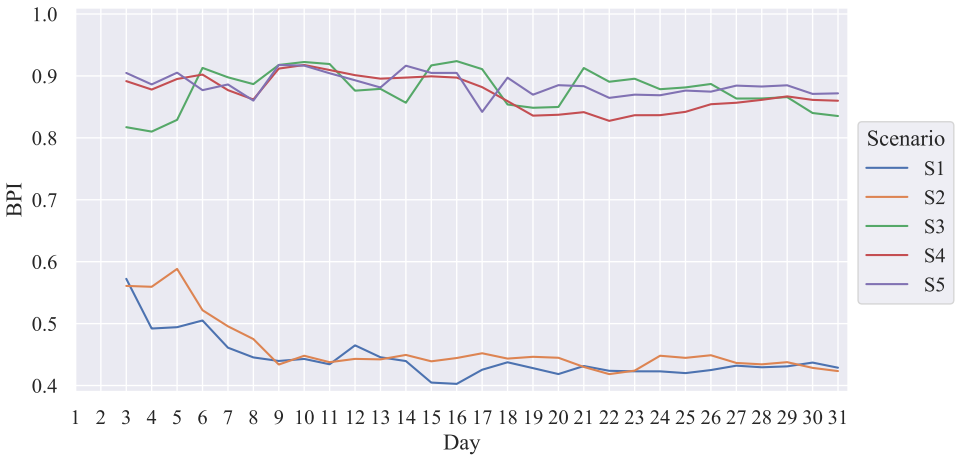
Figure 31: Performance metrics and agents' triggers over time for tested scenarios



Source: Created by the author.

in extremely low IR applications. Namely, scenarios 1 and 2 achieved their higher BPIs, between $0.5 < BPI < 0.6$, only in the first 4 days – when $IR > 0.25\%$ (Figure 33a). Additionally, the exposure to changes in data characteristics created a high variability in results – which is noticeable generally through the box plots of Figure 31 and standard deviation in Table 16.

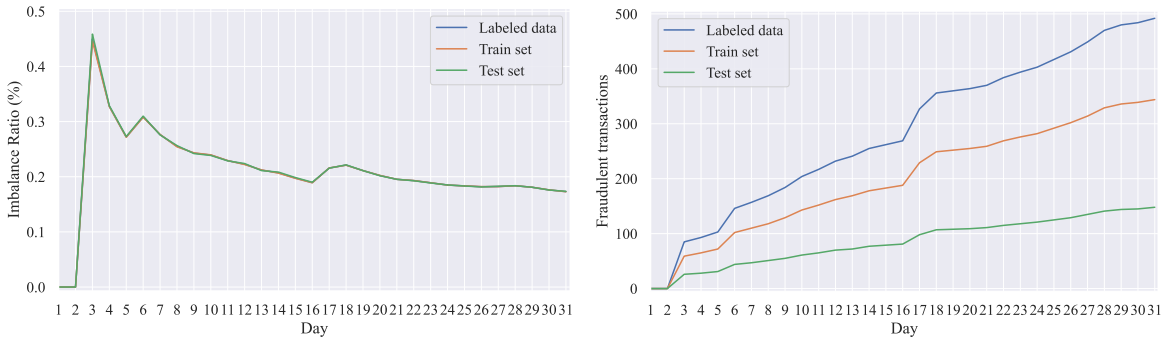
Figure 32: Comparison of BPI over time for tested scenarios



Source: Created by the author.

In contrast, scenarios 3 to 5, applying ensemble learning through RF, achieved $BPI > 0.8$ throughout the entire month – even at the lowest IR (0.173%) on day 31. As expected from the systematic mapping in Section 3.1, ensemble ML added processing complexity in S3, causing a 110% increment in average training time against S2. Nevertheless, applying the preferred sampling and ML from the finance domain with probability threshold optimization significantly improved targeted performance metrics in S3. For instance, BPI and F-Measure, measures of PR equilibrium, enhanced an average of 91% and 302%, respectively. The upgrade occurred primarily due to a 622% average increase in precision, sacrificing recall with a slight decrease of 8%.

Figure 33: Data characteristics over time: (a) Imbalance Ratio; (b) fraudulent transactions



(a) (b)

Source: Created by the author.

Interestingly, S3 allows the examination of a general decay in performance after periodical updates every 3 days – through data generation and model training. Even though the update interval is low, there is high variability in performance within each period of Figure 31. This variability can also be seen through the high standard deviation in Table 16.

Table 16: Analyses of performance metrics and processing times for S1-S5

Analysis	Metric	Scenario				
		1	2	3	4	5
Average performance	BPI	0.44341	0.45863	0.87736	0.87223	0.88570
	G-Mean	0.93888	0.94186	0.90931	0.90056	0.89481
	Precision	0.10094	0.12670	0.91477	0.91781	0.94975
	Recall	0.90004	0.90119	0.82749	0.81146	0.80030
	ROC-AUC	0.93995	0.94296	0.91365	0.90565	0.90057
	F-Measure	0.17493	0.21572	0.86778	0.86065	0.86835
	Accuracy	0.97970	0.98457	0.99947	0.99945	0.99949
Standard deviation in performance	BPI	0.03405	0.04305	0.03251	0.02662	0.01809
	G-Mean	0.01583	0.01644	0.02221	0.01789	0.01953
	Precision	0.06561	0.06964	0.04823	0.03899	0.03351
	Recall	0.03448	0.03260	0.04076	0.03237	0.03439
	ROC-AUC	0.01502	0.01566	0.02036	0.01617	0.01750
	F-Measure	0.09337	0.09668	0.03164	0.02557	0.01883
	Accuracy	0.00668	0.00510	0.00026	0.00014	0.00010
Average time	Data generation	00:00:06	00:00:05	00:00:04	00:00:06	01:56:42
	Training	00:03:47	00:11:26	00:24:01	00:12:46	00:28:28
Total time	Data generation	00:01:01	00:00:47	00:00:41	00:01:05	21:23:42
	Training	00:37:45	01:54:21	04:00:07	02:07:36	06:10:02

Source: Created by the author.

The inclusion of the reactive agents *Data Manager* and *Training Controller* in S4 enabled automated triggers for data generation and model training. Hence, negative changes in performance were quickly dealt with by *Training Controller*. As a result, the variability decreased by approximately 20% for every performance metric compared to S3 – while maintaining roughly the same average performance.

The automated triggers in S4 show the importance of real-time monitoring of newly labeled data and performance metrics. When the fraud rate increased rapidly, changing IR between days 3-13 and 16-19 (Figure 33a), the frequency of "DM Sample" triggers increased at a maximum of every 2 days. Conversely, when the IR stabilized and labeled data did not accumulate 7% more frauds, data generation awaited for even 4 days without regenerating training data. The triggering behavior and performance results due to the inclusion of both reactive agents show that their governance enables Heimdall to withstand drastic changes in instance space.

Additionally, the *Training Controller* also prioritized retraining over optimized training every time by triggering "TC Retrain" when perceiving a $1.5 - 3\%$ drop in BPI, anticipating a superior decrease and preventing the need for the computationally expensive process. Training

triggers also concentrated within the first 3 weeks, when data characteristics changed the most. However, after day 19, the agent only triggered "TC Optimized" due to the periodicity premise of an optimization every week (after 7 days).

On the whole, the results from process automation through the proposed reactive agents in S4 indicate extensive gains in efficiency for the CCFDS:

- Similar average performance to S3;
- Increased performance stability through lower standard deviation than S3 and previous scenarios;
- Decreased total processing time, in data generation and training, by 47%;
- Adaptive efficiency of performance to processing time – decreased in warm-up conditions by frequently triggering data generation and model training to achieve acceptable performance, and increased by reducing triggers in hot conditions, when labeled data stabilized instance space.

After increasing efficiency through reactive agents in S4, replacing standard oversampling (SMOTE) with high-performance oversampling (CGAN) yielded even better results in S5. This sampling technique increased average BPI and precision by approximately 1.5% and 3.5%, respectively, sacrificing recall by 1.4%. However, as expected from the literature, these results come with a significant computational resource cost, as suggested by Vargas et al. (2022). Namely, the total data generation time took almost 22 hours, increasing by $118,395\%$ compared to S4.

Comparatively, the conjunction of standard sampling with ensemble learning (S3) stands for excellent results – achieving better stability and efficiency when controlled by reactive agents (S4). In contrast, combining high-performance oversampling with ensemble learning (S5) produces the best results and stability overall, even in warm-up conditions. Hence, if data generation time meets the offline gap or if the associated processing load does not impact the external system, high-performance oversampling provides a straightforward path for better results.

Moreover, in addition to the analysis over time, Table 17 compares overall results from the tested configurations in S1-S5 against best-performing models mapped by Priscilla and Prabha (2019) using the whole ULB-CCFD dataset. These tests apply the same strategy of 70-30% stratified train-test split with 5-fold cross-validation. S4 was not tested since it has the same configuration as S3, adding reactive agents to independently govern processes over time.

Table 17 shows a common problem in related works: using ROC-AUC or accuracy as an objective metric for imbalanced data, ignoring the importance of the equilibrium defined for precision and recall. Ultimately, the evaluation of 3 studies considered several metrics and visibly favored precision over recall, consequently reducing FP (AWOYEMI; ADETUNMBI; OLUWADARE, 2017; FIORE et al., 2019; RAZA; QAYYUM, 2019). These works corroborate the importance of evaluating the most meaningful metrics for improving imbalanced datasets (JIANG et al., 2019).

In this environment, Heimdall's S2 shows the importance of probability thredhold opti-

Table 17: Performance comparison between tested configurations and related works

Work	Wang et al. (2018)	Saia and Carta (2019)	Awoyemi, Adetunmbi and Oluwadare (2017)	Pumsirirat and Yan (2018)	Fiore et al. (2019)	Raza and Qayyum (2019)	**Heimdall S1**	**Heimdall S2**	**Heimdall S3**	**Heimdall S5**
Sampling	CC	SMOTE	US+OS	-	GAN	-	**SMOTE**	**SMOTE**	**SMOTE**	**CGAN**
ML model	RF	RF	KNN	AE	DL+SF	AdaBoost	**LR**	**LR**	**RF**	**RF**
BPI(8,6)	-	-	**0.92650**	-	**0.84666**	**0.87429**	**0.43071**	**0.63475**	**0.87263**	**0.89914**
G-Mean	-	-	-	-	-	-	**0.90724**	**0.62596**	**0.89287**	**0.90417**
Precision	-	-	1.00000	-	0.93204	0.94500	**0.13043**	**0.81690**	**0.92913**	**0.96032**
Recall	-	-	0.82850	-	0.73282	0.78000	**0.83108**	**0.39189**	**0.79730**	**0.81757**
ROC-AUC	0.96520	0.98000	-	0.96030	-	-	**0.91073**	**0.69587**	**0.89860**	**0.90875**
F-Measure	-	0.93000	-	-	0.82051	0.85400	**0.22548**	**0.52968**	**0.85818**	**0.88321**
Specificity	-	-	1.00000	-	0.99994	-	**0.99039**	**0.99985**	**0.99989**	**0.99994**
Accuracy	-	-	0.97150	-	0.99963	-	**0.99011**	**0.99879**	**0.99954**	**0.99963**

Source: Created by the author.

mization more visibly, achieving a 47% increase in BPI by increasing precision five-fold and sacrificing recall by 53%.

Two of the best-performing related works comply with the findings for the finance domain (Section 3.1.3.7) – applying GAN oversampling and a DL model (FIORE et al., 2019), as well as ensemble learning (RAZA; QAYYUM, 2019). However, the best result claimed by related works does not describe the hybrid sampling techniques, impairing further analysis (AWOYEMI; ADETUNMBI; OLUWADARE, 2017).

Finally, Heimdall's tested configurations presented increasingly better results from scenarios 1, 2, 3, and 5. While S3 achieved an acceptable performance, S5 achieved the best performance overall, combining the best takeaways from reviewed works (Table 8) and applying high-performance oversampling with ensemble learning. These results are comparable to the best-performing related works (AWOYEMI; ADETUNMBI; OLUWADARE, 2017; RAZA; QAYYUM, 2019).

To sum up, answering the questions posed at the end of Section 5.2, high-performance oversampling and ensemble learning do enable deployment in warm-up conditions. In fact, even ensemble learning with standard sampling can achieve acceptable results. However, it is essential that the frequency of training under these conditions must be higher than in hot conditions.

In this sense, the proposed reactive agents (*Data Manager* and *Trainer Controller*) introduce a flexible solution for efficiently automating CCFDS processes – increasing processing in warm-up conditions and later maintaining performance within predefined ranges. In addition to the flexibility of rule-based agents, Heimdall solutions applied to CCFD presented prime and stable results throughout tests over time and when compared to related works. Thus, Heimdall offers an excellent set of solutions for imbalanced data applications, tested for CCFD in an extremely low IR.

5.4 Final considerations

This chapter detailed the implementation of test environments based on Heimdall (Section 5.2) focused on evaluating the architecture and novel proposed features, as well as comparing results of proposed practices against related works (Section 5.3). Hence, the evaluation applied a well-known and highly imbalanced dataset for CCFD – explored in Section 5.1.

Ultimately, the results from Heimdall's experimental evaluation validate the following assertions:

- Performance over time is an important method to validate performance of ML solutions in imbalanced data applications, specially in warm-up conditions – when there are few labeled anomalies and the IR changes considerably;
- The combination of high-performance oversampling and ensemble learning yields the best results for imbalanced data overall. However, the efficiency of performance to pro-

cessing power is much greater for ensemble learning – producing excellent results even when combined with standard sampling techniques;

- Probability threshold optimization can significantly improve a ML model's performance. Incidentally, BPI (Equation 4.1) proved to be a flexible indicator to measure and balance PR according to the application's priorities within this optimization – as opposed to standard metrics in related works;
- Automating ML pipelines through the 2 proposed reactive agents, responsible for independent governance of data and learning – *Data Manager* and *Training Controller*, respectively –, attains adaptive efficiency. Consequently, these agents produce better and more stable performance by sacrificing efficiency in warm-up conditions, and maintaining excellent performance and efficiency in hot conditions;
- Heimdall is a resourceful architecture to implement online prediction systems in applications suffering from data imbalance. The functionalities and rulesets proposed in Chapters 4 and 3, partially tested in Section 5.2 and analyzed Section 5.3, present excellent results over time and indicate a promising pathway for future developments.

6 CONCLUSION

In a time of rapid technological growth and data availability, ML plays a significant role in process automation. However, a large proportion of real-world problems suffer from data imbalance, impairing their capacity to apply ML solutions.

Hence, this dissertation introduced Heimdall, an architecture for online ML through imbalanced data. The proposed architecture closes current research gaps, proposing new functionalities and compiling a set of good practices for developing real-time ML systems to solve the imbalance problem through sampling techniques.

This study began by presenting a theoretical background on imbalanced data and supervised ML, laying the foundations for each subject. Subsequently, this work performed three literature reviews on the subjects. The first review systematically mapped the state of the art on sampling techniques and ML models for imbalanced data applications of various domain areas – investigating their performance and domain preference, defining new taxonomies, and compiling lessons learned for new solutions. The last two reviews searched for software architectures focused on active learning from imbalanced data and efficient supervised learning designs, gathering relevant functionalities.

In effect, Heimdall unified the essential concepts from reviewed architectures and detailed a set of rules and algorithms for imbalanced data applications. In addition, the proposed architecture innovated from existing research by proposing a new indicator (BPI) and two reactive agents to guide and optimize active learning through imbalanced data. After verifying their contribution separately, the combination of these three novel functionalities proved to significantly improve the prototype's performance in comparison to related works and enable fully automated pipelines through adaptive efficiency – producing better and more sable performance by sacrificing efficiency in warm-up condition, and later maintaining excellent performance and efficiency in hot conditions.

6.1 Contributions

This dissertation has the following main scientific contributions:

- Consolidation of best practices in sampling techniques and ML models in imbalanced data applications, as well as topologies and functionalities of software architectures for online learning systems – based on literature reviews;
- BPI: a flexible metric to evaluate a ML model's performance in imbalanced data – focused on balancing PR according to the application's needs and further enhancing probability threshold optimization;
- Adaptive efficiency: a strategy for ML automation through two reactive agents independently managing data and ML according to predefined rulesets – enabling better and more stable performance by sacrificing efficiency in warm-up conditions, and maintaining ex-

cellent performance and efficiency in hot conditions;

- Heimdall: an architecture for ML in imbalanced data, compiling consolidated best practices, probability threshold optimization through BPI, data and model governance through reactive agents, among other functionalities to extend artificial intelligence capabilities for existing systems.

Hence, this dissertation is a valuable reference for future online ML systems through imbalanced data of any domain area – integrating state of the art and novel proposed solutions.

6.2 Future works

While the ULB-CCFD dataset enables fair comparisons to related works and the CCFDS tests focus on evaluating the solutions for ML with imbalanced data proposed in Heimdall, specifically the most innovative – such as BPI and reactive agents –, only some of Heimdall's functionalities have been implemented. The proposed architecture describes several preprocessing and ML algorithms that would significantly improve online predictions in imbalanced data applications with raw datasets.

In this sense, Heimdall presents opportunities for future works by implementing the architecture and rulesets defined in this research to solve real-world problems explored in all domains of Section 3.1.3.2. For instance, Supervisory Control And Data Acquisition (SCADA) systems could apply time series processes, such as anomaly labeling (Figure 23), and outsource complex analyses through the *Examiner*.

Additionally, new works could improve upon the evaluation of performance over time by expanding the analyzed time horizon and exploring data flow in smaller streams or batches. The current work was limited to sending daily batches due to implementation time constraints, as well as due to the dataset's size and IR. Moreover, efficiency of performance over time could be improved through the application of intelligent multi-agents, rather than rule-based reactive agents.

The proposed topology is highly efficient for local applications – such as SCADA systems. However, future works could expand on this research's findings by generalizing for distributed/cloud systems.

Finally, the literature reviews from Chapter 3 encompassed works mostly up to 2021 in accordance to the dissertation's progress. Nevertheless, the advances and accelerated interest in generative artificial intelligence applications after 2021 may yield new solutions applying high-performance oversampling, such as GAN and its variations.

6.3 Publications

The studies for this dissertation currently produced two research papers. The first, published in the journal "Knowledge and Information Systems", surveying works on sampling techniques

for ML with imbalanced data (VARGAS et al., 2022). The second, proposing Heimdall and evaluating the prototype described in this dissertation, submitted to the same journal. Hence:

1. VARGAS, V. W.; ARANDA, J. A. S.; COSTA, R. S.; PEREIRA, P. R. S.; BARBOSA, J. L. V. Imbalanced data preprocessing techniques for Machine Learning: a systematic mapping study. *Knowledge and Information Systems*. ISSN 0219-3116. *Status: published in 09/11/22*. DOI: https://doi.org/10.1007/s10115-022-01772-8.
2. VARGAS, V. W.; ARANDA, J. A. S.; COSTA, R. S.; PEREIRA, P. R. S.; BARBOSA, J. L. V. A software architecture for online Machine Learning in imbalanced data applications. *Knowledge and Information Systems*. ISSN 0219-3116. *Status: submitted in 23/09/23*.

The subsequent step for this research consists of adapting the developed prototype for online power system fault prediction (engineering domain) in a real-world SCADA for distribution systems – applying short circuit and power flow simulations for high-performance oversampling, a research gap explored in Section 3.1.3.4.

In addition to the dissertation research, the author supported the publication of three more papers:

1. ARANDA, J. A. S.; COSTA, R. S.; VARGAS, V. W.; PEREIRA, P. R. S.; BARBOSA, J. L. V.; VIANNA, M. P. Context-aware Edge Computing and Internet of Things in Smart Grids: A systematic mapping study. *Computers and Electrical Engineering*. ISSN 0045-7906. *Status: published in 02/03/22*. DOI: https://doi.org/10.1016/j.compeleceng.2022.107826.
2. COSTA, R. S.; ARANDA, J. A. S.; VARGAS, V. W.; PEREIRA, P. R. S.; BARBOSA, J. L. V.; VIANNA, M. P. Data Analysis Techniques Applied to Distribution Systems: A Systematic Mapping Study. *Electric Power Components and Systems*. ISSN 1532-5016. *Status: published in 13/02/23*. DOI: https://doi.org/10.1080/15325008.2023.2175927.
3. ARANDA, J. A. S.; COSTA, R. S.; VARGAS, V. W.; PEREIRA, P. R. S.; BARBOSA, J. L. V.; VIANNA, M. P.; SILVA, E. L. M. OntoFreya: A Power distribution ontology for electric metrics classification. *Applied Ontology*. ISSN 1570-5838. *Status: submitted in 19/05/23*.

REFERENCES

AHMED, C. M.; RAMAN, G.; MATHUR, A. P. Challenges in machine learning based approaches for real-time anomaly detection in industrial control systems. **Proceedings of the 6th ACM on Cyber-Physical System Security Workshop**, [S.l.], v. 7, 2020.

ALTEXSOFT. **Fraud detection, machine learning in fintech and ecommerce**. 2021.

AMARASINGHE, T.; APONSO, A.; KRISHNARAJAH, N. Critical analysis of machine learning based approaches for fraud detection in financial transactions. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING TECHNOLOGIES, 2018., 2018, New York, USA. **Proceedings...** ACM, 2018. p. 12–17.

AWOYEMI, J. O.; ADETUNMBI, A. O.; OLUWADARE, S. A. Credit card fraud detection using machine learning techniques: a comparative analysis. In: INTERNATIONAL CONFERENCE ON COMPUTING NETWORKING AND INFORMATICS (ICCNI), 2017., 2017. **Anais...** Institute of Electrical and Electronics Engineers Inc., 2017. v. 2017-January, p. 1–9.

BALDOMINOS, A. et al. A scalable machine learning online service for big data real-time analysis. In: IEEE SYMPOSIUM ON COMPUTATIONAL INTELLIGENCE IN BIG DATA (CIBD), 2014., 2015. **Anais...** IEEE, 2015. p. 1–8.

BARATA, R. et al. Active learning for imbalanced data under cold start. **Proceedings of the Second ACM International Conference on AI in Finance**, [S.l.], p. 1–9, 11 2021.

BATYUK, A.; VOITYSHYN, V.; VERHUN, V. Software architecture design of the real- time processes monitoring platform. In: IEEE 2ND INTERNATIONAL CONFERENCE ON DATA STREAM MINING AND PROCESSING, 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p. 98–101.

BENHAR, H.; IDRI, A.; FERNÁNDEZ-ALEMÁN, J. L. Data preprocessing for heart disease classification: a systematic literature review. **Computer Methods and Programs in Biomedicine**, [S.l.], v. 195, p. 105635, 2020.

BHATORE, S.; MOHAN, L.; REDDY, Y. R. Machine learning techniques for credit risk evaluation: a systematic literature review. **Journal of Banking and Financial Technology**, [S.l.], v. 4, n. 1, p. 111–138, 2020.

BROWNLEE, J. **Imbalanced classification with python**: choose better metrics, balance skewed classes, and apply cost-sensitive learning. v1.3. ed. [S.l.]: Independently published, 2021.

BUDA, M.; MAKI, A.; MAZUROWSKI, M. A. A systematic study of the class imbalance problem in convolutional neural networks. **Neural Networks**, [S.l.], v. 106, p. 249–259, 2018.

CERQUITELLI, T. et al. Towards a real-time unsupervised estimation of predictive model degradation. In: ACM INTERNATIONAL CONFERENCE PROCEEDING SERIES, 2019. **Anais...** [S.l.: s.n.], 2019. p. 1–6.

CHANG, Q.; LIN, S.; LIU, X. Stacked-svm: a dynamic svm framework for telephone fraud identification from imbalanced cdrs. In: ACAI 2019: PROCEEDINGS OF THE 2019 2ND INTERNATIONAL CONFERENCE ON ALGORITHMS, COMPUTING AND ARTIFICIAL INTELLIGENCE, 2019, New York, USA. **Anais...** ACM, 2019. v. 9, p. 112–120.

CHOI, J.; JEON, C. Cost-based heterogeneous learning framework for real-time spam detection in social networks with expert decisions. **IEEE Access**, [S.l.], v. 9, p. 103573–103587, 2021.

CHUGH, G.; KUMAR, S.; SINGH, N. Survey on machine learning and deep learning applications in breast cancer diagnosis. **Cognitive Computation**, [S.l.], p. 1–20, 2021.

COHEN, G. et al. Learning from imbalanced data in surveillance of nosocomial infection. **Artificial Intelligence in Medicine**, [S.l.], v. 37, n. 1, p. 7–18, 2006.

COOPER, I. D. What is a "mapping study?". **Journal of the Medical Library Association**, [S.l.], v. 104, n. 1, p. 76–78, jan 2016.

De Almeida, L. G. et al. Data analysis techniques in vehicle communication networks: systematic mapping of literature. **IEEE Access**, [S.l.], v. 8, p. 199503–199512, oct 2020.

DEWI, C. et al. Improve performance of extreme learning machine in classification of patchouli varieties with imbalanced class. In: SIET '20: PROCEEDINGS OF THE 5TH INTERNATIONAL CONFERENCE ON SUSTAINABLE INFORMATION ENGINEERING AND TECHNOLOGY, 2020, New York, USA. **Anais...** ACM, 2020. p. 16–22.

DONG, Y.; WANG, X. A new over-sampling approach– random-smote for learning from imbalanced data sets. In: KSEM 2011: 5TH INTERNATIONAL CONFERENCE ON KNOWLEDGE SCIENCE, ENGINEERING AND MANAGEMENT, 2011, Irvine, USA. **Anais...** Springer, 2011. p. 343–352.

DäSTNER, K. et al. Classification of military aircraft in real-time radar systems based on supervised machine learning with labelled ads-b data. **2018 Symposium on Sensor Data Fusion: Trends, Solutions, Applications, SDF 2018**, [S.l.], 11 2018.

FARIS, H. et al. Improving financial bankruptcy prediction in a highly imbalanced class distribution using oversampling and ensemble learning: a case from the spanish market. **Progress in Artificial Intelligence**, [S.l.], v. 9, n. 1, p. 31–53, 2020.

FELIX, E. A.; LEE, S. P. Systematic literature review of preprocessing techniques for imbalanced data. **IET Software**, [S.l.], v. 13, n. 6, p. 479–496, 2019.

FERNáNDEZ, A. et al. **Foundations on imbalanced classification**. [S.l.]: Springer, Cham, 2018. 19-46 p.

FILHO, A. H. et al. Imbalanced learning techniques for improving the performance of statistical models in automated essay scoring. In: KNOWLEDGE-BASED AND INTELLIGENT INFORMATION & ENGINEERING SYSTEMS: PROCEEDINGS OF THE 23RD INTERNATIONAL CONFERENCE KES2019, 2019, Budapest, Hungary. **Anais...** Elsevier B.V., 2019. v. 159, p. 764–773.

FIORE, U. et al. Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. **Information Sciences**, [S.l.], v. 479, p. 448–455, 4 2019.

FOTOUHI, S.; ASADI, S.; KATTAN, M. W. A comprehensive data level analysis for cancer diagnosis on imbalanced data. **Journal of Biomedical Informatics**, [S.l.], v. 90, p. 103089, 2019.

GANGWAR, A. K.; RAVI, V. Wip: generative adversarial network for oversampling data in credit card fraud detection. In: ICISS 2019: 15TH INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS SECURITY, 2019, Hyderabad, India. **Anais...** Springer, 2019. v. 11952, p. 123–134.

GICIĆ, A.; SUBASI, A. Credit scoring for a microcredit data set using the synthetic minority oversampling technique and ensemble classifiers. **Expert Systems**, [S.l.], v. 36, n. 2, p. 1–22, 2018.

GOOGLE. **Classification**: precision and recall | machine learning crash course | google developers. 2020.

GUZELLA, T. S.; CAMINHAS, W. M. A review of machine learning approaches to spam filtering. **Expert Systems with Applications**, [S.l.], v. 36, n. 7, p. 10206–10222, 2009.

GéRON, A. **Hands-on machine learning with scikit-learn and tensorflow**: concepts, tools, and techniques to build intelligent systems. 1st. ed. [S.l.]: O'Reilly, 2017.

HALDAR, S. et al. Improved epilepsy detection method by addressing class imbalance problem. In: IEEE 9TH ANNUAL INFORMATION TECHNOLOGY, ELECTRONICS AND MOBILE COMMUNICATION CONFERENCE (IEMCON), 2018., 2019, Vancouver, BC, Canada. **Anais...** IEEE, 2019. p. 934–939.

HAN, X. et al. A gaussian mixture model based combined resampling algorithm for classification of imbalanced credit data sets. **International Journal of Machine Learning and Cybernetics**, [S.l.], v. 10, n. 12, p. 3687–3699, 2019.

HANUSSEK, M.; BLOHM, M.; KINTZ, M. Can AutoML outperform humans? An evaluation on popular OpenML datasets using AutoML Benchmark. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, ROBOTICS AND CONTROL, 2020., 2020, Cairo, Egypt. **Anais...** ACM, 2020. p. 29–32.

HAPKE, H.; NELSON, C. **Building machine learning pipelines automating model life cycles with tensorflow**. 1st. ed. [S.l.]: O'Reilly, 2020.

HE, H.; MA, Y. **Imbalanced learning – foundations, algorithms and applications**. [S.l.]: Wiley, 2013.

HU, Z. et al. Deep learning for image-based cancer detection and diagnosis – a survey. **Pattern Recognition**, [S.l.], v. 83, p. 134–149, 2018.

IDRI, A. et al. A systematic map of medical data preprocessing in knowledge discovery. **Computer Methods and Programs in Biomedicine**, [S.l.], v. 162, p. 69–85, 2018.

ISHTIAQ, U. et al. Diabetic retinopathy detection through artificial intelligent techniques: a review and open issues. **Multimedia Tools and Applications**, [S.l.], v. 79, p. 15209–15252, 2020.

ITOO, F.; MEENAKSHI; SINGH, S. Comparison and analysis of logistic regression, naïve bayes and knn machine learning algorithms for credit card fraud detection. **International Journal of Information Technology**, [S.l.], p. 1–9, 2020.

JIANG, J. et al. A novel multi-module neural network system for imbalanced heartbeats classification. **Expert Systems with Applications: X**, [S.l.], v. 1, p. 100003, 2019.

JIANG, N.; LI, N. A wind turbine frequent principal fault detection and localization approach with imbalanced data using an improved synthetic oversampling technique. **International Journal of Electrical Power and Energy Systems**, [S.l.], v. 126, Part A, p. 106595, 2021.

JOHNSON, J. M.; KHOSHGOFTAAR, T. M. Survey on deep learning with class imbalance. **Journal of Big Data**, [S.l.], v. 6, p. 1–54, 2019.

KAUR, H.; PANNU, H. S.; MALHI, A. K. A systematic review on imbalanced data challenges in machine learning: applications and solutions. **ACM Computing Surveys**, [S.l.], v. 52, n. 4, p. 1–36, 2019.

Keshav S. How to read a paper. **ACM SIGCOMM Computer Communication Review**, [S.l.], v. 37, n. 3, p. 83–84, jul 2007.

KITCHENHAM, B. et al. Systematic literature reviews in software engineering – a tertiary study. **Information and Software Technology**, [S.l.], v. 52, n. 8, p. 792–805, aug 2010.

LEE, Y. O.; KIM, Y. J. The effect of resampling on data-imbalanced conditions for prediction towards nuclear receptor profiling using deep learning. **Molecular Informatics**, [S.l.], v. 39, n. 8, p. 1900131, 2020.

LEI, Y. et al. Applications of machine learning to machine fault diagnosis: a review and roadmap. **Mechanical Systems and Signal Processing**, [S.l.], v. 138, p. 106587, 2020.

LI, Q.; MAO, Y. A review of boosting methods for imbalanced data classification. **Pattern Analysis and Applications**, [S.l.], v. 17, p. 679–693, 2014.

LI, X. et al. Building an online defect detection system for large-scale photovoltaic plants. **BuildSys 2019 - Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation**, [S.l.], p. 253–262, 11 2019.

LI, Z.; JING, X. Y.; ZHU, X. Progress on approaches to software defect prediction. **IET Software**, [S.l.], v. 12, n. 3, p. 161–175, 2018.

LIU, Q.; MA, G.; CHENG, C. Data fusion generative adversarial network for multi-class imbalanced fault diagnosis of rotating machinery. **IEEE Access**, [S.l.], v. 8, p. 70111–70124, 2020.

LIU, S. et al. Addressing the class imbalance problem in twitter spam detection using ensemble learning. **Computers and Security**, [S.l.], v. 69, p. 35–49, 2017.

MA, J. et al. Predicting seminal quality via imbalanced learning with evolutionary safe-level synthetic minority over-sampling technique. **Cognitive Computation**, [S.l.], p. 1–12, 2019.

MAHADEVAN, A.; AROCK, M. A class imbalance-aware review rating prediction using hybrid sampling and ensemble learning. **Multimedia Tools and Applications**, [S.l.], v. 80, n. 5, p. 6911–6938, 2021.

MALHOTRA, R. A systematic review of machine learning techniques for software fault prediction. **Applied Soft Computing**, [S.l.], v. 27, p. 504–518, 2015.

MALHOTRA, R.; KAMAL, S. An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data. **Neurocomputing**, [S.l.], v. 343, p. 120–140, 2019.

MALHOTRA, R.; LATA, K. An empirical study on predictability of software maintainability using imbalanced data. **Software Quality Journal**, [S.l.], v. 28, n. 4, p. 1581–1614, 2020.

MARQUÉS, A. I.; GARCÍA, V.; SÁNCHEZ, J. S. On the suitability of resampling techniques for the class imbalance problem in credit scoring. **Journal of the Operational Research Society**, [S.l.], v. 64, n. 7, p. 1060–1070, 2013.

MCMAHON, A. P. **Machine learning engineering with python**: manage the production life cycle of machine learning models using mlops with practical examples. 1. ed. [S.l.]: Packt Publishing Ltd., 2021. 1-277 p.

MURESAN, S. et al. Pre-processing flow for enhancing learning from medical data. In: IEEE 11TH INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTER COMMUNICATION AND PROCESSING, ICCP 2015, 2015., 2015, Cluj-Napoca, Romania. **Anais...** IEEE, 2015. p. 27–34.

MüLLER, A. C.; GUIDO, S. **Introduction to machine learning with python**: a guide for data scientists. 1st. ed. [S.l.]: O'Reilly, 2017.

NNAMOKO, N.; KORKONTZELOS, I. Efficient treatment of outliers and class imbalance for diabetes prediction. **Artificial Intelligence in Medicine**, [S.l.], v. 104, p. 101815, 2020.

PADGHAM, L.; WINIKOFF, M. **Developing intelligent agent systems – a practical guide**. [S.l.]: John Wiley and Sons Ltd, 2004.

PANDEY, S. K.; MISHRA, R. B.; TRIPATHI, A. K. Machine learning based methods for software fault prediction: a survey. **Expert Systems with Applications**, [S.l.], v. 172, p. 114595, 2021.

PEREIRA, R. M. et al. Covid-19 identification in chest x-ray images on flat and hierarchical classification scenarios. **Computer Methods and Programs in Biomedicine**, [S.l.], v. 194, p. 105532, 2020.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: an update. **Information and Software Technology**, [S.l.], v. 64, p. 1–18, aug 2015.

PRISCILLA, C. V.; PRABHA, D. P. Credit card fraud detection: a systematic review. In: FIRST INTERNATIONAL CONFERENCE ON INNOVATIVE COMPUTING AND CUTTING-EDGE TECHNOLOGIES (ICICCT 2019), 2019, Istanbul, Turkey. **Proceedings...** Springer, 2019. p. 290–303.

PUMSIRIRAT, A.; YAN, L. Credit card fraud detection using deep learning based on auto-encoder and restricted boltzmann machine. **International Journal of Advanced Computer Science and Applications**, [S.l.], v. 9, p. 18–25, 55 2018.

PURNAMI, S. W.; TRAPSILASIWI, R. K. Smote-least square support vector machine for classification of multiclass imbalanced data. In: ICMLC 2017: PROCEEDINGS OF THE 9TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND COMPUTING, 2017, New York, USA. **Anais...** ACM, 2017. p. 107–111.

RAZA, M.; QAYYUM, U. Classical and deep learning classifiers for anomaly detection. In: INTERNATIONAL BHURBAN CONFERENCE ON APPLIED SCIENCES AND TECHNOLOGY, IBCAST 2019, 2019., 2019. **Anais...** Institute of Electrical and Electronics Engineers Inc., 2019. p. 614–618.

REKHA, G.; REDDY, V. K.; TYAGI, A. K. An earth mover's distance-based undersampling approach for handling class-imbalanced data. **International Journal of Intelligent Information and Database Systems**, [S.l.], v. 13, n. 2-4, p. 376–392, 2020.

RUAN, H. et al. Deep learning-based fault prediction in wireless sensor network embedded cyber-physical systems for industrial processes. **IEEE Access**, [S.l.], v. 10, p. 10867–10879, 2022.

RUSTAM, Z. et al. Hybrid preprocessing method for support vector machine for classification of imbalanced cerebral infarction datasets. **International Journal on Advanced Science, Engineering and Information Technology**, [S.l.], v. 9, n. 2, p. 685–691, 2019.

SAIA, R.; CARTA, S. Evaluating the benefits of using proactive transformed-domain-based techniques in fraud detection tasks. **Future Generation Computer Systems**, [S.l.], v. 93, p. 18–32, 4 2019.

SANTOS, M. S. et al. A new cluster-based oversampling method for improving survival prediction of hepatocellular carcinoma patients. **Journal of Biomedical Informatics**, [S.l.], v. 58, p. 49–59, 2015.

SAP. **Standardized technical architecture modeling – conceptual and design level**. [S.l.: s.n.], 2007. (March).

SCIKIT-LEARN. **Support vector machines – complexity**. 2021.

SHAKEEL, F.; SABHITHA, A. S.; SHARMA, S. Exploratory review on class imbalance problem: an overview. In: INTERNATIONAL CONFERENCE ON COMPUTING, COMMUNICATION AND NETWORKING TECHNOLOGIES (ICCCNT), 2017., 2017, Delhi, India. **Anais...** IEEE, 2017. p. 1–8.

SHAMSUDIN, H. et al. Combining oversampling and undersampling techniques for imbalanced classification: a comparative study using credit card fraudulent transaction dataset. In: IEEE 16TH INTERNATIONAL CONFERENCE ON CONTROL AND AUTOMATION (ICCA), 2020., 2020, Singapore. **Anais...** IEEE, 2020. p. 803–808.

SILVA, R. D. A.; BRAGA, R. T. V. Simulating systems-of-systems with agent-based modeling: a systematic literature review. **IEEE Systems Journal**, [S.l.], v. 14, n. 3, p. 3609–3617, sep 2020.

SIRSAT, M. S.; FERMÉ, E.; CÂMARA, J. Machine learning for brain stroke: a review. **Journal of Stroke and Cerebrovascular Diseases**, [S.l.], v. 29, n. 10, p. 105162, 2020.

SMITI, S.; SOUI, M. Bankruptcy prediction using deep learning approach based on borderline smote. **Information Systems Frontiers**, [S.l.], v. 22, n. 5, p. 1067–1083, 2020.

SPELMEN, V. S.; PORKODI, R. A review on handling imbalanced data. In: INTERNATIONAL CONFERENCE ON CURRENT TRENDS TOWARDS CONVERGING TECHNOLOGIES (ICCTCT), 2018., 2018, Coimbatore, India. **Anais...** IEEE, 2018. p. 1–11.

SUSAN, S.; KUMAR, A. The balancing trick: optimized sampling of imbalanced datasets—a brief survey of the recent state of the art. **Engineering Reports**, [S.l.], v. 3, n. 4, p. 1–24, 2020.

TASHKANDI, A.; WIESE, L. A hybrid machine learning approach for improving mortality risk prediction on imbalanced data. In: WAS2019: PROCEEDINGS OF THE 21ST INTERNATIONAL CONFERENCE ON INFORMATION INTEGRATION AND WEB-BASED APPLICATIONS & SERVICES, 2019, New York, USA. **Anais...** ACM, 2019. p. 83–92.

THANOUN, M. Y.; YASEEN, M. T. A comparative study of parkinson disease diagnosis in machine learning. In: ICAAI 2020: 2020 THE 4TH INTERNATIONAL CONFERENCE ON ADVANCES IN ARTIFICIAL INTELLIGENCE, 2020, New York, USA. **Anais...** ACM, 2020. p. 23–28.

TRA, V.; DUONG, B. P.; KIM, J. M. Improving diagnostic performance of a power transformer using an adaptive over-sampling method for imbalanced data. **IEEE Transactions on Dielectrics and Electrical Insulation**, [S.l.], v. 26, n. 4, p. 1325–1333, 2019.

ULB. **Credit card fraud detection | kaggle**. 2018.

VARGAS, V. W. de et al. Imbalanced data preprocessing techniques for machine learning: a systematic mapping study. **Knowledge and Information Systems**, [S.l.], v. 65, p. 31–57, 1 2022.

VU, L. et al. Learning from imbalanced data for encrypted traffic identification problem. In: SOICT '16: PROCEEDINGS OF THE SEVENTH SYMPOSIUM ON INFORMATION AND COMMUNICATION TECHNOLOGY, 2016, New York, USA. **Anais...** ACM, 2016. p. 147–152.

WANG, H. et al. An ensemble learning framework for credit card fraud detection based on training set partitioning and clustering. In: IEEE SMARTWORLD/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI, 2018., 2018. **Anais...** IEEE, 2018. p. 94–98.

WANG, H.; YE, W. Transient stability evaluation model based on ssdae with imbalanced correction. **IET Generation, Transmission and Distribution**, [S.l.], v. 14, n. 11, p. 2209–2216, 2020.

WONG, G. Y.; LEUNG, F. H.; LING, S. H. A novel evolutionary preprocessing method based on over-sampling and under-sampling for imbalanced datasets. In: IECON 2013 - 39TH ANNUAL CONFERENCE OF THE IEEE INDUSTRIAL ELECTRONICS SOCIETY, 2014, Vienna, Austria. **Anais...** IEEE, 2014. p. 2354–2359.

YAN, K. et al. Unsupervised learning for fault detection and diagnosis of air handling units. **Energy and Buildings**, [S.l.], v. 210, p. 109689, 2020.

YAN, S. et al. Improving lung cancer prognosis assessment by incorporating synthetic minority oversampling technique and score fusion method. **Medical Physics**, [S.l.], v. 43, n. 6, p. 2694–2703, 2016.

ZHANG, C.; ZHOU, Y.; DENG, Y. Vcos: a novel synergistic oversampling algorithm in binary imbalance classification. **IEEE Access**, [S.l.], v. 7, p. 145435–145443, 2019.

ZHANG, J. et al. Imbalanced classification of mental workload using a cost-sensitive majority weighted minority oversampling strategy. **Cognition, Technology and Work**, [S.l.], v. 19, n. 4, p. 633–653, 2017.

ZHANG, T. et al. Intelligent fault diagnosis of machines with small & imbalanced data: a state-of-the-art review and possible extensions. **ISA Transactions**, [S.l.], 2021.

ZHANG, X. et al. Efficiently predicting hot spots in ppis by combining random forest and synthetic minority over-sampling technique. **IEEE/ACM Transactions on Computational Biology and Bioinformatics**, [S.l.], v. 16, n. 3, p. 774–781, 2019.

ZHAO, S. X.; WANG, X. long; YUE, Q. sheng. A novel mixed sampling algorithm for imbalanced data based on xgboost. In: CWSN 2020: 14TH CHINA CONFERENCE ON WIRELESS SENSOR NETWORKS, 2020, Dunhuang, China. **Anais...** Springer, 2020. p. 181–196.

ZHOU, L. Performance of corporate bankruptcy prediction models on imbalanced dataset: the effect of sampling methods. **Knowledge-Based Systems**, [S.l.], v. 41, p. 16–25, 2013.

ZHOU, Q. et al. K-means clustering based undersampling for lower back pain data. In: ICBDT 2020: PROCEEDINGS OF THE 2020 3RD INTERNATIONAL CONFERENCE ON BIG DATA TECHNOLOGIES, 2020, New York, USA. **Anais...** ACM, 2020. p. 53–57.