Developing and implementing watershed-based classification algorithms for imbalanced datasets : an application to the detection of acute chest syndrome in patients with sickle cell disease

Yamna Ouchtar

Université Gustave Eiffel

Developing and Implementing Watershed-based Classification Algorithms for Imbalanced Datasets: an Application to the Detection of Acute Chest Syndrome in patients with Sickle Cell Disease.

Thèse de doctorat de l'Université Gustave Eiffel

École doctorale n°532, Mathématiques et Sciences et Technologie de l'Information et de la Communication, MSTIC
Spécialité de doctorat: Informatique
Unité de recherche : Labortoire d'Informatique Gaspard-Monge, LIGM

Thèse présentée et soutenue à l'Université Gustave Eiffel, le 13/11/2023, par :

Yamna OUCHTAR

Composition du Jury

Hugues TALBOT Professeur, Université Gustave Eiffel	Président du jury
Valentine BROUSSE Docteur, Université de Paris - Cité	Examinatrice
Nora OUZIR Maitre de conférence, Central Supelec	Examinatrice
Alexandre Xavier FALCAO Professeur, University of Campinas (UNICAMP)	Rapporteur
Camille KURTZ Professeur, Laboratoire d'Informatique Paris Descartes	Rapporteur
Zahia ZAIDAT Chargée de Mission Recherche, Région Île-de-France	Invité

Encadrement de la thèse

Laurent NAJMAN Professeur, Université Gustave Eiffel	Directeur de thèse
Pablo BARTOLUCCI Professeur, Henri Mondor, Université Paris-Est Créteil	Co-Directeur de thèse
Benjamin PERRET Professeur, Université Gustave Eiffel	Co-Encadrant de thèse

Contents

Abstract **5**

Remerciements **7**

Anacronyms **8**

Résumé Long **9**

Introduction & Outline **13**
References 18

1 Acute Chest Syndrome Detection in Sickle Cell Patients: Exploring Literature and Challenges **20**
1.1 Sickle Cell Disease & Acute Chest Syndrome (ACS) 20
1.2 Predictive Severity Study (PRESEV) 23
1.2.1 Study Presentation 23
1.2.2 Data Exploration 24
1.2.3 Statistical Analysis 26
1.3 Detection of ACS: A Typical Imbalanced Classification Problem 29
1.3.1 Analyzing Previous Results 30
1.3.2 Transforming PRESEV into a Classification Problem 31
1.4 Conclusion 33
References 33

2 Imbalanced Classification Challenge: A Review of the Literature **35**
2.1 Description of Imbalanced Data 36
2.1.1 Imbalanced Data Challenges 37
2.1.2 Scores and Metrics 38
2.2 Imbalanced Methods 40
2.2.1 Data Level / Sampling Method 40
2.2.2 Algorithmic Methods 43
2.2.3 Cost-Sensitive Learning 46

2.3 Conclusion 46
References 47

3 ImbPip: A Pipeline for Comparing Imbalanced Methods 50
3.1 Pipeline Implementation 50
3.2 Applications 53
3.2.1 Results on real-world datasets 53
3.2.2 Results on PRESEV dataset 54
3.3 Conclusion 55
References 56

4 Bridging Image Detection Segmentation and Imbalanced Classification: Exploring the Watershed Theory 59
4.1 Exploring the Literature of Watershed Transformations . . 60
4.1.1 From Watersheds for Images... 60
4.1.2 To Watersheds for Classification 62
4.2 Watershed & Hierarchies 67
4.2.1 Ultrametric Distance 68
4.2.2 Watershed Hierarchies 70
4.3 Watershed and Imbalanced Dataset 70
4.3.1 Semi-Supervised Watershed for Imbalanced Data . 71
4.3.2 Watershed Cuts and Iterated Watershed for Imbalanced Data 73
4.4 Conclusion 75
References 76

5 WSSMOTE: a Novel Oversampling Method 78
5.1 Algorithm Implementation 78
5.2 Results 81
5.2.1 Results on PRESEV Dataset 82
5.2.2 Results on Randomly Selected Dataset 83
5.3 Conclusion 84
References 85

6 Exploring Metric Learning Theory & Application to Imbalanced Datasets 86
6.1 Metric Learning for Balanced Dataset 86
6.2 Metric Learning for Imbalanced Datasets 90
6.3 Metric Learning & Oversampling Comparisons 91
6.4 Conclusions 95
References 95

7 WML: an Ultrametric Learning Method for Imbalanced Data **97**
7.1 Algorithm Implementation 97
7.2 Results 99
7.2.1 Tests and Visualization 99
7.2.2 Performance of WML and WMLI on Random Imbalanced Datasets 102
7.2.3 Performance of WML and WMLI in Combination with Oversampling on Random Imbalanced Datasets 103
7.2.4 Performance of WML and WMLI in PRESEV . . . 105
7.3 Conclusions 105
References 105

Conclusions & Perspectives **107**

Appendices **110**
References 118

Abstract

Sickle cell disease is a rare, chronic and potentially fatal inherited disorder affecting red blood cells. In particular, acute chest syndrome (ACS) is a feared complication due to its association with increased mortality in hospitalized patients. To reduce this mortality, biomarker variables are examined on a patient's arrival in the emergency department, followed by a statistical prediction if the syndrome will occur. When making these statistical predictions, the use of conventional statistical methods or machine learning algorithms often leads to problems such as over-fitting and high variability. This is due to the detection of acute chest syndrome (ACS) being a typical example of an imbalanced classification problem. Our main objective is to find syndrome cases which represent a subset within the set of patients.

The concept of imbalanced datasets is key in this context. A binary dataset is considered imbalanced when there is a significant disparity between the occurrences of the minority class (in our case, the occurrence of ACS) and the majority class (patients unaffected by ACS). Traditional machine learning techniques are generally designed under the assumption of balanced datasets. Consequently, their application to imbalanced datasets often produces non-optimal performance and introduces biases. Due to these challenges, a high number of methods, called imbalanced methods, have been designed and implemented.

Nonetheless, our investigation unveiled the absence of an established framework to systematically compare distinct imbalanced methods. Furthermore, we will demonstrate that despite the profusion of methodologies, none of them yields an enhancement in the acute chest syndrome (ACS) detection. In response, a structured pipeline has been formulated to systematically assess and compare diverse imbalanced methods. Simultaneously, recognizing the inherent limitations of existing methodologies, an innovative approach using mathematical morphology with watershed approaches was proposed. The idea of combining mathematical morphology and more specifically watersheds is based on the fact that the detection of small objects in an image has been explored more deeply than the detection of small classes in a digital dataset. This concept is also based on the progressive correlation established between watersheds applied to

images and their application in classification. Consequently, we designed three distinct algorithms based on watershed graph theory and hierarchical principles. The first algorithm is rooted in the traditional oversampling approach, while the second algorithm adopts a method centered on learning based metric optimization. Finally, the third algorithm introduces an **algorithm-level** built on watershed hierarchies, specifically designed to reduce dependence on the dataset's structure.

The findings highlight the efficacy of watershed-based imbalanced classification algorithms in improving the detection of acute chest syndrome (ACS). These algorithms effectively address concerns related to overfitting and variability, translating to better reproducibility. For instance, the proposed watershed-based oversampling technique, called WSSMOTE, reduced overfitting from 13.3% to 1.2%. Moreover, the applicability of these algorithms can be extended to real-world classification problems. In certain cases, these algorithms led to improvements in performance metrics. For example, the accuracy of **some** real-world predictions was notably enhanced through the application of the watershed optimization metric learning method. This underlines the practical benefits and adaptability of the proposed watershed-based algorithms.

Keywords Watershed Transformations, Imbalanced Dataset, Framework development, Oversampling methods, Metric Learning methods, Hierarchy

Remerciements

Avant tout, je souhaite exprimer ma gratitude envers les membres de l'hôpital Mondor, pour leur patience, leur temps et leur accès aux multiples bases de données. Mes remerciements vont au P. Pablo Bartolucci, à Anne-Laure Pham-Hung d'Alexandry d'Orengian, ainsi qu'à Christian Kassasseya. Je tiens également à remercier chaleureusement Didier Caucal pour son suivi attentif.

Un immense merci à Benjamin Perret et Laurent Najman pour leur encadrement dévoué, leur disponibilité et leur expertise. Mes remerciements s'adressent également à la région Île-de-France et à Sidarth Radjou pour avoir financé et rendu possible cette thèse.

Je saisis cette occasion pour exprimer ma reconnaissance envers Carine Pivoteau pour m'avoir offert l'opportunité d'enseigner. Mes remerciements vont aussi aux autres membres du laboratoire : Thanh, Sarah, Caroline et Mariia.

Enfin, je ne saurais conclure sans adresser mes plus sincères remerciements à ma famille et à mes amis : Eïda, Lisa, Tasnime, Kinane, Camilo, Hassene, Fanny, Simon, Marine, Fabien, Thomas... Merci pour votre soutien et le temps que vous m'avez consacré.

Anacronyms

SCD	Sickle Cell Disease
ACS	Acute Chest Syndrom
PRESEV	Predictive Severity Study
ML	Machine Learning
SMOTE	Synthetic Minority Over-sampling Technique
MLT	Metric Learning Theory
WML	Watershed Metric Learning
WMLI	Watershed Metric Learning with Imbalanced Characteristics

Résumé Long

Le syndrome thoracique aigu (STA) constitue une complication sévère et mortelle de la drépanocytose, se manifestant environ 2,5 jours après l'admission hospitalière du patient. Bien que de nombreuses études aient été précédemment menées pour quantifier, comprendre et analyser ce syndrome, aucune n'avait jusqu'alors tenté d'anticiper l'apparition de ce phénomène dès l'arrivée des patients aux services d'urgence. Une nouvelle étude prospective novatrice, dénommée PRESEV, a donc été initiée, se composant de deux volets distincts, PRESEV1 et PRESEV2, visant à examiner le STA et à prédire son apparition au moyen de biomarqueurs. La première phase, PRESEV1, a été conduite exclusivement au sein de l'hôpital Mondor, tandis que PRESEV2 s'est déroulée dans divers centres médicaux à travers le monde. Ces études se fixent deux objectifs majeurs: tout d'abord, anticiper avec une forte valeur prédictive négative (VPN) les patients enclins au développement du STA, dans le but d'améliorer leur traitement et de réduire les taux de mortalité; simultanément, prévoir avec une valeur prédictive positive (VPP) élevée les patients peu susceptibles de développer un STA, pour améliorer l'attribution des ressources hospitalières et garantir, si nécessaire, un suivi à domicile.

Lors d'une première analyse, l'ensemble de données PRESEV1 a été soumis aux méthodes habituelles d'apprentissage automatique, produisant des résultats prometteurs. Cependant, lors de l'application de ces mêmes modèles à l'ensemble de données PRESEV2, des problèmes de forte variabilité et de surajustement sont survenus. Ces deux problèmes majeurs trouvent leur origine dans le déséquilibre marqué entre les classes au sein de PRESEV. En effet, à l'arrivée des patients aux urgences, seulement environ 20% d'entre eux développeront un STA. Ainsi, la base de données PRESEV présente deux classes déséquilibrées : une classe composée des patients présentant un STA (représentant 20% de la base de données) et une autre classe regroupant les patients qui n'évolueront pas vers un STA (représentant 80% des patients). Ce déséquilibre est une problématique récurrente dans de nombreuses applications, au-delà de la base de données que nous étudions, telles que la prédiction de maladies rares ou de fraudes bancaires. Pour contrer ces disparités de classe, d'innombrables approches, telles que le suréchantillonnage et l'utilisation d'algorithmes sensibles aux

coûts, ont été développées au fil du temps.

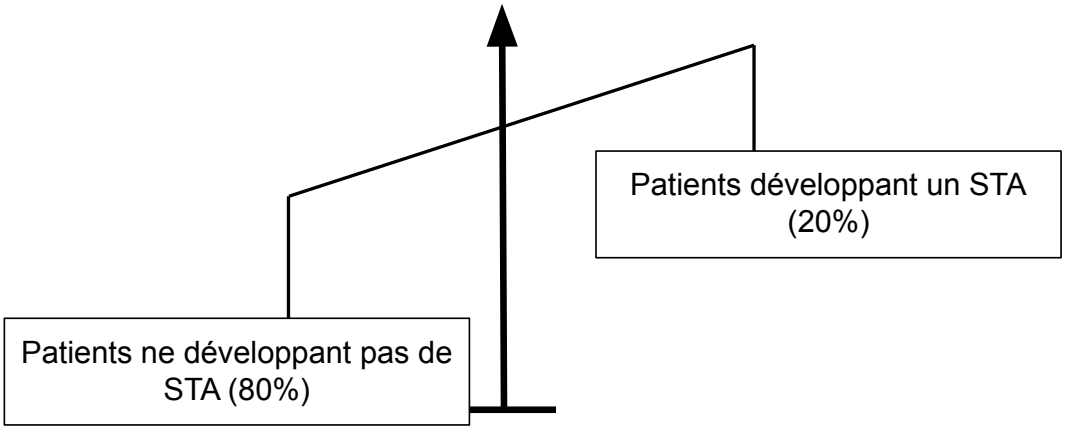


Figure 1: Illustration du déséquilibre lié à l'apparition ou non de STA au sein de la base de données PRESEV.

Cependant, l'application de ces méthodes de rééquilibrage se révèle complexe. En effet, la recherche de la méthode conduisant aux résultats optimaux s'avère ardue en raison de l'absence d'une bibliothèque centralisée regroupant les divers codes relatifs aux méthodes, ou d'une approche normalisée. De plus, ce défi est amplifié par l'absence de facteur directeur permettant d'opter pour une méthode plutôt qu'une autre. Par exemple, les rapports de déséquilibre et le nombre de caractéristiques ne suffisent pas à choisir une méthode de réequilibrage plutot qu'une autre. C'est pourquoi nous avons élaboré un pipeline, appelé ImbPip, visant à comparer plusieurs méthodes. Cette démarche garantit une reproductibilité ainsi que des comparaisons plus fiables, ce qui facilite l'analyse à la fois des scores et de la variabilité des méthodes. Ce pipeline a démontré son efficacité lors de tests sur diverses bases de données. Cependant, au cours de mes années de thèse, un pipeline similaire a été développé et rendu public. C'est donc ce dernier, dénommé "smote_variant", qui sera adopté et modifié lors de nos travaux futurs.

En utilisant ce pipeline, nous avons évalué différentes méthodes de réequilibrage sur notre ensemble de données PRESEV. Il est devenu évident qu'aucune stratégie n'était en mesure d'améliorer à la fois les scores de VPP et de VPN tout en réduisant la variabilité. Cette problématique nous a poussés à explorer d'autres approches, ce qui a abouti à l'intégration de concepts issus de la segmentation d'images en tant que stratégie de rééquilibrage. Plus spécifiquement, nous avons exploré l'application de l'algorithme de ligne de partage des eaux, initialement conçu pour détecter des objets dans des images, en tant que méthode de réequilibrage pour les ensembles de données numériques déséquilibrés. L'algorithme de ligne de

partage des eaux, bien qu'initialement conçu pour le traitement d'images, s'est révélé prometteur lorsqu'il a été directement appliqué à des ensembles de données présentant des déséquilibres. Cette transition a été facilitée par la nature adaptable de son algorithme basé sur la construction de graphes. Cette adaptation a conduit à la création de WSSMOTE, une méthode de suréchantillonnage basée sur l'algorithme de partage des eaux. Les résultats obtenus avec WSSMOTE sur l'ensemble de données PRESEV se sont avérés prometteurs. En effet, WSSMOTE a permis de réduire la variabilité de 13,3% à 1,2%, tout en maintenant des scores de VPP et de VPN stables et bons. De plus, WSSMOTE a également démontré sa capacité à améliorer les prédictions sur d'autres ensembles de données déséquilibrés provenant du monde réel. Cependant, il est crucial de souligner que, tout comme les autres méthodes de suréchantillonnage, WSSMOTE n'offre pas une solution universelle, car elle n'a pas conduit à une amélioration des prédictions pour tous les ensembles de données étudiés.

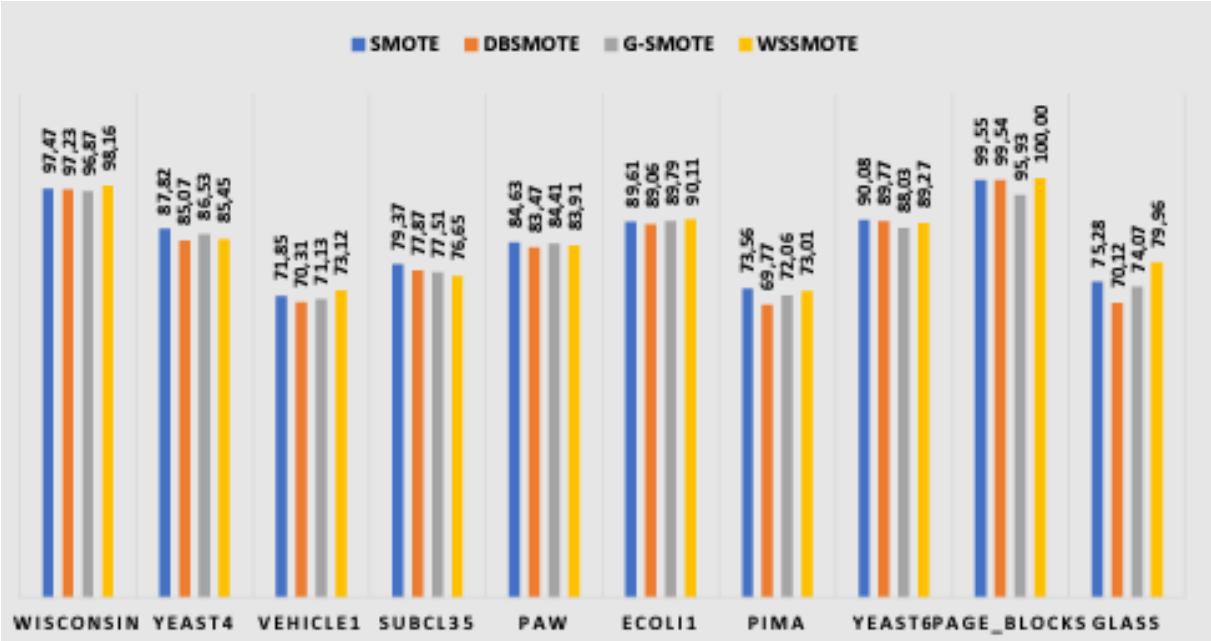


Figure 2: Analyse comparative des scores de moyenne géométrique obtenus sur diverses bases de données provenant de cas concrets. On observe que WSSMOTE contribue à l'amélioration des prédictions dans certains scénarios, mais cette amélioration n'est pas généralisable.

Dans ce contexte et suite au succès que nous avons rencontré avec notre approche WSSMOTE, nous avons entrepris une exploration dans le domaine des méthodes d'apprentissage de métrique, en particulier celles qui reposent sur des problèmes d'optimisation. En utilisant la théorie des graphes ainsi qu'un algorithme de descente de gradient, ces approches basées sur la transformation des poids des arêtes d'un graphe et la position des points de données ont permis d'optimiser la projection des données. Cette optimisation a pour but de créer des regroupements plus cohérents,

facilitant ainsi l'application de méthodes de suréchantillonnage et de classification. Bien que les méthodes WML et WMLI que nous avons mises en œuvre aient montré leur efficacité sur divers ensembles de données déséquilibrées, les résultats sur l'ensemble PRESEV n'ont pas enregistré d'améliorations.

En prenant conscience des limites des méthodes existantes, qui se concentrent uniquement sur la disposition relative des points entre eux, notre attention s'est tournée vers une modification directe de la hiérarchie de l'algorithme de ligne de partage des eaux. Bien que plusieurs approches aient été testées, comme la modification de la hiérarchie via l'élévation ou la propagation de marqueurs, aucune n'a abouti à des résultats convaincants. Ceci pourrait être attribué à la simplicité de la méthodologie. Cependant, l'idée de modifier directement l'algorithme de ligne de partage des eaux et la combinaison de méthode d'imagerie dans le cadre de problemes de classification numérique de données déséquilibrées demeure intéressante pour de futures explorations.

Introduction & Outline

Sickle cell disease is a rare, chronic and potentially fatal hereditary disease affecting red blood cells. It was first identified in the early 20th century by James Herrick [10], a Chicago physician. Today, it is the most prevalent monogenic disease in the world, and has been recognized as a public health priority by several organizations such as the World Health Organization, the United Nations and UNESCO. Consequently, developing a global strategy to understand this disease is of paramount importance. Indeed, understanding the sickle cell disease and its complications will improve the diagnosis and the treatment of affected patients. Under specific conditions, the hemoglobin of SCD patients presents a unique polymerization behavior within red blood cells, resulting in their typical sickle-shaped appearance. These altered cells are difficult to deform and can obstruct blood capillaries [6]. This leads to vaso-occlusive crises (VOC), extremely painful events that mainly affect the bones. These crises account for a significant proportion of emergency room visits and subsequent hospitalizations [1]. Acute chest syndrome (ACS) appears to be the most feared complication and the leading cause of mortality in patients hospitalized with VOC [7, 5, 8]. Usually occurring around 2.5 days after a VOC crisis, ACS affects around 17% of patients hospitalized for VOC [12].

In order to effectively identify ACS and administer appropriate treatment to patients, a dataset consisting of patient biomarkers has been generated. This dataset has been then analyzed using conventional statistical and machine learning techniques. However, the results obtained are subject to significant over-fitting and variability. These issues occur due to the particular characteristics of this dataset. In fact, the dataset in question is composed of two distinct groups: the first includes a majority of patients who do not develop ACS, while the second represents a minority of patients who do develop ACS. Consequently, when traditional machine learning methods are employed to predict the occurrence of ACS, issues of over-fitting and variability stand out.

This study is a typical example of an imbalanced classification problem. One can define a majority class, here patients without ACS, against a minority class, here patients with ACS. More theorically, a binary dataset is considered imbalanced if there is a significant disproportion between the

minority class and the majority one. The concept of imbalanced datasets is of major importance, as conventional machine learning methods often give poor results and introduce biases when confronted with imbalanced datasets. This can be attributed to the fact that machine learning algorithms, such as support vector machines or nearest neighbors, are generally designed for classification problems involving balanced datasets. In this case, both classes are uniformly represented and, consequently, treated with equal importance. Specific techniques have been developed to improve prediction accuracy and reduce variabilities associated with imbalanced classification problems. These methods, called imbalanced methods, have been widely discussed and described in various papers, such as [9, 2, 4]. In brief, the imbalanced methods can be categorized into three distinct groups:

- **Cost-Sensitive Learning**: This category is usually used for tasks like feature selection. In the context of imbalanced datasets, it serves to accentuate the minority class. This can be accomplished by assigning weights to the minority data either during the preprocessing part or directly at the algorithmic level.
- **Data Level / Sampling Methods**: Data sampling constitutes a preprocessing step. It involves either generating new data points from the minority class or reducing data points from the majority class. This category represents the most widely adopted methods to deal with imbalanced dataset.
- **Algorithmic Methods**: Algorithmic methods maintain the integrity of the original dataset while modifying the behavior of the classifier. This approach ensures that the data distribution is preserved, even though it necessitates a comprehensive understanding of the classifier mechanics. A range of traditional algorithms, such as support vector machines, decision trees, and nearest neighbors, have been adapted to address this aspect.

Despite recent developments in the imbalanced field, two problems remain unresolved. The first difficulty has been discussed in depth and is illustrated by articles such as [3]. The main idea is that no existing method for dealing with imbalanced data appears to be a universal silver bullet. In other words, each set of imbalanced data requires a specific approach, and this approach cannot be determined purely on the basis of factors such as the number of features, samples... The second problem concerns the lack of a methodological framework or pipeline to guide users in choosing an appropriate approach for dealing with imbalanced datasets. Essentially, there is a lack of search grids or methodologies to systematically compare

several imbalance methods. The study of ACS in patients with sickle cell disease is a pertinent example of both challenges. Existing imbalanced methods are proving inadequate for the dataset in question, leading to poor results. Furthermore, the absence of a standardized pipeline makes systematic exploration of alternative methods extremely difficult. These issues underline the complexity of dealing with imbalanced datasets, and emphasize the need for appropriate approaches and a systematic framework for imbalanced method comparison and selection.

To address the last challenge, we have designed an open-source pipeline that facilitates the comparison of various imbalanced methods and identifies the optimal one. Additionally, to find a suitable imbalanced method for the classification problem associated with the detection of ACS for SCD patients, we have forged a connection between digital data and image fields. Specifically, we have recognized the parallels between the task of object detection in computer vision and the imbalanced problem. In object detection, a small subset of pixels represents the object, constituting the minority class, while the pixels depicting the background form the majority class. This perspective has led us to explore the potential of computer vision techniques and more specifically of mathematical morphology (MM). MM is a field developed by G. Matheron and J. Serra in 1964 [11]. It is a non-linear approach used for image analysis, incorporating geometric principles, topology, lattice theory, and mathematical functions. Within the domain of mathematical morphology, a specific technique known as watersheds holds an important place. Watersheds are employed for object detection in images and also find applications in image segmentation. Over time, the capabilities of watershed methods have expanded beyond image analysis. This evolution has led to the development of algorithms and connections that extend the utility of watersheds from image analysis to the processing of digital data.

The thesis is structured into seven chapters contributing to the development of an open-source pipeline, investigating imbalanced datasets, exploring the watershed concept, and designing three algorithms centered around watershed techniques to enhance the detection of ACS. The following structure is employed for the next chapters. Chapter 1 explores the task of identifying acute chest syndrome (ACS) in patients with sickle cell disease, chapter 2 discusses the challenges of processing imbalanced data for ACS detection, chapter 3 presents the development of the ImbPip pipeline for selecting appropriate imbalanced methods, chapter 4 examines the use of image detection techniques, in particular the watershed method, to improve ACS detection, chapter 5 presents the new WSSMOTE oversampling method and the successful results obtained in detecting ACS,

chapter 6 looks at metric learning theory and chapter 7 presents ultrametric optimization algorithms, while the appendices describe some exploration of algorithms using watershed hierarchy methodology for imbalanced data.

Chapter 1 **Acute Chest Syndrome Detection in Sickle Cell Patients: Exploring Literature and Challenges**

In this chapter, we focus on the challenging task of identifying ACS in people with sickle cell disease (SCD). The first steps are to investigate the nature of sickle cell disease itself, unravel its intricacies and understand the mechanisms underlying the emergence of ACS. Next, we will set off on a statistical exploration of the Predictive Severity Study (PRESEV) dataset. This dataset comprises biomarkers data from a wide-ranging survey of patients with SCD, focusing on those admitted to the emergency department. Our analysis method includes a detailed examination of previous results obtained using standard machine learning methods. Results reflect over-fitting and variability due to the imbalanced criteria of the dataset.

Chapter 2 **Imbalanced Classification Challenge: A Review of the Literature**

Given that ACS detection illustrates a classic imbalanced classification problem, our goal is to understand the multi-faceted challenges involved in working with imbalanced data. In parallel, an understanding of the relevant metrics to be employed becomes essential. Our exploration extends to a retrospective analysis of the methods designed over time to address imbalanced data challenges. In this context, one point stands out: the absence of a framework for comparing different imbalance methods. This absence increases the complexity of finding the best imbalance method to use for ACS detection.

Chapter 3 **ImbPip: A Pipeline for Comparing Imbalanced Methods**

With the aim of identifying the most appropriate imbalanced method to tackle the ACS detection classification challenge, we have designed, developed, and implemented a robust pipeline. This pipeline is designed to adapt to any imbalanced dataset and any metrics, finally delivering the optimal imbalanced method along with its associated parameters and results. Validation of our pipeline was achieved through testing on real-world imbalanced datasets. Subsequently, we will apply this pipeline to the PRESEV dataset and observe that none of the imbalanced methods tested improve ACS detection in sickle cell patients. It should be noted that the initial version of the pipeline focuses only on sampling methods, which are the most commonly used

by scientists to address real-world challenges. Other methods will be incorporated into the pipeline later, in Chapter 6.

Chapter 4 **Bridging Image Detection and Imbalanced Classification: Exploring the Watershed Theory**

As existing imbalanced methods do not improve ACS detection, we study the interaction between image detection and imbalanced dataset predictions. Over time, numerous methods have emerged for detecting small details or objects in images, as well as for refining object segmentation. This bridges the gap between object detection and imbalanced datasets, as the challenges as both share similar challenges. The watershed method is one of the main techniques for image segmentation and object detection. Watersheds are part of the theory of mathematical morphology and are based on the principle of hydrology when basins and hills are filled with water. The watershed has been developed for image segmentation, particularly in fields such as medical imaging and computer vision. In addition, we discuss its use in graph theory and data analysis for tasks such as clustering and classification. The final section of this chapter includes a comparative analysis of the watershed method and other semi-supervised approaches on imbalanced real-world datasets. While watersheds as semi-supervised methods can improve some predictions, the watershed method performs best when used as an unsupervised technique, as a clustering method.

Chapter 5 **WSSMOTE: a Novel Oversampling Method**

Our study shows that watershed-based clustering methods perform well compared to traditional clustering techniques when dealing with imbalanced data. The potential benefit of these methods as a preprocessing step to balance our dataset and subsequently improve predictive results becomes obvious. So, we designed a new oversampling approach, called Watershed-based Synthetic Minority Over-sampling Technique (WSSMOTE). To validate its effectiveness, we carried out tests on various randomly selected imbalanced datasets. Encouragingly, we observed significant improvements, particularly in the negative and positive predictive values. More importantly, these improvements were reflected in our analysis of the PRESEV dataset, further highlighting the potential of the WSSMOTE approach to improve ACS detection outcomes in sickle cell patients.

Chapter 6 **Exploring Metric Learning Theory & Application to Imbalanced Datasets**

In some cases, the complex nature of imbalanced data prevents usual supervised machine learning algorithms and oversampling methods from working to their full potential. In response to this problem, metric learning methods offer a possible solution. These methods aim to reshape the feature space, improving the separation of data points based on their respective labels, ultimately facilitating more efficient processing. With this in mind, we conducted a thorough review of the existing literature on different metric learning methods, exploring their applications in balanced and imbalanced cases. To extend our understanding, we adapted the method comparison pipeline, allowing us to directly compare these metric learning approaches with other existing oversampling methods.

Chapter 7 **WML: an Ultrametric Learning Method for Imbalanced Datasets**

Driven by the success of WSSMOTE and the positive impact of metric learning, we have developed two ultrametric-based optimization algorithms, called WML and WMLI. These algorithms take advantage of the properties of graphs to improve results without the need for synthetic data points. We show that while both methods improve predictions in some cases, such as oversampling methods, there is no perfect data-level preprocessing method.

Appendices **Watershed Hierarchy & Marker Propagation for Imbalanced Data**

Both optimization algorithms avoid creating artificial data points. However, they are still pre-processing techniques based on the intrinsic structure of the data. Like other pre-processing methods, they also face limitations in terms of reproducibility across all datasets. With this in mind, our research led us to design an algorithm rooted directly in watershed methodology, designed to meet the challenge of imbalanced data. To this end, we focused on the hierarchical aspect of watersheds and more precisely on marker propagation in the hierarchy. Note that this chapter is an opening.

References

[1] S.K. Ballas and M. Lusardi. “Hospital readmission for adult acute sickle cell painful episodes: frequency, etiology and prognostic significance”. In: *American Journal of Hematology* 79 (2005).

[2] A. Fernández et al. *Learning from Imbalanced Data Sets*. Springer Cham, 2018.

[3] A. B. Hassanat et al. “Stop Oversampling for Class Imbalance Learning: A Critical Review”. In: *Digital Object Identifier* (2022).

[4] J. M. Johnson and T. M. Khoshgoftaar. "Survey on deep learning with class imbalance". In: *Journal of Big Data* 6 (2019).

[5] B. Maitre et al. "Acute chest syndrome in adults with sickle cell disease". In: *Chest* 117 (2000).

[6] N. Murray and A. May. "Painful crises in sickle cell disease-patients' perspectives". In: *British Medical Journal* 297 (1988).

[7] V. Perronne et al. "Patterns of mortality in sickle cell disease in adults in France and England". In: *Hematol J.* 3 (2002).

[8] O.S. Platt et al. "Mortality in sickle cell disease. Life expectancy and risk factors for early death". In: *New England Journal of Medicine* 330 (1994).

[9] B. Santos et al. "Synthetic Over Sampling Methods for Handling Class Imbalanced Problems: A review". In: *IOP Conference Series: Earth and Environmental Science* 58 (2017).

[10] T. L. Savitt and M. F. Goldberg. "Herrick's 1910 case report of sickle cell anemia. The rest of the story". In: *JAMA* (1989).

[11] J. Serra. *Image Analysis and Mathematical Morphology*. 1982.

[12] E.P. Vichinsky et al. "Causes and outcomes of the acute chest syndrome on sickle cell disease. National Acute Chess Syndrome Study Group". In: 342 (2000).

Chapter 1

Acute Chest Syndrome Detection in Sickle Cell Patients: Exploring Literature and Challenges

Sickle cell disease is a serious genetic disorder characterized by an average life expectancy of around 40 years. Its prevalence is increasing worldwide, with particularly high rates in African countries, making it an urgent public health problem. The study of the effects and underlying causes of this disease is now of crucial importance. Numerous prospective studies have been conducted, including two on a particularly fatal syndrome associated with the disease called Acute Chest Syndrome (ACS). This chapter will give an overview of this syndrome's consequences. It will also outline an analysis of two related datasets, present initial results and explained the next challenges.

1.1 Sickle Cell Disease & Acute Chest Syndrome (ACS)

Sickle cell disease (SCD) stands as the most prevalent monogenetic disorder. Diagnosis of this disease necessitates patients to be homozygous, denoting that both alleles of the specific gene must carry mutations. This implies that individuals with sickle cell disease inherit the genetic alteration from both parents. In other terms, possessing a single mutated allele does not lead to sickle cell disease; but instead, it provides an advantage in combatting malaria (cf. article [10]). Due to the primary function of the affected gene, regions where malaria is endemic, like sub-Saharan Africa, the Middle East, and the Mediterranean, are more susceptible to sickle cell disease prevalence. However, historical factors such as the slave trade and contemporary population migrations have diffused the disease across the globe. This geographical distribution of SCD prevalence is illustrated in Figure 1.1. Consequently, the count of sickle cell patients is rising, posing a significant public health challenge for future generations. Presently, SCD accounts for over 300,000 annual births, with this number expected to rise.

In order to manage this problem effectively, we first need to understand its biological complexities.

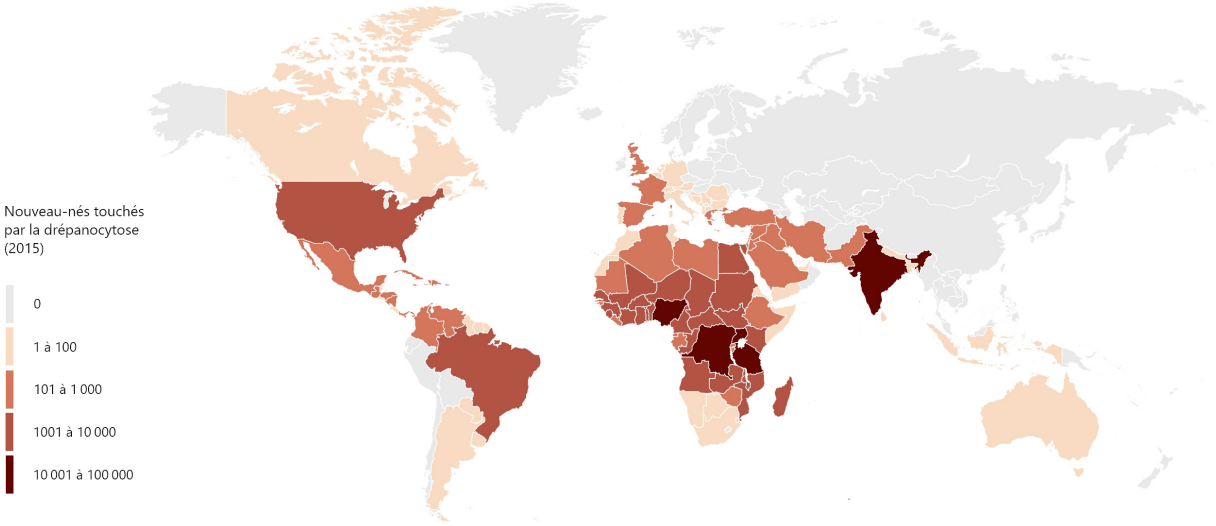


Figure 1.1: Geographical distribution of patients with sickle cell disease. Figure extracted from the French sickle cell disease prevention website. The legend indicates the number of newborns with sickle cell disease in 2015.

SCD is a genetic disorder caused by a mutation in the gene located on chromosome 11 that encodes for hemoglobin [14]. This genetic mutation results in an abnormal variant of beta-globin, a protein expressed by red blood cells (RBCs), particularly by reticulocytes. Reticulocytes are immature RBCs that eventually mature into erythrocytes, the oxygen-carrying cells in the bloodstream. However, due to the mutation, erythrocytes assume an abnormal form. Under conditions of low oxygen levels, these abnormal RBCs change shape and exhibit altered physical characteristics. They transition from their typical flexible biconcave structure to a rigid elongated one. This distortion is initially temporary and reversible, but with repeated occurrences, it becomes persistent. In addition, despite SCD originating from a single genetic mutation, the consequences stemming from these altered RBCs are diverse and variable [8]. One of the consequences is that the lifespan of the abnormal RBCs is considerably shortened, leading to a shortage of RBCs in the bloodstream and eventually causing anemia. A second one is that due to their sickle shape, these RBCs can obstruct small blood vessels, initiating vaso-occlusion which results in intense pain and even organ dysfunction. Additional complications may arise, encompassing inflammation, increased blood coagulation, heightened platelet activation, and more. The schematic in Diagram 1.2 succinctly outlines the interactions between genetic mutations and the resulting phenotypic alterations.

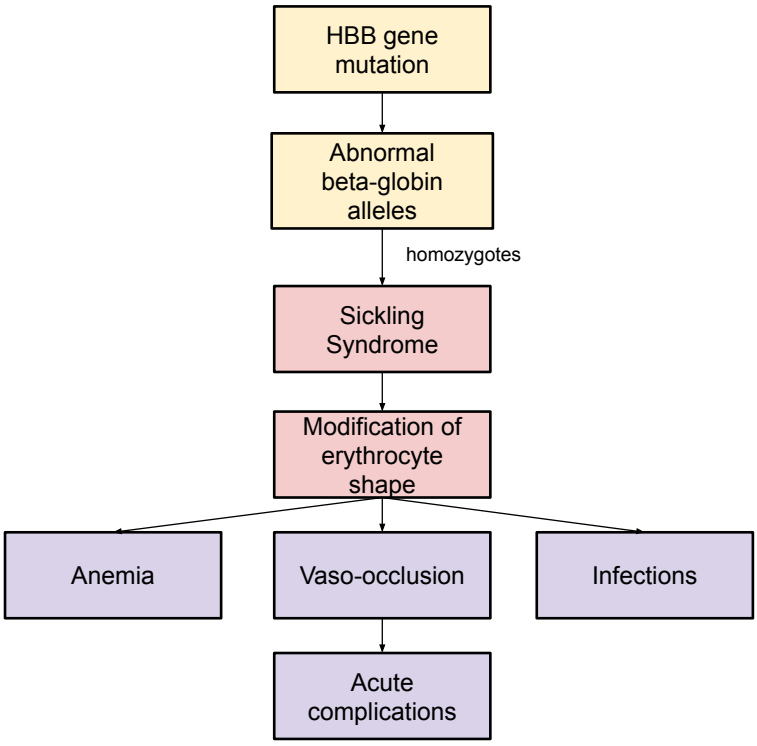


Figure 1.2: Summary of sickle cell disease. The yellow boxes correspond to the genetic steps, the red to the phenotypic ones and the purple to the consequences.

The interactions among these complications are not yet fully understood. As detailed in the article [12], individuals with SCD might display elevated levels of C-reactive protein (CRP) along with various other markers of inflammation. Consequently, diagnosing their condition can be exceptionally challenging. However, advances have been made through studies of sickle cell disease populations. In particular, since the inaugural Jamaican sickle-cell cohort study in 1973, progress has been made. This is demonstrated by the increase in life expectancy, as shown in the article [9]. Life expectancy has risen from childhood to an estimated median survival of 45-48 years in Western countries.

Nevertheless, while interventions like oxygenation and transfusion can offer assistance, the sole substantial treatment is hydroxyurea. Unfortunately, hydroxyurea is costly and is only limited to privileged individuals in a small range of countries. In addition, pressure on public health services further complicates the provision of care for patients with sickle cell disease. As highlighted in the article [10], a way to improve management and achieve more precise treatments lies in improved prediction of SCD patients and the factors underlying their mortality. For instance, a predominant reason for hospitalization among SCD patients is vaso-occlusive crisis (VOC), while acute chest syndrome (ACS) emerges as the primary cause of death during these hospital stays. A comprehensive study of over 3,700 patients, explained in article [9], looked at mortality rates in

ACS patients following a VOC crisis. This study found that over 33% of patients succumbed to ACS-related complications. The considerable proportion of ACS-related deaths is explained in article [1], by the fact that ACS typically manifests around 2.5 days after admission for VOC. Thus, because of premature discharge, patients often return for re-admission at the peak of the ACS episode, reducing the efficacy of treatment. The study describes in article [13] points out that if patients could be identified earlier and receive prompt treatment involving transfusion, bronchodilators, and aggressive interventions, their chances of recovery would significantly improve. Therefore, by predicting the likelihood that a patient will experience an episode of ACS, the total number of ACS-related deaths could potentially be reduced. It is in this context that the Predictive Severity Study (PRESEV) was conceived.

1.2 Predictive Severity Study (PRESEV)

In the article by P. Bartolucci et al. [2], a comprehensive review was conducted to analyze the distinctions between patients with vaso-occlusive crisis (VOC) who are likely to develop acute chest syndrome (ACS) and those who are not. This survey, known as the Predictive Severity Study (PRESEV), has two main objectives. The initial goal is to identify biomarkers that play a role in the progression of ACS. The study aims to identify specific biological indicators involved in the development of acute chest syndrome. The secondary objective revolves around the construction of a predictive scoring system. This scoring mechanism is designed to assess the risk of ACS occurrence when a patient is admitted to the hospital with a vaso-occlusive crisis. The ultimate aim is to create a tool capable of anticipating the occurrence of ACS on the basis of certain predetermined criteria.

1.2.1 Study Presentation

The PRESEV dataset consists of two distinct prospective phases. The initial phase, called PRESEV1 [2], was conducted at Henri Mondor Hospital in Créteil, France. On the other hand, called PRESEV2 [7], is an international prospective study that includes patients from both African and European regions. Both phases of the study exclusively involve adult patients who are afflicted by severe vaso-occlusive crisis (VOC). Severity in this context is defined as the presence of pain or sensibility affecting at least one area of the body, which is not adequately controlled by grade II analgesics and necessitates the use of opioids. The main objective of the PRESEV study is to investigate and understand the correlation of biomark-

ers on the development of acute chest syndrome (ACS). By examining these biomarkers, the study aims to establish a predictive model for the occurrence of ACS. This predictive model is intended to classify patients into distinct risk categories: low, moderate and high. Patients considered to be at low risk could benefit from a short hospital stay or even outpatient follow-up. On the other hand, patients identified as high risk could be referred to specialized centers where intensified ACS prevention protocols are available. As a result, prediction of ACS could potentially lead to reduced unecessary hospital stays, lower mortality rates and improved patient care and support through more targeted interventions.

1.2.2 Data Exploration

A thorough understanding of the dataset is a crucial initial step before proceeding with modeling or analysis. A good understanding of the dataset enables you to identify patterns, characteristics and potential challenges associated with the data.

- **PRESEV1 Dataset**
  - This dataset is composed a total of 244 patients.
  - Among these, 203 patients did not experience an episode of acute chest syndrome (ACS), whereas 41 patients developed an ACS episode. Hence, approximately 16.8% of the patients within the PRESEV1 study group developed an ACS.
  - The data collection for this study was conducted at Henri Mondor Hospital in Créteil, France.
  - The study investigates a range of biomarkers, including: Oxygen saturation (SatOx), Respiratory frequency (RF), Systolic blood pressure (Syst), Diastolic blood pressure (Diast), Temperature, Hemoglobin levels (Hem), Reticulocyte count (Ret), Leukocyte count (Leuc), Platelet count (Platelets), Alanine transaminase (ALAT), Aspartate transaminase (ASAT), Lactate dehydrogenase (LDH), Bilirubin concentration (BiliC), Total bilirubin concentration (BiliT), C-reactive protein (CRP), Urea levels (Uree), Pain etc.
- **PRESEV2 Dataset**
  - This dataset is composed a total of 393 patients.
  - Among these, 317 patients did not experience an episode of acute chest syndrome (ACS), whereas 76 patients developed an ACS episode. Hence, approximately 17.7% of the patients within the PRESEV1 study group developed an ACS.

- The data collection for this study was conducted in multiple centers, including Mali, Henri Mondor Hospital, Tenon Hospital, Louis Mourrier Hospital, Avicenne Hospital, Rouen Hospital, Georges-Pompidou Hospital, Togo, Belgium, Versailles Hospital, Saint Denis Hospital, and England.
- The study investigates a range of biomarkers, including: Oxygen saturation (SatOx), Respiratory frequency (RF), Systolic blood pressure (Syst), Diastolic blood pressure (Diast), Temperature, Hemoglobin levels (Hem), Reticulocyte count (Ret), Leukocyte count (Leuc), Platelet count (Platelets), Alanine transaminase (ALAT), Aspartate transaminase (ASAT), Lactate dehydrogenase (LDH), Bilirubin concentration (BiliC), Total bilirubin concentration (BiliT), C-reactive protein (CRP), Urea levels (Uree) etc.

The number of patients recruited are summarized in Table 1.1. The categorical pain score (CPS) is the only categorical feature. It is defined using Figure 1.3.

	Number of patients	**Number of patients without secondary ACS**	**Number of patients with secondary ACS**
PRESEV1	244	203	41
PRESEV2	393	317	76

Table 1.1: Number of patients and features for PRESEV1 and PRESEV2 studies

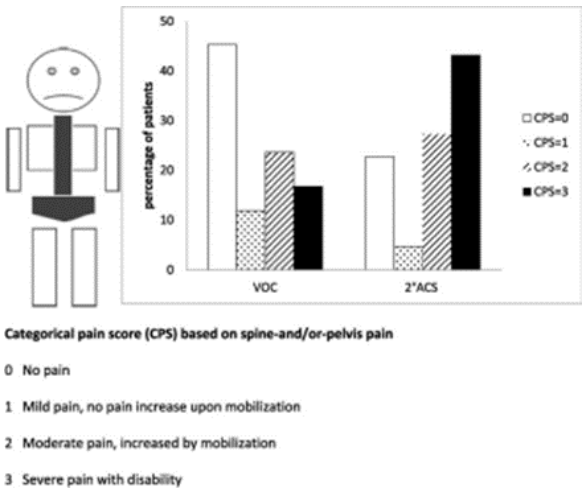


Figure 1.3: Definition of categorical pain score (CPS), figure from article [2].

By analyzing the information in the table 1.1, it becomes clear that there is a notable disparity between the number of patients who have developed ACS and those who have not. On average, only about 20 out of 100 patients experienced an episode of ACS. As a result, the PRESEV datasets show an imbalanced distribution. In addition, an important observation is that the biomarkers studied are not consistent across the two datasets and

are quite numerous. A preliminary analysis exploring statistical distinctions between patients who have developed ACS and those who have not is therefore required. This exploration aims to identify biomarkers that have a significant correlation with the development of ACS. In addition, it is crucial to recognize that the LDH requires normalization due to its dependence on the referral center. Prior to any further analysis, the dataset was cleaned and the LDH feature normalized to ensure data consistency and validity.

1.2.3 Statistical Analysis

After cleaning and normalizing the dataset, a statistical analysis was performed. Given the imbalanced nature of the PRESEV dataset, the direct use of complex methods such as principal component analysis (PCA) may be problematic. Consequently, histograms and boxplots were employed to visualize and compare the distributions between patients who developed secondary ACS and those who did not. The outcomes of this analysis are illustrated in figures 1.4,1.5, 1.6, and 1.7. To generate these histograms, we use the Struges rules to define the number of classes. We denote ACS as the population of patients who developed secondary ACS and no-ACS as the patients without ACS. At first sight, we notice a difference in distribution between ACS and no-ACS patients on the following features:

- **For the PRESEV1 study**: ALAT, ASAT, Bili_C, Bili_T, CPS, CRP, LDH, Leuc, Ret and Urea.
- **For the PRESEV2 study**: ASAT, CPS, LDH, FR and Ret.

While data visualization is a first step, it is not a definitive method to draw conclusions. To improve the reliability of the interpretation, we have supplemented our statistical analysis with hypothesis-testing methods, namely the Wilcoxon-Mann-Whitney test [4] and Student's t-test. These two tests compare two independent populations and are intended to provide a more rigorous examination of potential differences. In the Wilcoxon-Mann-Whitney test, the null hypothesis (H0) asserts that the two distributions being compared are equal. Conversely, in Student's t-test, the null hypothesis (H0) claims that the two means being compared are equal, assuming a normal distribution. To make sense of these tests, the p-values obtained have been compiled in: table 1.2 for the PRESEV1 study and table 1.3 for the PRESEV2 study. A p-value of less than 0.05 means that the null hypothesis can be rejected. This conclusion implies that there are statistically significant differences in the distributions or means of characteristics between ACS and non-ACS patients. This rig-

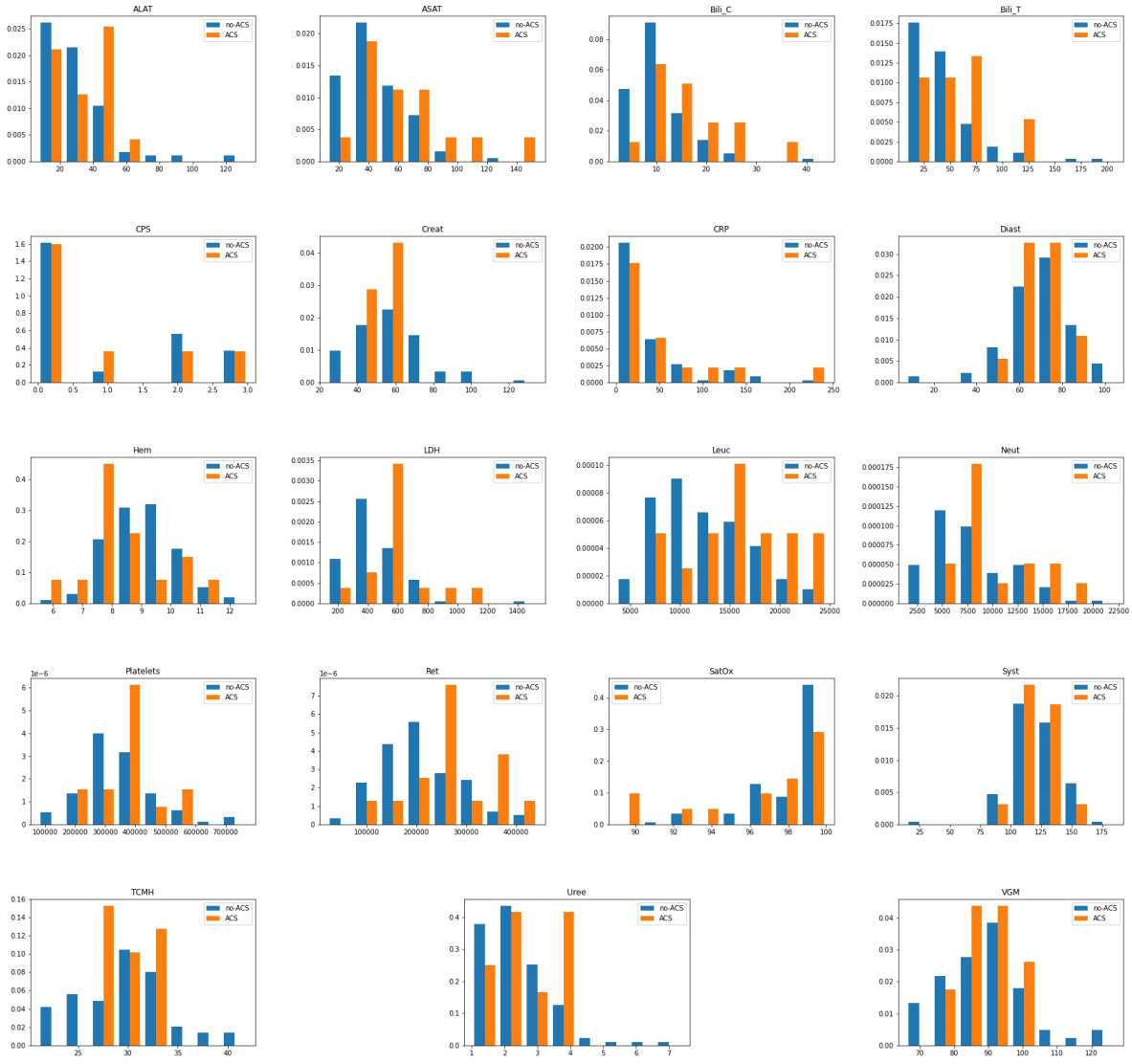


Figure 1.4: Histograms obtained using PRESEV1 dataset.

orous approach allows us to understand more concretely the role of these features in distinguishing between the two groups of patients.

Feature	SatOx	FR	Syst	Diast	Hem	VGM	TCMH	Ret	Leuc	Neut	Platelets	ALAT	ASAT	LDH	Bili_T	Bili_C	CRP	Urea	Creat	CPS
p_value Wilcoxon-Mann-Whitney test	0.142	**0.05**	0.89	0.70	**0.02**	0.80	0.63	**0.005**	**0.01**	**0.04**	0.55	0.21	**0.03**	**0.004**	0.08	**0.009**	0.57	0.10	0.41	0.89
p_value Student test	**0.02**	0.20	0.80	0.60	**0.03**	0.99	0.68	**0.003**	**0.008**	**0.07**	0.66	0.53	**0.003**	**0.005**	0.17	**0.003**	0.38	0.19	0.35	0.80

Table 1.2: P_values obtained for PRESEV1 Dataset.

Feature	Hem	Ret	Leuc	CPS	Syst	Diast	SatOx	FR	Platelets	LDH	Urea	Creat	ASAT	ALAT	BiliT	BiliC	CRP
p_value Wilcoxon-Mann-Whitney test	0.25	0.09	**0.005**	**0.0004**	0.52	0.49	0.14	0.16	0.13	**0.003**	**0.003**	0.43	**0.03**	0.46	0.75	0.29	**0.01**
p_value Student	0.26	0.08	0.48	**0.007**	0.82	0.52	0.18	0.12	0.30	**0.02**	**0.02**	0.68	0.38	0.76	0.44	0.48	**0.003**

Table 1.3: P_values obtained for PRESEV2 Dataset.

The generated p-values indeed emphasize the significance of various biomarkers in distinguishing between ACS and no-ACS patients. In PRESEV1, markers like Ret, Leuc, Neut, ASAT, LDH, Bili_C, CPS, and SatOx

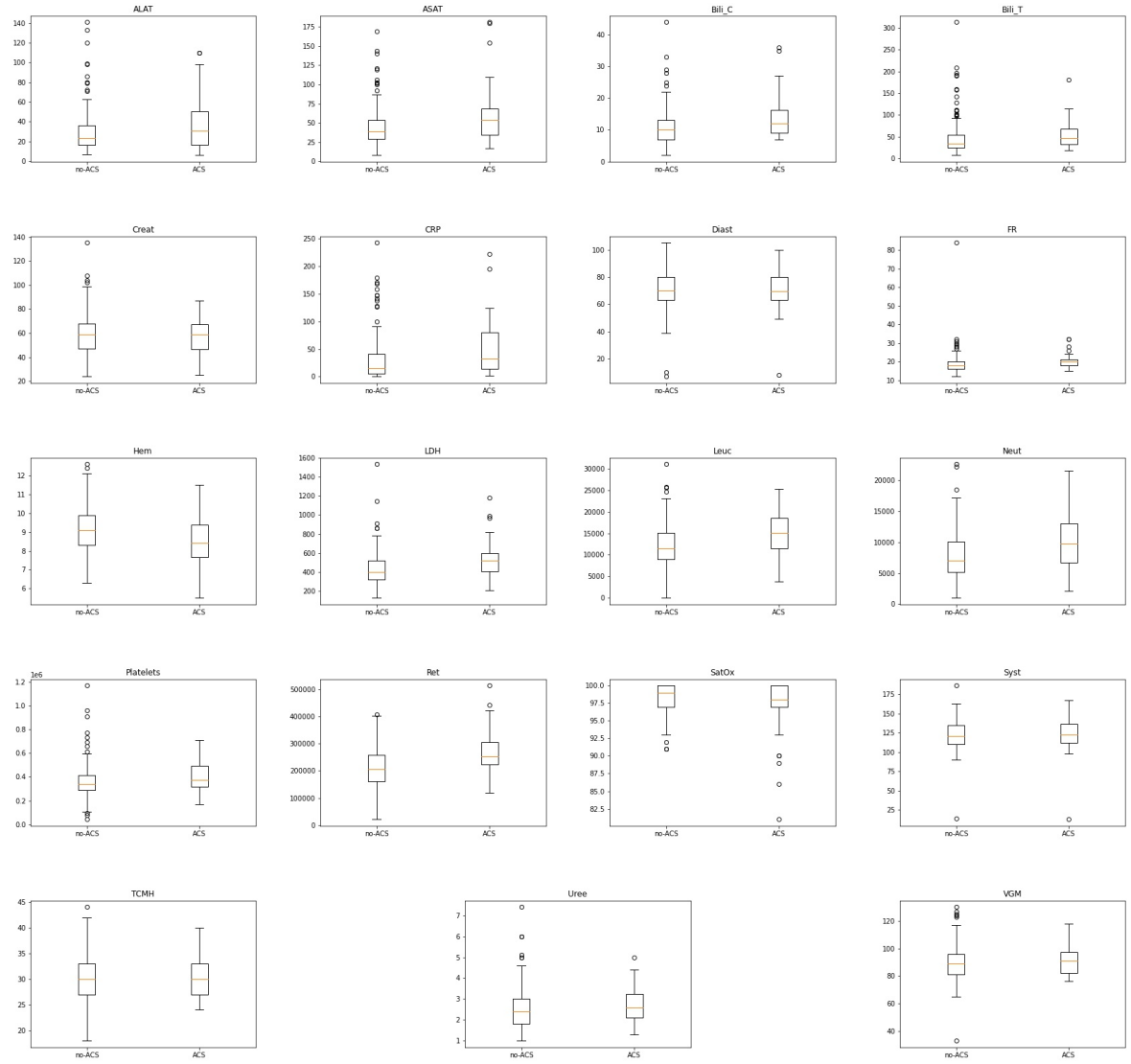


Figure 1.5: Boxplots obtained using PRESEV1 dataset.

hold notable importance. Similarly, in PRESEV2, Ret, Leuc, ASAT, LDH, Urea, CRP, and CPS stand out as influential markers. Combining the insights from both tables reveals that the following markers are potentially impactful in predicting the development of ACS: Leuc, Ret, CPS, LDH, CRP, Urea, and ASAT. In addition, since sickle cell disease originates from a genetic alteration of chromosome 11, which codes for hemoglobin, the hemoglobin (Hem) characteristic should also play an essential role.

With these key features identified, the next step is to build a predictive scoring system based on clinical and biological markers. This scoring score system aims to predict the probability of a patient developing ACS, thereby contributing to more accurate risk evaluation and patient management. The development of this predictive score could make a significant contribution to improving care and outcomes for patients with sickle cell disease.

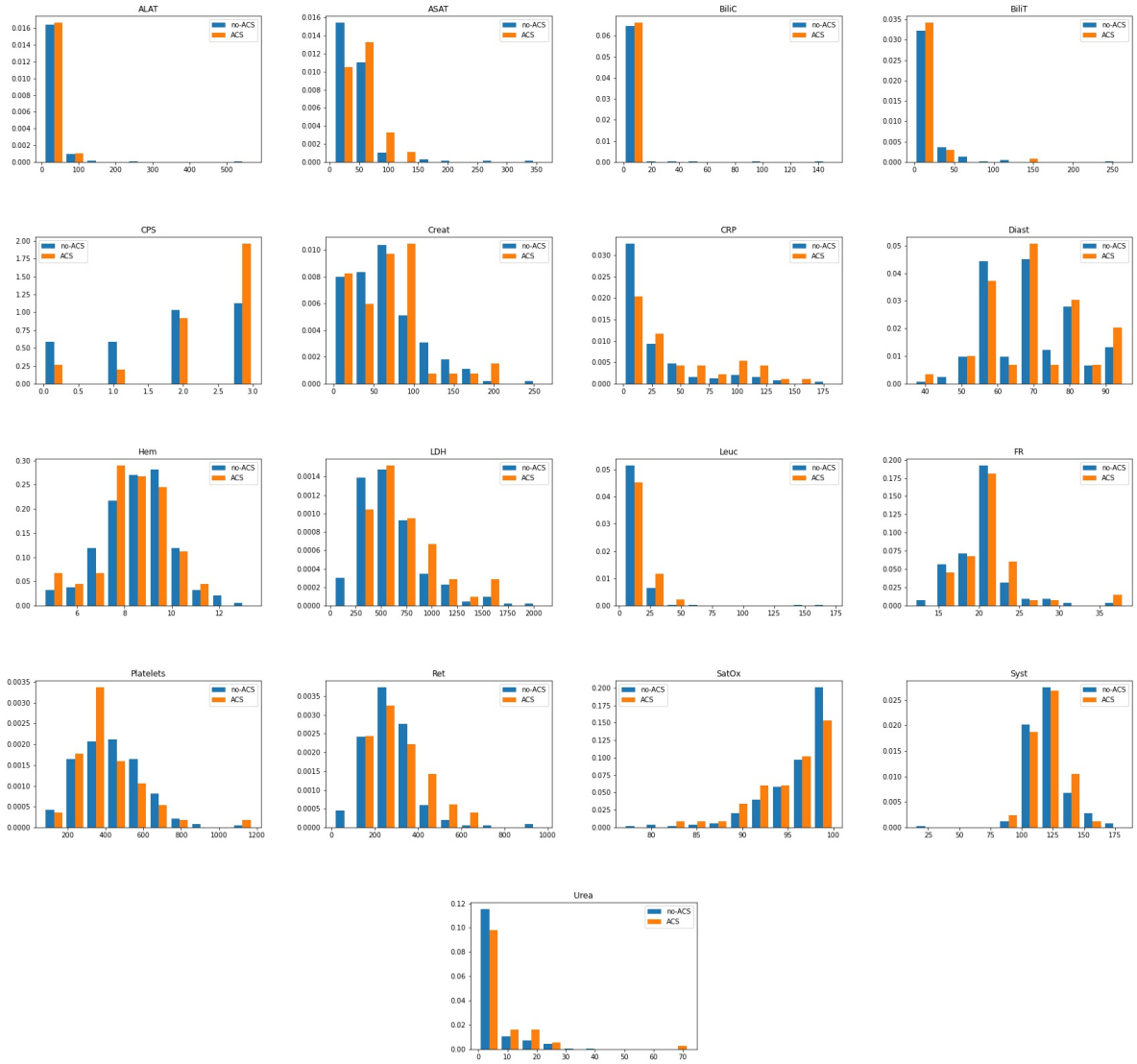


Figure 1.6: Histograms obtained using PRESEV2 dataset.

1.3 Detection of ACS: A Typical Imbalanced Classification Problem

As we discussed earlier, the main objective of the PRESEV study is to predict the likelihood of a low or high risk of developing ACS. By creating a scoring system, patients can be classified accordingly - those considered low risk can be remotely monitored or given a short hospital stay, while those classified as high risk can be referred to specialized centers where improved preventive ACS protocols are available. Thus, the key objective of this classification challenge is to minimize prediction errors for ACS patients while maximizing prediction for non-ACS patients. Essentially, the aim is to achieve the highest possible negative predictive value (NPV) and the highest positive predictive value (PPV). Given the crucial importance of identifying ACS cases, a strict standard has been set for NPV, with an error rate of 5% being the acceptable limit. In other words, a satisfactory

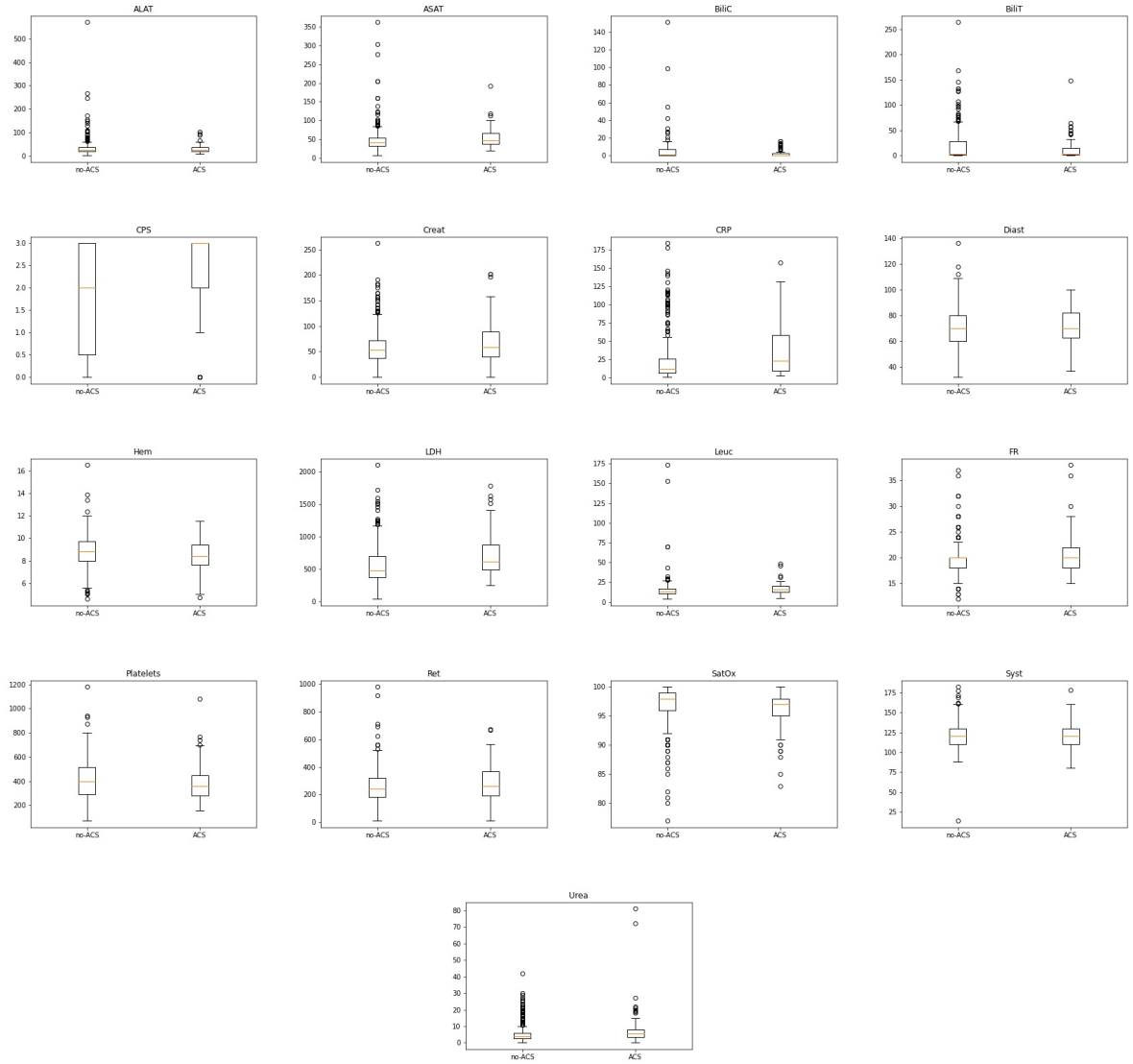


Figure 1.7: Histograms obtained using PRESEV2 dataset.

NPV is one that exceeds 95%. This highlights the importance of reducing false negatives to ensure effective identification and treatment of patients at risk of ACS.

1.3.1 Analyzing Previous Results

In previous studies PRESEV1 [2] and PRESEV2 [7], a score was generated using the PRESEV1 data as training set and PRESEV2 dataset as testing one. In addition, article [2] selected four bio-markers (CPS, Leuc, Ret and Hem) and a score method has been developed through a logistic regression method. This scoring method can be summarized as follows:

- RBC $\leq$ 216: 0 point
  RBC $>$ 216: 6 points
- CPS = 0 or 1: 0 point
  CPS = 2: 4 points

CPS = 3: 6 points

- Leukocytes $\leq$ 11: 0 point
  Leukocytes $>$ 11: 3 points
- Haemoglobin $>$ 9: 0 point
  Haemoglobin $\leq$ 9: 1 point

Consequently, the use of this scoring method developed in the articles [2] and [7] produced the resulting negative predictive value (NPV) and positive predictive value (PPV) results presented in table 1.4.

	PPV (%)	NPV (%)
PRESEV1	44,7	98,9
PRESEV2	27,6	95,8

Table 1.4: PPV and NPV obtained by the method developed in articles [2, 7]

The results of PRESEV2 serve to confirm the findings of PRESEV1. Nevertheless, there is significant variability in the positive predictive value (PPV) between PRESEV1 and PRESEV2, amounting to approximately 38%. This variance can be attributed to two main factors. Firstly, PRESEV1 constitutes a mono-center study, resulting in homogeneity within the dataset. In contrast, PRESEV2 is a multi-center study, leading to a more heterogeneous dataset. This difference in data origin contributes to the observed variability in the PPV. Additionally, during the statistical analysis phase, it became apparent that ACS occurrences were relatively infrequent, only happening around 20 times out of 100 instances. This imbalance in occurrence rates between ACS and non-ACS cases introduces an asymmetry in the number of samples within each class. The fact that our database is imbalanced is of major importance, as this can have a significant impact on the performance and reliability of predictive models. It is essential to address this imbalance and adopt appropriate strategies to ensure fair representation of both classes in order to obtain accurate predictions and meaningful information.

1.3.2 Transforming PRESEV into a Classification Problem

To introduce strategies for handling imbalanced datasets, the initial step is to transform the PRESEV study into a conventional classification problem. The initial step involves transitioning the PRESEV classification challenge into a machine learning one. To accomplish this, a series of actions must be undertaken. Given that the dataset has support cleaning procedures, the next action is to reduce the number of features. Dimensionality reduction techniques appear to be the optimal approach for this task. However, due

to the imbalanced nature of our PRESEV dataset, conventional techniques such as principal component analysis (PCA) or autoencoders may not yield significant results. These techniques tend to give priority to the majority class, often neglecting the minority class, resulting in biased feature importance. For this reason, a more time-intensive approach was adopted, involving the exhaustive testing of all feature combinations. In addition, it was decided to merge the PRESEV1 and PRESEV2 datasets, creating a unified dataset called PRESEVC. To ensure a comprehensive valuation, test and training parts were randomly generated. Importantly, particular attention was paid to the equal representation of minority and majority data points in both parts. This approach was employed to reduce the impact of class imbalance and promote a more equitable training and testing process.

In order to reproduce the results presented in the previous section, and in papers [2, 7], we executed standard machine learning methods. More specifically, we used methods such as random forest [3], adaboost [5], MLP [6], SVM [11] and logistic regression [15]. The evaluation process was conducted using a cross-validation methodology to ensure robust and unbiased results. By combining the NPV and PPV values obtained from various feature combinations, we obtain an aggregate table presented in table 1.5.

Features	**ML method**	**NPV**	**PPV**
Ret, Leuc, Hem, ASAT, LDH, Bili_C, CPS, Urea, CRP	AdaBoost	32.1 ± 30	84 ± 1.3
Ret, Leuc, Hem, ASAT, LDH, Bili_C, CPS, Urea	KNN	37 ± 29	84 ± 2
Ret, Leuc, Hem, ASAT, LDH, Bili_C, CPS, CRP	RF	29.23 ± 18.4	85.4 ± 4.1
Ret, Leuc, Hem, ASAT, LDH, Bili_C, Urea, CRP	RF	33.3 ± 29.7	83.5 ±2.2
Ret, Leuc, Hem, ASAT, LDH, CPS, Urea, CRP	AdaBoost	45.7 ± 30.9	84.5 ± 1.6
Ret, Leuc, Hem, ASAT, Bili_C, CPS, Urea, CRP	RF	37.3 ± 28.3	85.1 ± 2.6
Ret, Leuc, Hem, ASAT, Bili_C, CPS, Urea, CRP	RF	37.3 ± 28.3	85.1 ± 2.6
Ret, Leuc, Hem, LDH, Bili_C, CPS, Urea, CRP	AdaBoost	33.9 ± 29.3	84.4 ± 3.3
Ret, Leuc, Hem, ASAT, LDH, CPS, Urea	KNN	32.5 ± 28.9	83.3 ± 1.9
Ret, Leuc, Hem, ASAT, CPS, Urea, CRP	AdaBoost	44.6 ± 30.6	85.1 ± 1.5
...	...	...	...

Table 1.5: NPV and PPV values on the testing part obtained using different feature selections.

This table shows that the most effective feature combination is Hem, Ret, Leuc, LDH, Urea and CRP. However, the results obtained are not as outstanding as expected, which could be explained by the imbalanced nature of the PRESEV dataset. In response, a modification was introduced the code. The prediction outputs were transformed into a probabilistic format. Different thresholds were applied to adapt the predictions. This technique produced results that are presented in table 1.6. These results are close to those obtained in previous studies [2, 7]. Nevertheless, it

should be noted that the PPV value decreased and that a certain degree of variance in results was observed between the training and testing phases (over-fitting variance), as well as between repeated experiments (cross-validation variance). Both these variations could be due to a lack of data, or attributed to the fact that PRESEV is an imbalanced classification problem and not a classic classification one.

PRESEVC	PPV	NPV
Train	24.5 ± 3.9	94.1 ± 3.9
Test	26.1 ± 0.7	100 ± 0

Table 1.6: NPV and PPV value obtained using AdaBoost, probabilities output and feature selection (Hem, Ret, Leuc, LDH, Urea and CRP).

1.4 Conclusion

Sickle cell disease is of major concern, its prevalence being continuously on the rise. By predicting whether a patient suffering a vaso-occlusive crisis (VOC) will develop an acute chest syndrome (ACS), it becomes possible to reduce prolonged hospital stays and adapt treatments to individual patients, thereby reducing the total number of deaths. Initial predictions were made in papers [2] and [7], and then reproduced in this chapter using conventional machine learning techniques. However, these results presented a degree of variability, mainly due to the imbalanced nature of the dataset. There is an imbalance between the class of patients suffering from ACS and the class of patients who will not suffer from ACS. PRESEV therefore has the typical characteristics of an imbalanced dataset, requiring the application of imbalanced methods for accurate ACS detection. Thus, the next chapter will provide an overview of the challenges associated with solving an imbalanced classification problem. It will look at the complexities posed by imbalanced class distributions and explore candidate methods that can be exploited to tackle this problem.

References

[1] S. Ballas and M. Lusardi. “Hospital readmission for adult acute sickle cell painful episodes: frequency, etiology, and prognostic significance”. In: *American Journal of Hematology* 79 (2005).

[2] P. Bartolucci et al. “Score Predicting Acute Chest Syndrome During Vaso-occlusive Crises in Adult Sickle-cell Disease Patients”. In: *EBioMedicine* 10 (2016). URL: https://linkinghub.elsevier.com/retrieve/pii/S2352396416302961.

[3] Breiman. “Random Forests”. In: 45 (2001).

[4] V. DePuy, V. W. Berger, and y. Zhou. “Wilcoxon-Mann-Whitney Test: Overview”. In: *Encyclopedia of Statistics in Behavioral Science* (2005).

[5] Y. Freund and R. Schapire. "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting". In: (1995).

[6] G. Hinton. "Connectionist learning procedures". In: *Artificial intelligence* (1989).

[7] C. Kassasseya et al. "Validation of a Predictive Score of Acute Chest Syndrome (presev-2 study) in Adults". In: *Blood* 136 (2020). URL: https://ashpublications.org/blood/article/136.

[8] G. Kato et al. "Sickle cell disease". In: *Nature Reviews Disease Primers* 4 (2018).

[9] O.Platt et al. "Mortality in sickle cell disease — life expectancy and risk factors for early death". In: *New England Journal of Medicine* 330 (1994).

[10] F. Piel, M. Steinberg, and D. Rees. "Sickle Cell Disease". In: *The New England Journal of Medicine* (2017).

[11] J. C. Platt. "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods". In: *Advances in Lage margin Classifiers* (1999).

[12] E. Sparkenbaugh and R. Pawlinski. "Interplay between coagulation and vascular inflammation in sickle cell disease". In: *British Journal of Hematology* 162 (2013).

[13] E. Vichinsky et al. "Causes and outcomes of the acute chest syndrome in sickle cell disease". In: *New England Journal of Medicine* 342 (2000).

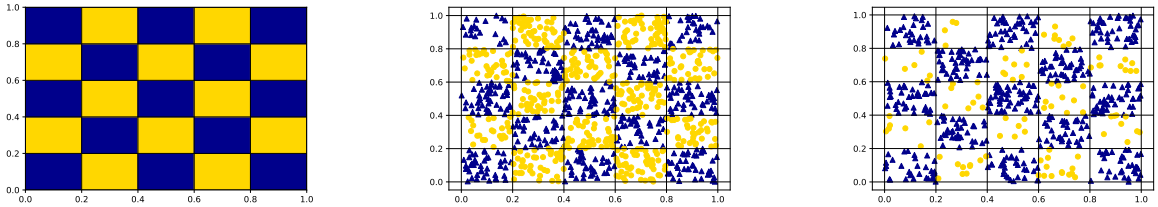
[14] R. Ware et al. "Sickle Cell Disease". In: *The Lancet* 390 (2017).

[15] C. Zhu et al. "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization". In: *ACM Transactions on Mathematical Software* (1997).

Chapter 2

Imbalanced Classification Challenge: A Review of the Literature

An imbalanced binary dataset is defined by a significant disparity between the number of data points belonging to each class. In other words, it is defined by a clear distinction between a dominant majority class and a less represented minority class. A typical visual example [11] of an imbalanced binary dataset is the chessboard one, figure 2.1a. In an ideal balance word, the chessboard dataset adopts the configuration shown in figure2.1b, in which the two classes - the blue and yellow ones - comprise an equal number of 500 data points each. However, the scenario changes in an imbalanced context, illustrated in figure 2.1c, where the chessboard data takes on a different appearance. In this case, the majority class (blue) comprises 500 data points, while the minority class (yellow) comprises just 100 data points. This visualization effectively highlights the challenge of separating minority and majority classes in an imbalanced context.



(a) Chessboard composed of blue and yellow squares.

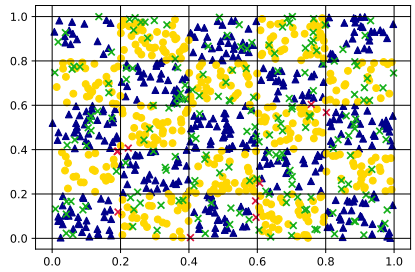
(b) Chessboard in the balanced world.

(c) Chessboard in the imbalanced world.

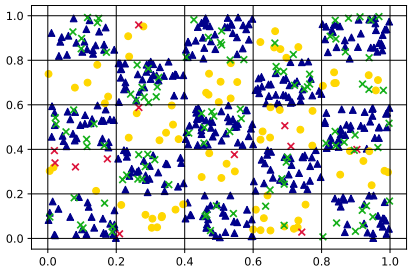
Figure 2.1: Example of a chessboard through balanced and imbalanced world. Blue triangles belong to blue squares and represent the majority class, while yellow circles belong to yellow squares and represent the minority class.

As part of an experiment, we decided to predict classes in two distinct situations: one with a balanced dataset and the other with an imbalanced dataset. Our experimental process involved randomly partitioning the dataset into training and test subsets. We then used a standard random forest classifier to make predictions and to evaluate the error rates.

The results are shown in Figure 2.2. Notably, in the balanced scenario, the error rate averaged 5%, while in the imbalanced scenario it reached 10%, more than double the error rate of the balanced scenario.



(a) Prediction using Random Forest Classifier on balanced chessboard



(b) Prediction using Random Forest Classifier on imbalanced chessboard

Figure 2.2: Prediction using a random forest classifier on the chessboard example. The blue triangles represent the majority class in the training part, the yellow circles represent the minority class in the training part, while the crosses belong to the testing part. Green crosses represent testing data points correctly predicted, while red crosses represent testing data points incorrectly predicted.

Consequently, our concerns have been validated, meaning that a loss of predictive accuracy accompanies the presence of imbalanced data. Clearly, imbalanced class distribution has a significant impact on predictive performance. This chapter explores in depth the complexities associated with imbalanced classification and the strategies proposed to effectively address these challenges.

2.1 Description of Imbalanced Data

Imbalanced datasets are a common phenomenon in a variety of fields, including medical disease diagnosis, fraud detection and image recognition. A remarkable resource known as KEEL [1] serves as a repository dedicated to the cataloging of datasets covering several domains. Table 2.1 shows a selection of imbalanced datasets from various fields.

	Abalone19 [26]	Page-blocks0	Paw	Pima [27]	Segment0 [14]	Vehicle1	Vowel0 [13]	Wisconsin	Yeast1	Haberman	Glass1	Ecoli1 [38]	Subcl35
Area	Biology	Class Text	Artificial Data	Diabetes	Image Segmentation	Transport	Deterding	Breast Cancer	Biology	Breast Cancer	Glass	Proteins	Artificial Data
Number of features	8	10	2	8	19	18	13	9	8	3	9	7	2
Number of samples	4174	5472	600	768	2308	846	988	683	1484	306	214	336	800
Number of minority data points	538	559	100	268	329	217	90	239	37	81	6	77	100
IR	12.9	8.79	5	1.87	6.02	2.9	9.11	34.97	2.46	2.78	35.46	22.94	7

Table 2.1: Description of Imbalanced Datasets selected from the KEEL repository [1].

In this first section, we explore the concept of imbalanced datasets. We explain in detail what constitutes an imbalanced dataset, elucidating the challenges it presents.

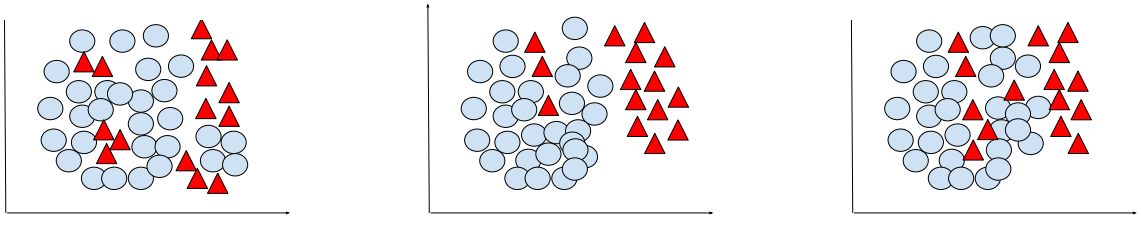
2.1.1 Imbalanced Data Challenges

Imbalanced datasets are characterized by a predominant majority class, a measurable aspect that can be evaluated using an imbalance ratio introduced in article [28]:

$$IR = \frac{\text{number of minority data points}}{\text{number of majority data points}}.$$

This imbalance ratio is used as a measure to capture the degree of disparity between minority and majority classes. A low ratio means that the dataset has a significant imbalance, while a high ratio indicates that the dataset is more evenly distributed. However, it is important to note that while the imbalance ratio provides insight into the structure of the dataset, it doesn't intrinsically determine the complexity of predictive tasks. This point is explored in the article [41], which explains that datasets with the same imbalance ratio can produce different results. The predictability of imbalanced classification problems depends on a complex set of parameters, including the imbalance ratio itself, the number of features, the sample size and the inherent characteristics of the data. However, it should be recognized that the challenges introduced by majority class dominance can lead to certain generalized errors.

As explained in the article [18], the exploration of imbalanced classification problems takes place in two distinct scenarios. The first scenario arises when the two classes can be separated distinctly using linear methods. As a result, the transformation of an imbalanced classification problem into a balanced one is particularly easy to solve directly. In contrast, the second scenario arises when linear separation between the classes is impossible. As a result, the classification problem becomes more complex, leading to three main challenges described in Figure 2.3 and explained in the book [11].



(a) Disjunctions (b) Outliers (c) Overlapping

Figure 2.3: The three main issues that can occur when the data is imbalanced. The blue circles represent the majority dataset while the red triangles represent the minority dataset.

The first challenge, called "disjunction", is illustrated in Figure 2.3a. In a balanced context, red triangles tend to connect to neighboring red

triangles, improving the ease with which they can be classified. However, this cohesion diminishes in an imbalanced context, allowing red triangles to become isolated, thus causing disjunctions. This first challenge introduces further complexity into the classification process.

The second challenge concerns "outliers", described in figure 2.3b. This circumstance causes classification algorithms to perceive certain data points as noisy, a misconception due to the sparsity of data within the minority class. Consequently, this problem constrains the application of statistical techniques such as feature selection, increasing the complexities associated with dealing with imbalanced data.

The third main challenge, known as "overlap", is illustrated in Figure 2.3c. It materializes when the two classes are closely imbricated, making the delimitation of a decision boundary particularly complex. Strongly influenced by the weight of the majority class, this boundary is skewed in favor of the majority class rather than the minority class, highlighting the complexity of the segregation between the two classes.

The trio of challenges described above, which predominate in imbalanced datasets, push conventional classification methods to prioritize the majority class, often to the detriment of the minority class. Consequently, using standard scoring measures to rank and compare classifications can lead to misinterpretations, as these measures focus primarily on the performance of the majority class.

2.1.2 Scores and Metrics

The two typical scores used for balanced classification are **accuracy** and **error rate**. They are defined using the notion of confusion matrix, explained in table 2.2.

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Table 2.2: Confusion Matrix

Accuracy (Acc) is computed using the formula:

$$Acc = \frac{TP + TN}{TP + TN + FN + FP}$$

where TP represents true positives, TN stands for true negatives, FN denotes false negatives, and FP corresponds to false positives. Simultaneously, the error rate can be determined as $error = 1 - Acc$. In the context of a balanced dataset, where both classes have the same importance, these

metrics work well. However, when it comes to imbalanced datasets, where class weights differ significantly, using these metrics can lead to erroneous interpretations. In particular, in imbalanced scenarios, the number of predicted minority data points is often outweighed by the number of predicted majority data points. This asymmetry can mask the models' effectiveness in identifying and correctly classifying the minority class. Consequently, when comparing several imbalanced classification results, relying only on *Acc* and *error* can give misleading results. For more accurate evaluations, it is recommended to use other measures that take into account the complexities of imbalanced datasets. These measures include :

- Precision = Positive Predictive Value PPV = $\frac{TP}{TP+FP}$, evaluation of the number of correct predicted minority data points over all the data points predicted as minority.
- Recall = Sensitivity = $\frac{TP}{TP+FN}$, evaluation of the number of predicted minority data points over all the minority data points.
- F1 score = $2 * \frac{Precision*Recall}{Precision+Recall}$, analysis prediction done over the minority class.
- AUC Score = $\frac{FP}{FP+TN}$, to evaluate the impact of the classifier.
- GMean = $\sqrt{\frac{TP}{TP+FN} * \frac{TN}{TN+FP}}$ allows to balance the evaluation between minority and majority data points. The GMean score can be low even if the majority class is well predicted.
- Balanced Accuracy = $\frac{Precision+Recall}{2}$, looks like accuracy but takes into account the imbalanced criteria.

By way of illustration, let's compare the predictions generated from two distinct datasets, as shown in 2.1. The results of this comparison have been compiled in Table 2.3. Noticeable disparities in scores appear between the balanced and imbalanced chessboard datasets.

	Error Rate	Precision	Recall	F1 score	GMean	Balanced Accuracy
Imbalanced ChessBoard	10.8	100	35	51.8	59.2	67.5
Balanced ChessBoard	7.5	94.7	90	92.3	92.5	92.4

Table 2.3: Scores obtained using random forest clasiffier for the prediction of the balanced chessboard 2.2a and the imbalanced chessboard 2.2b.

For example, measures such as F1 score, geometric mean (GMean), balanced precision and recall perform significantly better on the balanced chessboard dataset. This difference highlights the significant impact of

class imbalance, which adversely influences prediction results. Clearly, the choice of evaluation metric is of key significance when evaluating imbalanced classifications, as it will support the analysis and understanding of the main associated challenges. It is therefore imperative to carefully select an appropriate evaluation metric that takes into account the challenges of imbalanced classifications. By doing so, it is possible to gain a true understanding of the difficulties posed by class imbalances, and to accurately assess the performance of predictive models in this context.

In addition, the results presented in Table 2.3 highlights a notable distinction between performance measures derived from imbalanced and balanced datasets. As we have seen, conventional machine learning algorithms, such as the random forest algorithm, are intrinsically designed for situations in which the datasets are balanced. Thus, in order to improve results, specialized techniques designed to handle imbalanced datasets have been developed. These techniques, often referred to as "imbalanced methods", emerged in response to the challenges related to imbalanced class distributions.

2.2 Imbalanced Methods

Imbalanced methods can be categorized into three groups: data sampling, algorithm modification, and cost-sensitive learning. This section will provide a comprehensive overview of these three groups, investigating into their distinct approaches and methodologies.

2.2.1 Data Level / Sampling Method

The data sampling strategy aims to modify the training set to obtain a balanced distribution. There are two basic approaches to this strategy: undersampling and oversampling. Undersampling involves reducing the size of the majority class by removing certain data points. However, in cases where the dataset is already limited, the application of undersampling methods may not be possible. Consequently, oversampling, which involves introducing new data points into the minority class, appears to be the most common and practical imbalanced method. In practice, a large range of over a hundred sampling methods has been developed to meet this challenge. Here, we present a concise overview of some of these methods:

Undersampling methods – Undersampling methods aim to reduce class imbalance by reducing the number of data points of the majority class in the original dataset. The fundamental approach is to randomly select and remove instances of the majority class. This simple technique is known as "random subsampling". However, it should be noted that the

blind removal of data points risks unintentionally removing precious information, which could complicate classification tasks. To avoid these issues, several other methods based on different concepts have been developed. The following list is not exhaustive.

- Undersampling method based on neighborhood:
  - Tomek Links (TL) [34]: This method is based on removing data points from the majority class through the notion of "Tomek Links". A set of two data points is defined as "Tomek" if they both belong to the majority class and if there is no other majority data point closer to each of them.
  - One Sided Selection (OSS) [5]: OSS is a mixed method of the TL one and the Condensed Nearest Neighbour Rule (US-CNN) [16] method. In fact, by applying TL followed by US-CNN, OSS selects and keeps only the majority data points next to the minority data points, i.e. next to the border. The TL method is used to delete noisy majority data points, i.e. far from the others whereas the US-CNN method works as follow. It first randomly selects a partition of majority data points and concatenates it with all other minority data points. We call this selection E. Then the method checks the remaining majority data points one by one by applying a one nearest neighbour method on E. If the data point is well ranked, it will be deleted. If not, i.e. if the data point is close to the boundary, it will be added to the partition E.
- Undersampling method based on clustering:
  - ClusterOSS [3]: Based on one-sided selection (OSS) [5] and clustering method, ClusterOSS removes majority data points accross different regions. The first step is to cluster the majority data points, using the kmeans [19] algorithm. Then, the OSS method is applied using clusters centres as the first partition, i.e. as E. At the end the TL method is applied to remove the last noisy data points.
- Undersampling method based on weight and probabilities:
  - Weighted Sampling [2]: By assuming that data points of different classes closed to each other are likely to be misclassified, this method is based on a probability classifier which selects the data points far from the boundary. The first step is to calculate a weight for all majority data points based on their distance from the minority data points. The user then defines a ratio and selects only those majority data points with a weight greater than this ratio.

Oversampling methods – Oversampling methods aim to reduce class imbalance by increasing the number of data points of the minority class in the dataset. To achieve this, new instances are generated on the basis of existing instances of the minority class. The simplest oversampling technique is to randomly select and duplicate existing data points. However, direct copying of identical instances can introduce noise and potentially disrupt classification accuracy. In addition, this method can alter the distribution of the data. To avoid these issues, several other methods based on different concepts have been developed. The following list is not exhaustive.

- Ordinary sampling and interpolation concept:
  - SMOTE [7]: It is the most popular oversampling method and many methods are based on it. The SMOTE algorithm can be decomposed into 3 steps. The first one is to construct the list of the k nearest neighbors of each minority data point. In a second time, n_added minority points are randomly chosen and for each minority data point x_1 picked, a random nearest neighbor x_2 of x_1 is chosen. The last step consists in building a new data point x between x_1 and x_2 with a linear interpolation:

    $$\mathrm{x} = x_1 + \alpha(x_2 - x_1) \tag{2.1}$$

    where α is a scalar uniformly sampled in $[0, 1]$.
- Selection based on significance, data distribution, density, or relationships between data points:
  - ProWSyn [4]: For each minority data point, ProWSyn defines a proximity level. This proximity level assesses the relationship and distance between the minority and majority points. This score is then normalised and considered as a weight. Minority points located at the borders, i.e. close to majority points, will have a higher weight than those far from the borders. The minority points are then selected in proportion to their weight, and a linear interpolation (2.1) is used to create a new point x.
- By defining the space where minority data points can be generated, based on the data distribution and empty spaces. For example, by using clustering:
  - Geometric SMOTE [8]: Geometric SMOTE is a generalization of SMOTE with the objective of reducing the number of minority data points generated in the majority areas. Thus, geometric SMOTE defines an elliptical area around the minority data points and, by deformation and truncation, secures the area where the new data points are generated.

– DBSMOTE [6]: DBSMOTE relies on the DBSCAN algorithm [31] to construct minority class clusters. This pre-clustering is used to estimate a local distribution of the data and to find the boundaries between the classes. Then, new minority data points are added using the equation (2.1) in the clusters.

One of the main challenges associated with data sampling techniques is the potential alteration of the data distribution. In some cases, the application of these techniques can also lead to an increase in the number of noisy data points. These effects can have an impact on the overall integrity of the dataset and, consequently, on the quality of the classification results. To address these concerns and ensure the preservation of the fundamental distribution and quality of the data, adaptations have been made to standard machine learning algorithms. These modifications are designed to enable these algorithms to operate more efficiently in the context of imbalanced data.

2.2.2 Algorithmic Methods

The development and implementation of algorithmic methods to solve imbalanced classification problems is a particularly complex enterprise. It requires a deep understanding of the algorithm's limitations in handling these type of datasets. These limitations may come from various aspects of the algorithm's theoretical foundations. In this section, we examine modifications to the Support Vector Machine (SVM) algorithm to improve its performance in imbalanced classification scenarios. So, first we need to deeply understand the theory of usual Support Vector Machine algorithm.

The fundamental objective of the support vector machine (SVM) is to determine a decision boundary, commonly known as a hyperplane. This boundary can be linear or adapted to be non-linear through the use of a projection kernel, in order to efficiently separating distinct classes within a dataset, as shown in Figure 2.4.

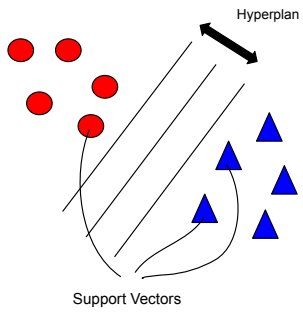


Figure 2.4: Representation of a usual Support Vector Machine (SVM) algorithm. The red circles represent the class A whereas the blue triangles represent the class B.

The concept of a non-linear decision boundary can be represented using a mapping function, denoted as $w \cdot \phi(x) + b = 0$, where w signifies weights and b denotes biases. Given that most datasets are not linearly separable, this leads to the formulation of an optimization problem as follows:

$$\min \left(\frac{1}{2} w \cdot w + C \sum_{i=1}^{l} \xi_i \right) \text{ subject to } \forall_{i=1,..,l} \forall_{\xi_i \geq 0} \; y_i (w \cdot \phi(x_i) + b) \geq 1 - \xi_i$$

Of notable importance for modifying the SVM algorithm is the $\sum_{i=1}^{l} \xi_i$ term. This term corresponds to the penalty factor, reflecting how many training instances might fall on the incorrect side of the decision boundary. This problem is transformed into a more solvable quadratic programming issue:

$$\max_{\alpha_i} \left(\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \phi(x_i) \phi(x_j) \right)$$
$$\text{subject to } \forall_{i=1,...,l} \forall_{0 \leq \alpha_i \leq C} y_i \alpha_i = 0$$

However, solving this expression by finding the mapping function ϕ can be really complex. To facilitate the solution, the mapping function is transformed into a kernel function, $K(x_i, x_j) = \phi(x_i)\phi(x_j)$. As a result, the SVM equation can be expressed as:

$$f(x) = \text{sign}(w \cdot \phi(x) + b) = \text{sign} \left(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b \right)$$

By examining these equations, three main challenges arise when SVM is applied to imbalanced datasets:

1. **Misclassification Cost (Parameter C)** - The parameter C represents the misclassification cost and serves as a penalty for errors in the training set. This cost is uniform for both classes, which means that errors in the minority class are penalized equally as those in the majority class. Consequently, the resulting hyperplane is biased towards the minority class due to the penalty on its errors.

2. **Number of Support Vectors (Parameter α_i)**: The significance of the parameter α_i, which denotes the number of support vectors, is more pronounced for the majority class compared to the minority class. This imbalance in support vectors can affect the SVM's generalization capacity towards the minority class, potentially leading to reduced performance.

3. **Data Density Influence on Decision Function**: The decision function's behavior is linked to the data point density. In the case

of imbalanced datasets, the limited number of minority class data points translates to insufficient data density in regions where the classification boundary should ideally be estimated, leadning to reduce performance.

To reduce the impact of these three challenges, SVM algorithms have been adapted and modified. Three notable modifications of SVM have been identified and listed in articles [11, 15]:

1. The first modification can be called kernel modifications. These modifications lead to different algorithms described in articles [36, 39, 35, ...]. Modifying the kernel function returns to modify the decision boundary and the margin. These modifications can be done before training the algorithm by weighting the data points belonging to their classes. But these modifications can also be done by scaling the kernel function based on a enlarging transformation on both sides of the decision boundary in a independant weights (cf. article [25]).

2. The second modification listed in book [11] is a modification of SVM by weighting training instances according to their importance. This method is very popular and is used in many different ways in other common machine learning methods. The first way to influence the instances is to weight the regularisation parameter according to the instance as described in the following equation (cf. article [37]):

$$min(\frac{1}{2}w.w + \sum_{i=1}^{l} C_i\xi_i)$$
$$\text{subject to } \forall_{i=1,..,l}\forall_{\xi_i \geq 0}\ y_i(w.\phi(x_i) + b) \geq 1 - C_i\xi_i$$

   A second way to modify the weight of SVM is to assign a different level of importance to each training instance by associating them to a fuzzy membership function such as in the following equation:

$$min(\frac{1}{2}w.w + C\sum_{i=1}^{l} f_i\xi_i)$$
$$\text{subject to } \forall_{i=1,..,l}\forall_{\xi_i \geq 0}\ y_i(w.\phi(x_i) + b) \geq 1 - \xi_i$$

   This membership function f_i was developed in the articles [24, 23]. f_i is associated to each instance and its defined through a decaying function returning values in [0, 1]. The aim of this function is to calibrate the margin and bring it closer to the majority data points.

3. The last modification can be performed through active learning. In fact, by selecting a balanced subset of the dataset, applying a standard SVM method and repeating the experiment, the SVM will be able

to deal better with imbalanced classification problem. The selection of balanced training subset can be done through different mecanisms such as by selecting only the minority data points far from the decision boundary or closed to the minority data points. A example of active learning SVM is explained in article [32].

All the modifications discussed above can also be applied to other machine learning algorithms, such as decision trees and ensemble methods like boosting. In particular, decision tree algorithms have been modified for imbalanced datasets, as explained in article [22]. In addition, ensemble methods, including boosting techniques, have been adapted to deal with class imbalance, as shown in articles such as [20, 10]. These modifications often involve the introduction of weights or functions to rebalance the learning process in favor of data points from minority classes.

However, it is important to recognize that while algorithm modification techniques have potential advantages for handling imbalanced data, they may be less flexible than sampling methods.

2.2.3 Cost-Sensitive Learning

Cost-sensitive learning (CSL) methods involve the integration of weights directly into machine learning algorithms, during the final stages. CSL is a category of algorithmic modifications. These weights are assigned to instances or classes according to their relative importance. While CSL methods are often used for feature selection to improve balanced classification problems, they can also be extended to deal with imbalances between classes. By adjusting the weights associated with different classes, CSL methods aim to improve predictions of imbalanced classifications. CSL techniques have been applied to a variety of algorithms, including Support Vector Machines (SVMs) [21], Decision Trees [33, 9], Nearest Neighbors [29], Neural Networks [40], and Metric Learning [12]. We will focused more in the Metric Learning theory and methods in Chapter 6. CSL is still under development especially for neural network algorithms.

2.3 Conclusion

Imbalanced datasets are a major challenge, leading to the emergence of new predictive methods and scores. “Imbalanced” methods can be split into three distinct categories: Data sampling, algorithm modification and cost-sensitive learning. Over the years, this has led to the development of over a hundred methods, including direct modifications to data points, modifications to conventional algorithms, the use of weights, probabilities

and so on. Nevertheless, it's important to recognize that, despite the extensive variety of these methods, none is a silver bullet. As explained and confirmed in numerous articles [30, 17], there is no single solution for imbalanced datasets. The main challenge is to determine the most appropriate method for a specific imbalanced dataset. In pursuit of this objective, we have designed and implemented a complete pipeline

References

[1] J. Alcalá-Fdez et al. "KEEL: a software tool to assess evolutionary algorithms for data mining problems". In: *Soft Computing* 13.3 (2009), pp. 307–318.

[2] A. Anand et al. "An approach for classification of highly imbalanced data using weighting and undersampling". In: *Amino Acids* 5 (2010).

[3] V. Barella, E. Costa, and A. Carvalho. "ClusterOSS: a new undersampling method for imbalanced learning". In: *Computer Science* (2014).

[4] S. Barua, M. Islam, and K. Murase. "ProWSyn: Proximity Weighted Synthetic Oversampling Technique for Imbalanced Data Set Learning". In: *PAKDD* Advances in Knowledge Discovery and Data Mining (2013).

[5] G. Batista, A. de Carvalho, and M.-C. Monard. "Applying One-Sided Selection to Unbalanced Datasets." In: *Lecture Notes in Computer Science* (2000).

[6] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap. "DBSMOTE: Density-Based Synthetic Minority Over-sampling TEchnique". In: *Applied Intelligence* 36 (2012).

[7] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (2002).

[8] G. Douzas and F. Bacao. "Geometric SMOTE: Effective oversampling for imbalanced learning through a geometric extension of SMOTE". In: *Information Sciences* 501 (2017).

[9] C. Drummond and R.C. Holte. "C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling." In: *ICML Workshop on Learning from Imbalanced Datasets II, Washington, DC* (2003).

[10] W. Fan et al. "AdaCost: Misclassification Cost-sensitive Boosting". In: *ICML'99: Proceedings of the Sixteenth International Conference on Machine Learning* (1999).

[11] A. Fernández et al. *Learning from Imbalanced Data Sets*. Springer Cham, 2018.

[12] L. Gautheron et al. "Metric Learning from Imbalanced Data." In: *International Conference on Tools with Artificial Intelligence (ICTAI)* (2019).

[13] R. P. Gorman and T. J. Sejnowski. "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets". In: *Neural Networks* 1 (1988).

[14] Massachusetts Vision Group. "UCI Image Segmentation Data". In: (2018). URL: https://www.openml.org/d/36.

[15] G. Haixiang et al. "Learning from class-imbalanced data: Review of methods and applications". In: *Expert Systems With Applications* 73 (2017).

[16] P. Hart. "The condensed nearest neighbor rule (Corresp.)" In: *IEEE Transactions on Information Theory* 14 (1968).

[17] A. B. Hassanat et al. “Stop Oversampling for Class Imbalance Learning: A Critical Review”. In: *Digital Object Identifier* (2022).

[18] N. Japkowicz. “The Class Imbalance Problem: Significance and Strategies”. In: *In proceedings of the 2000 international conference on artificial intelligence (ICAI)* (2000).

[19] X. Jin and J. Han. “K-Means Clustering”. In: *Encyclopedia of Machine Learning* (2007).

[20] M. V. Joshi, V. Kumar, and R. C. Agarwal. “Evaluating Boosting Algorithms to Classify Rare Classes: Comparison and Improvements”. In: *IEEE International Conference on Data Mining (ICDM)* (2001).

[21] S. Katsumata and A. Takeda. “Robust cost sensitive support vector machine.” In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015* (2015).

[22] P. Lenca et al. “A Comparison of Different Off-Centered Entropies to Deal with Class Imbalance for Decision Trees”. In: *Advances in Knowledge Discovery and Data Mining* 5012 (2008).

[23] C. Lin and S. Wang. “Fuzzy Support Vector Machines”. In: *IEEE Transactions on Neural Networks* 13 (2002).

[24] J. Liu. “Fuzzy support vector machine for imbalanced data with bordeline noise”. In: *Science Direct* 413 (2021).

[25] A. Maratea, A. Petrosino, and M. Manzo. “Adjusted F-measure and kernel scaling for imbalanced data learning”. In: *Information Sciences* 257 (2014).

[26] W. J. Nash et al. “The Population Biology of Abalone Haliotis species) in Tasmania. I. Blacklip Abalone (H. rubra) from the North Coast and the Islandsof Bass Strait”. In: *Sea Fisheries Division, Technical Report* (1994).

[27] N. Nnamoko and I. Korkontzelos. “Efficient treatment of outliers and class imbalance for diabetes prediction”. In: *Artificial Intelligence in Medicine* 104 (2020).

[28] A. Orriols-Puig and E. Bernad´o-Mansilla. “Evolutionary Rule-Based Systems for Imbalanced Datasets”. In: *Soft Computing* 13 (2009).

[29] Z. Qin et al. “Cost-sensitive classification with k-nearest neighbors.” In: *6th International Conference on Knowledge Science, Engineering and Management, KSEM2013* (2013).

[30] B. Santos et al. “Synthetic Over Sampling Methods for Handling Class Imbalanced Problems: A review”. In: *IOP Conference Series: Earth and Environmental Science* 58 (2017).

[31] E. Schubert et al. “DBSCAN revisited, revisited: why and how you should (still) use DBSCAN”. In: *ACM Transactions on Database Systems (TODS)* 42.3 (2017), pp. 1–21.

[32] R. Sundar and M. Punniyamoorthy. “Performance enhanced Boosted SVM for imbalanced datasets”. In: *Applied Soft Computing Journal* 83 (2019).

[33] K.M. Ting. “An instance-weighting method to induce cost-sensitive trees.” In: *IEEE Transactions on Knowledge and Data Engineering* 14 (2002).

[34] I. Tomek. “Two modifications of CNN”. In: *In Systems, Man, and Cybernetics* 6 (1976).

[35] Y. Xu. “Maximum Margin of Two Spheres Support Vector Machine for Imbalanced Data Classification”. In: *IEEE Transactions on Cybernetics* (2016).

[36] C. Yang, J. Yang, and J. Wang. "Margin calibration in SVM class-imbalanced learning". In: *Neurocomputing* 73 (2009).

[37] X. Yang, Q. Song, and Y. Wang. "A weighted support vector machine for data classification." In: *IJPRAI* 21 (2007).

[38] Y. Yang and C. Jobin. "Microbial imbalance and intestinal pathologies: connections and contributions". In: *Dis Model Mech* 7 (2014).

[39] H. Yu et al. "Support vector machine-based optimized decision threshold adjustement strategy for classifying imbalanced data". In: *Knowledge-Based Systems* 76 (2015).

[40] Z. Zhou and X. Liu. "Training cost-sensitive neural networks with methods addressing the class imbalance problem." In: *IEEE Transactions on Knowledge and Data Engineering* 18 (2006).

[41] R. Zhu, Y. Guo, and J.H. Xue. "Adjusting the imbalance ratio by the dimensionality of imbalanced data". In: *Pattern Recognition Letters* 133 (2020).

Chapter 3

ImbPip: A Pipeline for Comparing Imbalanced Methods

As demonstrated in the previous chapter, predictions made on imbalanced datasets using conventional machine learning methods are often biased. Consequently, various methods have been designed to improve predictive results, such as data sampling or cost-sensitive learning methods. However, in the real world, when users are looking to predict a specific imbalanced dataset, this process often involves manual, time-consuming comparisons of multiple algorithms to find the perfect one. In contrast, for balanced datasets, pipelines established in different libraries, such as "sklearn" or "keras", simplify the task of comparing ML models. This convenience is not extended to imbalanced datasets, where users are often limited to trying only the best-known methods, such as SMOTE [9], DBSMOTE [7], or GSMOTE [13]. As a result, prediction for imbalanced classification is often far from optimal. Recognizing this gap, we set out to develop a model comparison pipeline specifically designed for "imbalanced" methods. The aim is to create an accessible and efficient framework that enables users to navigate the complexities of imbalanced datasets with the same ease and efficiency that balanced datasets benefit from established libraries

3.1 Pipeline Implementation

Our pipeline is designed to identify the most efficient imbalanced classification method, adapted to a specific imbalanced classification problem. To do this, the pipeline takes into account two key elements: an imbalanced dataset and a designated scoring metric. Since strategies for binary and multiple-output datasets diverge, our pipeline focuses only on binary datasets. In addition, in its current state, the pipeline incorporates the score metrics described in section 2.1.2. If users wish to explore other scoring measures, they must implement them manually before running the pipeline.

The results of our pipeline consists of the name of the selected imbalance method that gives the best results and the corresponding parameters. In this first version of the pipeline, we have deliberately limited our scope to data sampling methods. These methods are widely recognised and available. It is important to note that effective use of these methods requires them to be combined with standard machine learning algorithms. Therefore, our pipeline will systematically evaluate and compare various combinations of data sampling methods and machine learning algorithms.

The first and most important criterion of our pipeline is the focus on **reproducibility**. To this end, our pipeline is open source and designed to be extensible, allowing the integration of various other imbalanced methods. The basis of reproducibility in our pipeline is based on cross-validation techniques. By splitting the dataset into multiple training and test subsets, we are able to quantify variability. Ultimately, this allows us to identify the method with the lowest variability, indicating its reliability and robustness. In the context of imbalanced method comparisons, the notions of reliability and variability are of great importance. Many of these methods are closely linked to the distribution of training data points, and this link can sometimes lead to problems such as over-fitting. Careful consideration of reliability and variability can minimize these concerns.

In line with our dedication to open source, the source code for our pipeline is available on GitHub https://github.com/yamnao/ImbPip. This facilitates access and encourages peer review, helping to improve the quality and reliability of the pipeline. We have named our pipeline 'ImbPip' and its conceptual framework is succinctly represented in the diagram in Figure 3.1.

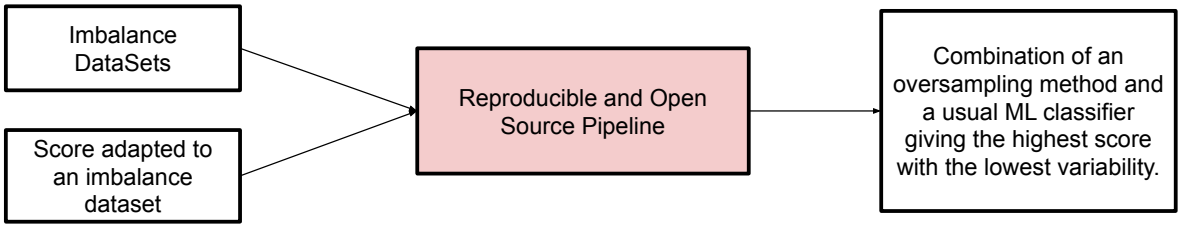


Figure 3.1: Outline of the ImbPip Pipeline.

Previous studies such as those described in the papers [22, 30] have focused on imbalanced method comparisons; these comparisons typically produce rankings without considering method parameters and variability. In contrast, our pipeline is designed to offer users a complete perspective by providing both method parameters and variability scores. Our main objective is to establish a reliable and reproducible pipeline. To do this, the

initial stage of the pipeline involves a meticulous examination of the variabilities. We have adopted a strategy of randomly segmenting the supplied dataset into 10 folds. In addition, we gave priority to the transparency of the code. To this end, we have stored these 10 folds, allowing users to faithfully reproduce our results. Also, by using these stored files, users can easily incorporate new scoring metrics. This allows the generation of results using different metrics without having to repeat all the experiments. This approach not only saves time, but also minimises the risk of introducing bias.

Our second priority for ImbPip is to compare results between several sampling methods. Thus, we selected the following 20 data sampling methods: Assembled SMOTE [34], CCR [24], Cluster SMOTE [11], CondensedNearestNeighbors [16], Cure SMOTE [27], DBSMOTE [7], DE Oversampling [10], EditedNearestNeighbours [18], GSMOTE [13], InstanceHardnessThreshold, KMeans SMOTE [20], Lee [26], NearMiss [28], PolynomFit SMOTE [15], OneSidedSelection [3], ProWSyn [2], SMOBD [8], SMOTE [9], SMOTE IPF [31], TomekLinks [32] and SMOTE-TomekLinks [4] . We chose all these methods because they are the most known or have the best ranking according to article [22]. We coded these methods using existing pseudo-codes. Nevertheless, to allow future users to add others data sampling methods, we implemented each method with a specific skeleton. As data sampling need to be used in combination with ML classifiers, we also picked 5 ML classifiers: SVM [33], RandomForest [6], KNearestNeighbors[12], AdaBoost [14] and MLPClassifier [19].

In brief, using the input data described above, we have exploited the potential of data sampling methods to predict the outcomes of a given imbalanced dataset. The essence of this process is to identify the most effective combination of these techniques. To do this, we systematically explore different combinations of parameters. Given the potential explosion in the number of parameters to be taken into account, we have imposed a constraint of a maximum of 30 feasible parameter combinations. The final step in our pipeline is to present the results in an understandable format. To do this, we opted for a tabular representation. Each row of the table corresponds to a selected imbalance score, and each column offers different information:

- Best data sampling method employed,
- Parameters associated with the chosen sampling method,
- Optimal ML classifier employed,
- Parameters of the selected ML classifier.

To offer a more intuitive understanding of these steps, we present the pseudo-code of the algorithm depicted in the Pseudo-code 1 below:

Algorithm 1: Pseudo-code of ImbPip

Data:

- Name_D: DataSet Name
- Sampling_List: list of sampling methods
- Classifiers_List: list of classifier algorithms
- Scores_List: list of scores under studied

```
/* Step 1:  Data Splitting                                          */
folds: split_data_into_10_folds(Name_D);
/* Step 2:  Data Sampling and ML Classifier                         */
best_combination: None;
best_score: 0;
results: init dictionary;
for S_strat in Sampling_List do
    S_Params: Get Parameters Combination for S_Strat;
    for S_P in S_Params do
        Sampling_Data: Generate sampling data using S_P and S_Strat on the 10
         folds files;
        for C_Strat in Classifiers_List do
            for train, test, train_labels, test_labels in Sampling_Data do
                result: evaluate_combination(train, test, train_labels, test_labels,
                 C_Strat, Scores_List);
                if result ≥ best_score then
                    best_score = result;
                    best_combination = (C_Strat, S_strat, S_P);
/* Step 3:  Present Results in Table                                */
```

Result: results, each row corresponds to a score and columns are completing with Mean score, sampling method, sampling method parameters, classifier method, classifier method parameters.

3.2 Applications

Having completed the design and development of our ImbPip pipeline, the next phase is to perform tests on different imbalanced datasets. To this end, we have selected five distinct imbalanced datasets, as illustrated in the table 2.1. The selected datasets are as follows: Paw, Vehicle1, Haberman, Ecoli1 and Subcl35. Then, we will try our pipeline on the detection of ACS.

3.2.1 Results on real-world datasets

The typical result from our ImbPip pipeline is shown in table 3.1. However, for the purpose of clarity and simplicity, we have grouped the results obtained from the other datasets into a unified table 3.2. To ensure the precision of our pipeline in generating optimal method combinations, we systematically report all scores obtained by the 20 data sampling meth-

ods. By manually comparing these results with the best results identified by ImbPip, we can confidently state that our pipeline does indeed generate the expected results. In addition, we have supported the performance of our pipeline by presenting results consistent with those elucidated in previous studies [17, 5].

	score	Sampling Method	Sampling Parameters Method	Classifier Method	Classifier Parameters Method
precision	68,9	Cluster SMOTE	n_neighbors:5 n_clusters:200	KNN	leaf_size:30 metric: minkowski n_neighbors: 5
recall	100	Assembled SMOTE	proportion:2 n_neighbors:3 pop:2 thres:0.3	SVM	C:1
f1	54,1	ProWSyn	proportion:0,5 k_neighbors:3 L:3 theta:0,8	RF	max_depth:5 min_s_leaf:1
auc	100	Assembled SMOTE	proportion:2 n_neighbors:3 pop:2 thres:0.3	SVM	C:1
gmean	82,7	Assembled SMOTE	proportion:2 n_neighbors:5 pop:2 thres:0.3	RF	max_depth:5 min_s_leaf:1
npv	100	Polynom Fit SMOTE	nb_add:2 interpolation:mesh	RF	max_depth:5 min_s_leaf:1

Table 3.1: Table obtained using ImbPip pipeline on Subcl35 dataset for 6 scores: precision, recall, f1, auc, gmean and npv.

		precision	recall	f1	auc	gmean	npv
Paw	(sampling, classifier)	OSS, KNN	Assembled, SVM	Lee, KNN	Assembled, SVM	Smote IPF, KNN	Polynomfit, AdaBoost
	mean score	72,8	100	58,1	100	83,1	100
Subcl35	(sampling, classifier)	Cluster SMOTE, KNN	Assembled, SVM	ProWSyn, RF	Assembled, SVM	Assembled, RF	Polynomfit, RF
	mean score	68,9	100	54,1	100	82,7	100
Vehicle1	(sampling, classifier)	OSS, KNN	CCR, SVM	SMOTE IPF, MLP	CCR, SVM	Smote IPF, MLP	SMOBD, SVM
	mean score	66,3	100	56,7	99,8	71,7	93,8
Harberman	(sampling, classifier)	kMeans SMOTE, SVM	Assembled, MLP	Assembled, SVM	Assembled, MLP	GSMOTE, AdaBoost	GSMOTE, KNN
	mean score	50	100	42,3	100	51,2	79,5
Ecoli1	(sampling, classifier)	DBSMOTE, MLP	SMOBD, MLP	Cure SMOTE, SVM	Cure SMOTE, RF	Cure SMOTE, SVM	SMOBD, MLP
	mean score	88,3	100	79,1	67,8	85,9	99,2

Table 3.2: Summarize of results obtained using the ImpPip on 5 imbalanced datsets.

It is clear that no single data sampling method provides a silver bullet solution. In this context, the ImbPip pipeline becomes a valuable tool to help users identify the most appropriate prediction method for a given imbalanced classification problem. Its ability to systematically evaluate and determine the best combinations between different techniques makes it an essential tool for dealing with imbalanced datasets.

3.2.2 Results on PRESEV dataset

We have extended the tests in our pipeline to address the PRESEV classification challenge. As discussed in the previous chapter, the task of detecting ACS using the PRESEV dataset is an imbalanced classification problem, and conventional machine learning methods tend to overfit and have large variability. To address this, we exploited our ImpBib pipeline to identify the best imbalanced methods capable of improving the highest positive

predictive value (PPV) and negative predictive value (NPV) scores. The results of this ranking procedure are presented in Table 3.3.

Oversampling	**Classifier**	**PPV**	**NPV**
Gaussian_SMOTE [25]	SVM	52 ± 27.7	83.7 ± 2.5
ROSE [29]	SVM	48.7 ± 24.1	85.1 ± 3.1
SMOBD [8]	SVM	48.3 ± 22.8	83.6 ± 2.4

Table 3.3: Ranking of oversampling methods for PRESEVC dataset on PPV and NPV values.

After this classification phase, we applied a methodology similar to that described in the previous chapter 1.The predictions of the dual machine learning algorithm were transformed into probability estimates. Summarising these results in the table 3.4, we noted a reduction in overfitting and cross variability, associated with an increase in NPV. However, it is worth noting that the PPV value decreased slightly compared to the figures reported in the articles [1, 21] and in table 1.4.

Gaussian_SMOTE	PPV	NPV
Train	24.6 ± 4.2	96.9 ± 2.9
Test	23.6 ± 3.5	96.5 ± 0.8

Table 3.4: PPV and NPV values obtained using Gaussian SMOTE [25] oversampling method on PRESEVC dataset.

3.3 Conclusion

As the results section shows, our ImbPip pipeline proved to be highly reproducible and flexible, capable of efficiently predicting a range of imbalanced classification problems. Its user-friendly nature makes it easy to implement, and the ability to add new data sampling methods reinforces its adaptability. Indeed, this version of the pipeline marks a significant advance previous pipeline such as the one edescribed in article [22], which focused primarily on classification and ranking of sampling methods. ImbPip's focus on considering method variability sets it apart, offering not only a general comparison, but also a reproducible result adapted to each dataset.

However, G. Kovács had also evolved his pipeline [22]. The new version published in 2022 seemed to echo the functionalities of the initial version of our ImbPip pipeline. Given this evolution and the availability of the new version of the Kovács pipeline, we decided not to continue with new versions of ImbPip. In addition, we have chosen to use this new pipeline called, smote_variants, and explained in the article [23] to generate future results.

It is interesting to note that regardless of the pipeline used (ImbPip or smote_variants), the results obtained on PRESEV are not convincing. If the variability of the results has been reduced, it is clear that the PPV score has also decreased. The application of various imbalanced methods did not provide significant results for the PRESEV dataset. Given this situation, our next objective is to develop new methods that are better adapted to the complexities of the PRESEV dataset. Thus, our next goal is to create methods that are fine-tuned to the complexities of the PRESEV dataset, with the dual aim of improving the PPV score and maintaining low over-fitting and cross-variability. To this end, we have linked one image segmentation technique known as the watersheds to the challenges associated with imbalanced datasets.

References

[1] P. Bartolucci et al. "Score Predicting Acute Chest Syndrome During Vaso-occlusive Crises in Adult Sickle-cell Disease Patients". In: *EBioMedicine* 10 (2016). URL: https://linkinghub.elsevier.com/retrieve/pii/S2352396416302961.

[2] S. Barua, M. Islam, and K. Murase. "ProWSyn: Proximity Weighted Synthetic Oversampling Technique for Imbalanced Data Set Learning". In: *PAKDD* Advances in Knowledge Discovery and Data Mining (2013).

[3] G. Batista, A. de Carvalho, and M.-C. Monard. "Applying One-Sided Selection to Unbalanced Datasets." In: *Lecture Notes in Computer Science* (2000).

[4] G. Batista, R. Prati, and M. Monard. "A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data". In: *SIGKDD Exploration Newsletter* 6 (2004).

[5] G.E. Batista, R.C. Prati, and M.C. Monard. "A Study of the Behavior of Several Methods for Balancing machine Learning Training Data". In: *ACM SIGKDD Explorations Newsletter* 6 (2004). URL: https://dl.acm.org/doi/10.1145/1007730.1007735.

[6] Breiman. "Random Forests". In: 45 (2001).

[7] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap. "DBSMOTE: Density-Based Synthetic Minority Over-sampling TEchnique". In: *Applied Intelligence* 36 (2012).

[8] Q. Cao and S. Wang. "Applying Over-sampling Technique Based on Data Density and Cost-sensitive SVM to Imbalanced Learning". In: *2011 International Conference on Information Management, Innovation Management and Industrial Engineering* 2 (2011).

[9] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (2002).

[10] L. Chen et al. "A Novel Differential Evolution-Clustering Hybrid Resampling Algorithm on Imbalanced Datasets". In: *2010 Third International Conference on Knowledge Discovery and Data Mining* (2010), pp. 81–85.

[11] D. Cieslak, N. Chawla, and A. Striegel. "Combating imbalance in network intrusion datasets". In: *2006 IEEE International Conference on Granular Computing* (2006), pp. 732–737.

[12] T. Cover and P. Hart. "Nearest neighbor pattern classification". In: *IEEE Transactions on Information Theory* 13 (1967).

[13] G. Douzas and F. Bacao. "Geometric SMOTE: Effective oversampling for imbalanced learning through a geometric extension of SMOTE". In: *Information Sciences* 501 (2017).

[14] Y. Freund and R. Schapire. "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting". In: (1995).

[15] S. Gazzah and N. Amara. "New Oversampling Approaches Based on Polynomial Fitting for Imbalanced DataSets". In: *2008 The Eighth IAPR International Workshop on Document Analysis Systems* (2008).

[16] P. Hart. "The condensed nearest neighbor rule (Corresp.)" In: *IEEE Transactions on Information Theory* 14 (1968).

[17] A. B. Hassanat et al. "Stop Oversampling for Class Imbalance Learning: A Critical Review". In: *Digital Object Identifier* (2022).

[18] K. Hattori and M. Takahashi. "A new edited k-nearest neighbor rule in the pattern classification problem". In: *Pattern Recognition* 33 (2000).

[19] G. Hinton. "Connectionist learning procedures". In: *Artificial intelligence* (1989).

[20] X. Jin and J. Han. "K-Means Clustering". In: *Encyclopedia of Machine Learning* (2007).

[21] C. Kassasseya et al. "Validation of a Predictive Score of Acute Chest Syndrome (presev-2 study) in Adults". In: *Blood* 136 (2020). URL: https://ashpublications.org/blood/article/136.

[22] G. Kovacs. "An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets". In: *Applied Soft Computing Journal* 83 (2019).

[23] G. Kovacs. "Smote-variants: a python implementation of 85 minority oversampling techniques". In: *Neurocomputing* (2019).

[24] M. Koziarski and M. Wozniak. "CCR: A combined cleaning and resampling algorithm for imbalanced data classification". In: *International Journal of Applied Mathematics and Computer Science* 27 (2017), pp. 727–736.

[25] H. Lee, J. Kim, and S. Kim. "Gaussian-Based SMOTE Algorithm for Solving Skewed Class Distributions". In: *The International Journal of Fuzzy Logic and Intelligent Systems* (2017).

[26] J. Lee, N. Kim, and J. Lee. "An Over-sampling Technique with Rejection for Imbalanced Class Learning". In: *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication* (2015).

[27] L. Ma and S. Fan. "CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests". In: *BMC Bioinformatics* 18 (2017), p. 169.

[28] I. Mani and I. Zhang. "kNN approach to unbalanced data distributions: a case study involving information extraction". In: *In Proceedings of workshop on learning from imbalanced datasets* (2003).

[29] Menard. “Training and assessing classification rules with imbalanced data”. In: *Data Mining and Knowledge Discovery* (2014).

[30] R. Mohammed, J. Rawashdeh, and M. A. Abdullah. “Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results”. In: *11th International Conference on Information and Communication Systems (ICICS)* (2020).

[31] J. Saez et al. “SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering”. In: *Information Sciences* 291 (2015), pp. 184–203.

[32] I. Tomek. “Two modifications of CNN”. In: *In Systems, Man, and Cybernetics* 6 (1976).

[33] X. Yang, Q. Song, and Y. Wang. “A weighted support vector machine for data classification.” In: *IJPRAI* 21 (2007).

[34] B. Zhou et al. “A quasi-linear SVM combined with assembled SMOTE for imbalanced data classification”. In: *The 2013 International Joint Conference on Neural Networks (IJCNN)* (2013).

Chapter 4

Bridging Image Detection Segmentation and Imbalanced Classification: Exploring the Watershed Theory

Many methods developed to deal with imbalanced datasets have mainly been based on conventional clustering and machine learning techniques, such as K-Means [11]. These methods often treat imbalanced data as numerical entities, where connections between data points are determined purely by the values of their features. However, it is possible to incorporate additional features, such as data point connectivity and spatial relationships, to improve these approaches. By taking spatial localization into account, we can potentially improve oversampling methods and better capture the intrinsic structure of the data. In addition, recent advances have explored the integration of graph-based methodologies, as shown in various articles [22, 23], to address the challenges posed by classification problems. The idea of using this type of approach for imbalanced datasets therefore came to mind. Moreover, the notion of a graph is particularly related to images, since an image can be interpreted as a structured graph. This link has led to the study of image-based techniques, particularly in the field of computer vision, where complex pattern detection techniques - such as object identification - have been widely developed.

For example, the following ideas could be developed:

- **Data augmentation**: We can enrich our training dataset with data augmentation. This involves creating synthetic samples using image segmentation masks, applying various transformations such as rotations, scaling, cropping and introducing noise. This not only balances the dataset, but also introduces a larger number of diverse samples, improving model generalization.
- **Feature extraction or region-based features**: Hierarchical seg-

mentation methods identify distinct regions in images. Feature extraction from these segmented regions can be used directly in algorithmic approaches, or even to complete the dataset with artificially created data points.

The field of mathematical morphology (MM) [18] brings together transformation, graph theory and hierarchies. Within the framework of mathematical morphology, one of the best-known techniques is the watershed transformation. In the following section, we look at the different types of watershed transformation and explore their potential efficacy in dealing with imbalanced datasets. Combining aspects of transformation, graph theory and hierarchy in MM opens the way to innovative solutions that could be beneficial.

4.1 Exploring the Literature of Watershed Transformations

4.1.1 From Watersheds for Images...

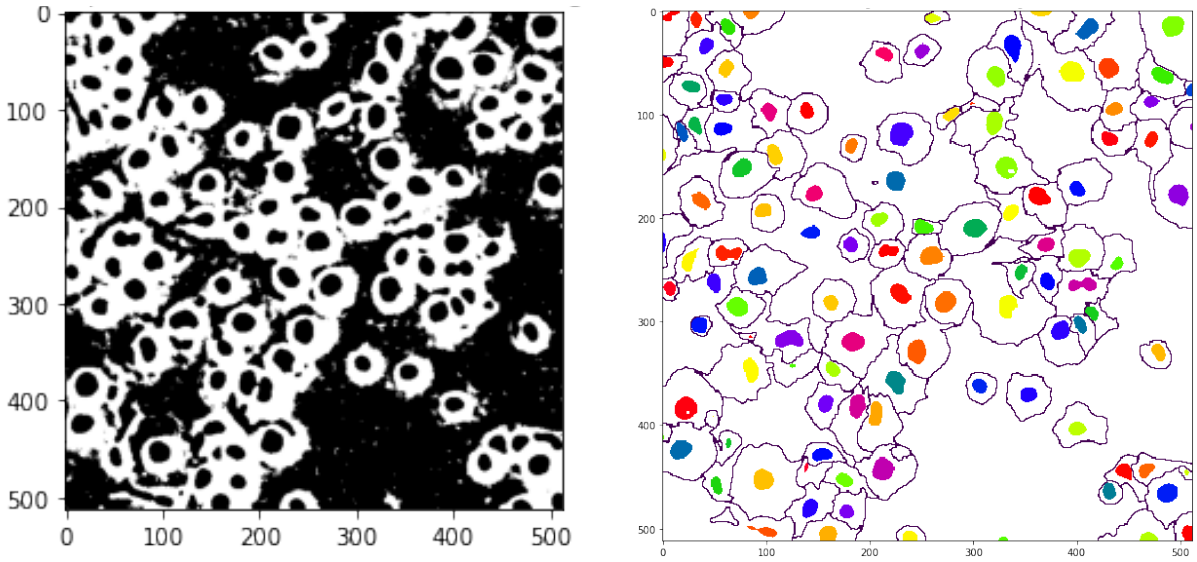
Image segmentation, a concept explained in Meyer's segmentation work [14], involves the task of partitioning an image into distinct regions, each with no overlap, while ensuring that each region is characterized by a uniform attribute. For example, each region may correspond to a specific color in the image. To achieve this partitioning, the image is treated as if it represented a topographic surface. In the context of grayscale images, lighter tones correspond to the highest points of the virtual "landscape", while darker tones represent the lowest altitudes. The lowest regions, similar to the valleys in this landscape, are identified as regional minima. This is where the watershed transformation principle, often referred to as the flooding principle, comes into operation (developed in article [1]). The process proceeds as follows:

1. **Identifying regional minima**: the initial step is to identify regional minima, i.e. the lowest points or valleys on the topographic surface (image).
2. **Flooding process**: Imagine that the topographic surface (image) is submerged in water, with the water level climbing uniformly. Water begins to penetrate through regional minima, leading to the emergence of "floods" from different valleys.
3. **Flood confluence management**: As the flood progresses, it can happen that several floods join together, resulting in confluences. To avoid this problem, fictitious barriers or "dams" are built between the merging floods.

4. **Watershed Functions**: When the flooding phase ends, only the constructed dams (barriers) remain as visible entities on the submerged surface. These dams correspond to what is known as a watershed function. Each watershed, like a dam, designates a catchment area, i.e. a distinct region of the image.

In conclusion, watershed transformation revolves around the concept of submerging the image and observing the natural formation of dams, which in turn delimit different regions. As each watershed symbolizes a distinct area, the application of watershed functions effectively segments the image into these distinct regions.

This technique can also be used for object detection. In this scenario, users must first annotate the image by placing object markers and background markers. Then, openings will be created in the marked regions corresponding to the objects. In simpler terms, the regions identified as objects by the markers will serve as minimum regions. As a result, barriers will be erected between the background and the objects. Figure 4.1 illustrates watershed transformations applied to object detection. In this case, the flooding principle is applied to a free image (accessible via the GitHub repository `/bnsreenu/python_for_microscopists`). The aim here is to identify cell boundaries using cell nuclei as markers.



(a) Original Image. Cells with nucleus.

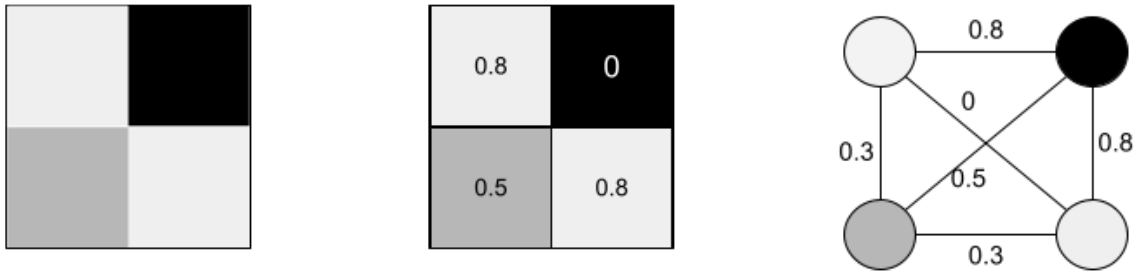
(b) Nucleus markers and cells boundaries.

Figure 4.1: Cell boundaries obtained using watershed transformations after marking each cell with its nucleus (color marks).

The flooding principle and its corresponding algorithm are the best known and most widely used. Nevertheless, watershed transformations can also be seen through the principle of water drop [6]. Both algorithms have certain similarities, but they use distinct approaches to understanding and implementing watershed transformation. Fundamentally, the flooding

principle as explained previously is based on the idea of "flooding" an image whereas the drop of water principle is based on the visualization of how drops of water flow down a topographic landscape (defined by using for example a gradient map) and form basins.

Nevertheless, both principles were first and foremost used for images. The image is perceived in its discrete digital manifestation - a matrix composed of picture elements, commonly referred to as pixels. Each pixel corresponds to color or grayscale information. Consequently, an image can be visualized as an edge-weighted graph (G, W), where the vertices of G correspond to pixels, the edges represent connections between pixels and the weights of the edges W symbolize dissimilarities between pixels. Figure 4.2 shows an image represented as an edge-weighted graph. In this representation, a minimum region is a set of interconnected pixels surrounded by pixels with significantly higher values. In other words, it is a group of vertices surrounded by edges whose weight is higher than that of the edges linked to the interconnected vertices. The edges between two zones of influence correspond to the watershed function and they separate two different regions.



(a) Original Grey-Scale Image

(b) Image in its digital form with grey-scale values.

(c) Graph obtained using the digital form.

Figure 4.2: Representation of a greyscale image in its original form and in its digital form. The resulting graph is an adjacency graph where the vertices correspond to the pixels. The weight of the edges is calculated as the difference between the greyscale values.

4.1.2 To Watersheds for Classification

The drop of water principle enables watershed transformations to be derived through the use of edge-weighted graphs [21, 7]. This principle serves as a link between different types of datasets, between images and digital data. Specifically, for images, vertices are on a 2D pixel grid and edges correspond to a graph with 4 or 8 adjacencies, depending on neighborhood connections. For digital data, vertices correspond to data points and edges represent the relationships between these points, generally based on adjacency. In both cases, edge weights are closely linked to vertex attributes, such as color for images or Euclidean distance between data points for dig-

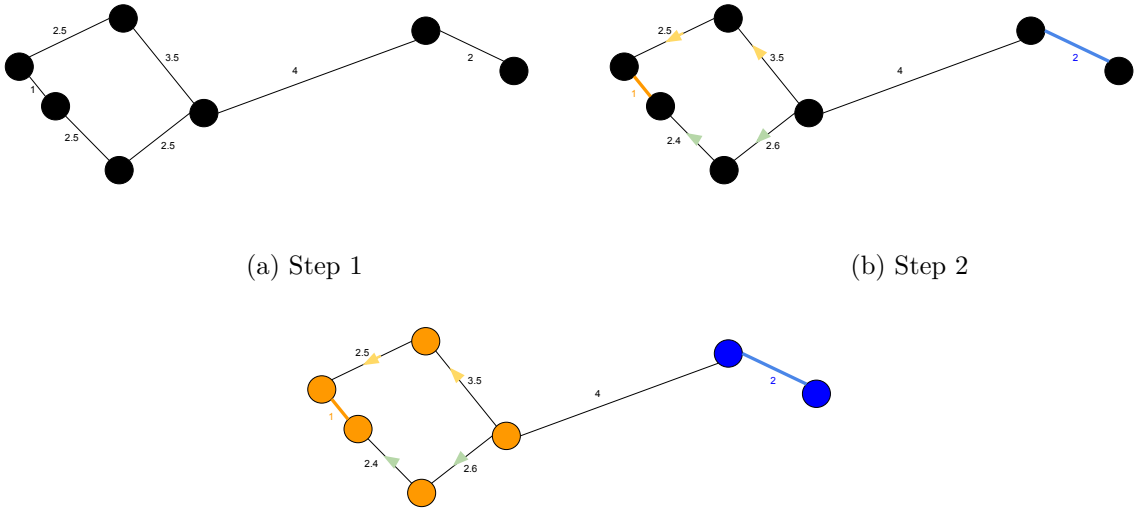
ital data. Thus, assuming that the digital graph is interconnected, the application of the watershed transformation remains consistent and uniform in both cases: images and digital data. This consistency is important, as the watershed transformation, often used for segmentation, can also be exploited for clustering. Therefore, two distinct watershed transformation-based techniques have been designed to address the challenges of clustering in digital dataset: Watershed Cuts [6] and Iterated Watershed [19].

The principle of drop of water allows to obtain watershed transformations by using edge-weighted graphs [21, 7]. S graph can be constructed on images and digital data, this principle buids a bridge between these different types of datasets. In fact, for an image, vertices correspond to a 2D pixel grid and edges to a graph with 4 or 8 adjacencies. Whereas in the case of digital data, vertices correspond to data points and edges to relationship between data points, for example if the data points are neighbours. In both cases, the weight of the edges is related to the criteria of the vertices, e.g. the colour in the case of the image or the Euclidean distance between the data points in the case of the digital data. However, assuming that the digital graph is connected, the watershed transformation can be applied in exactly the same way in both cases. In addition, as watershed transformations is used to perform segmentation, it can be used to perform clustering. Thus, two methods based on watershed transformations have been developed to resolve clustering problems: Watershed Cuts [6] and Iterated Watershed [19]. Similarly, watershed transformations find application in solving classification problems. In this context, the aim is to predict the class to which a new, unlabeled object belongs, based on a given collection of labeled objects. This algorithm has been named "Semi-Supervised Watershed" [2]. The following paragraphs explain the watershed, iterated watershed and semi-supervised watershed sections in greater detail.

4.1.2.1 Watershed Cuts

The Watershed Cuts algorithm is build on the water drop principle. For a better understanding, we'll look at its principles using pseudocode and visual support as described in the article [6]. Essentially, the Watershed Cuts algorithm is based on the concept of flow within an edge-weighted graph. To illustrate this process, let's take a simplified example using a graph of 2 nearest neighbors (Figure 4.3). Each edge of this graph is weighted by the Euclidean distances between points, figure 4.3a. We follow a first drop on a descending path, until a minimum edge is found (yellow arrows in figure 4.3b). If this minimum does not belong to any cluster, as in figure 4.3b, all vertices belonging to this path are marked with the same

labels. We then repeat this operation until all vertices belong to a cluster, as in figure 4.3c.



(a) Step 1

(b) Step 2

(c) Step 3

Figure 4.3: Obtaining two clusters using the Watershed Cut technique from an edge weighted graph of 2 nearest neighbors. The numbers on the edges represent the distance between two vertices, i.e., between two data points.

The algorithm's pseudo-code gives a better understanding of how it works, highlighting the importance of the "flow" aspect. In particular, most of the algorithm revolves around the concept of flow defined through the edge-weighted graph. A "flow" can be represented by the steepest path from one graph node to another, using the lowest weighted value on the different edges. This concept is related to the water drop principle, with each water drop flowing towards lower altitudes. The pseudo code of the algorithm is described in article [5] and can be summarized as follows:

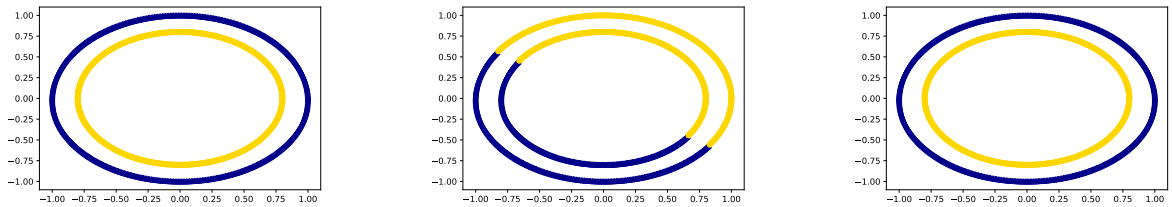
Algorithm 2: Watershed Cut

```
Data: An edge-weighted graph G ;
Result: A clustering partitioning of the data points
for Each node in the graph do
    Define the node as unlabelled ;
for Each unlabelled node in the graph x do
    Find a flow such that x is a top of the flow;
    Return -1 or the label of the flow;
    if -1 is returned then
        All the data points containing in this flow are labelled with a
         novel label;
    else
        All the data points containing in the flow are labelled with
         the label of the flow ;
```

4.1.2.2 Iterated Watershed

Watershed-based clustering is not limited to the Watershed Cuts approach. Another method known as Iterated Watersheds [19] also exploits watershed transformations for clustering. Although very similar to the K-Means [11] technique, the Iterated Watersheds method differs in its emphasis on preserving connectivity between data points. K-Means clustering and the iterated watershed algorithm share a two-stage structure and even the same input parameter: “k”, which designates the desired number of clusters. Here’s where the distinction comes in: both methodologies start with what’s known as the **maximization step**. In this stage, each data point is assigned to one of the “k” centers. In the K-Means method, a point is connected to the nearest center as a function of distance. In contrast, the iterated watershed algorithm operates on a path-based approach. A data point is connected to a center only if there is a minimum path connecting the two (the data point and the center) according to a designated function, often a distance function. Iterated Watershed incorporates the IFT [8] algorithm to perform this connectivity-maximization step. This distinctive approach enables Iterated Watersheds to take connectivity into account. Having completed the maximization step, which results in a partition of data points, the second step - known as the **expectation step** - follows. In this phase, new centers are calculated for each cluster formed by the partition. This process results in the creation of ”k” new centers. The maximization and expectation steps are then iteratively repeated until convergence is achieved.

To visually illustrate of K-Means and iterated clustering algorithms, figure 4.4 presents a side-by-side illustration of their application on a toy example. This comparison highlights the differences and results achieved by both techniques.



(a) Toy example: two circles.

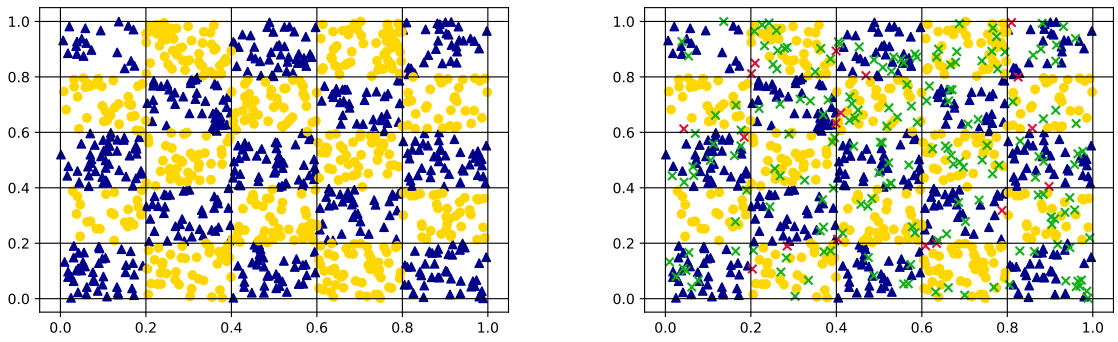
(b) Clusters obtained using KMeans.

(c) Clusters obtained using Iterated Watershed.

Figure 4.4: Clusters obtained on a toy example using two clustering methods: KMeans and Iterated Watershed.

4.1.2.3 Watershed for Semi-Supervised Classification

Watershed-based methodologies can also be extended to semi-supervised classification. Figure 4.5 illustrates the Watershed Semi-Supervised algorithm's application on a balanced chessboard dataset 2.1b.



(a) Balanced ChessBoard.

(b) Predictions on a balanced chessboard using Watershed for Semi-Supervised Classification algorithm.

Figure 4.5: Prediction using Watershed Semi-Supervised algorithm on the chessboard example. The blue triangles represent the majority class in the training part, the yellow circles represent the minority class in the training part, while the crosses belong to the testing part. The green crosses represent correctly predicted test data points, while the red crosses represent incorrectly predicted test data points.

The algorithm developed in the paper [2] is based on the principle of the Support Vector Machine (SVM) [15]. As explained in Figure 2.4 in Chapter 2, the objective of the SVM is to partition the data space into areas containing data points of the same class. To achieve this, the SVM algorithm seeks to define a hyperplane that maximizes the distance from

support vectors while optimizing prediction accuracy, avoiding instances lying on the wrong side of the hyperplane. This challenge is referred to as maximum margin partitioning. Notably, the paper [2] establishes that the Morph Median partition, defined by a watershed algorithm, consistently maximizes the maximum margin partition. The main difference between SVMs and semi-supervised watersheds lies in connectivity due to the use of graphs.

Watershed for Semi-Classification operates through an edge-weighted graph, where data points must be labeled with classes or as "none" if they don't belong to any class. This approach is rooted in the Minimum Spanning Forest Watershed algorithm (from the article [6]). The pseudo-code, detailed in the paper [2], outlines the following steps (Algorithm 3):

Algorithm 3: Semi-Supervised Watershed

Data: An edge-weighted graph G, with labelled seeds S.
Result: A clustering of the graph nodes

```
1 Sort the graph vertices by using theirs weights ;
2 for Each sorted vertex (e_x, e_y) in the graph do
3     if both e_x and e_y are labelled then
4         pass;
5     else
6         Assign the same label to e_x and e_y;
```

This algorithm merges labeled and unlabeled data points, enabling a maximum-margin partition to be defined. The article [2] establishes a direct link between watershed algorithms and SVM algorithms, highlighting the efficiency of the Watershed Semi-Supervised algorithm.

4.2 Watershed & Hierarchies

Most of the watershed transformation methods have been developed in the domain of graphs and edge weights. Consequently, adapting these algorithms to imbalanced classification algorithms requires modifications centered on Euclidean distance, spatial location and connectivity considerations. However, the watershed transformation framework introduces another point of view thanks to its link with hierarchies. This perspective opens the way for experimenting with various distance functions, for example ultrametric distances, and exploring the concatenation of regions through hierarchical structures.

4.2.1 Ultrametric Distance

4.2.1.1 Definition of Graph and Minimum Spanning Tree

A graph is noted G = (V,E), where V is a finite set and E is a set of pairs of distinct elements of V, i.e., $E \subseteq \{\{x, y\} \subseteq V | x \neq y\}$. Each element of V is called a vertex (of G), and each element of E is called an edge (of G). A sequence $\pi = (x_0, ..., x_n)$ of elements of a set X is a path (in X) from x_0 to x_n if x_{i-1}, x_i is an edge of G for any $i \in \{1, ..., n\}$. In addition, if w is a map from the edge set of G to the set $\mathbb{R}$ of real numbers, then the pair (G,w) is called an (edge) weighted graph, show figure 4.2.If (G,w) is a weighted graph, for any edge u of G, the value w(u) is called the weight of u (for w).

A hierarchy is denoted H and can be represented using an edge-weighted graph. Each node in the graph represents a leaf in the hierarchy and the edges between the nodes represent the level of the hierarchy. An example of a graph and hierarchy is shown in Figure 4.6.

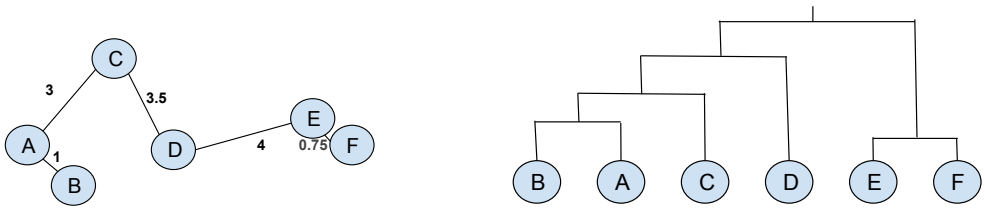


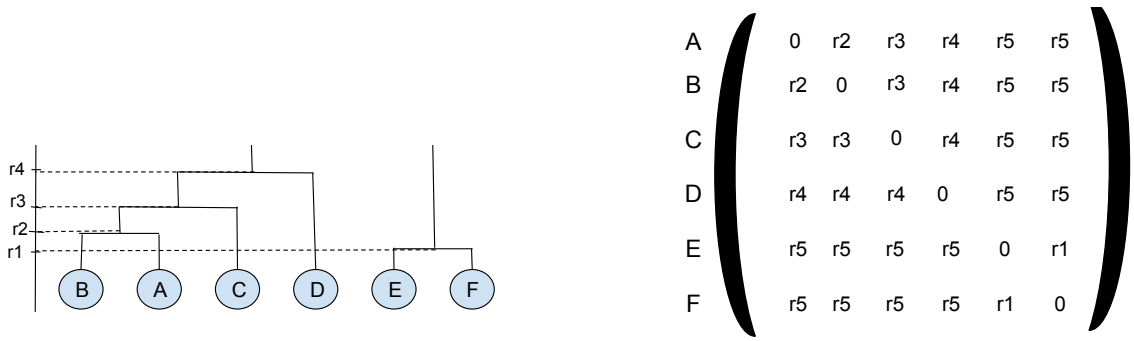
Figure 4.6: Example of edge-weighted graph using euclidean distance and one of its tree/hierarchy/dendrogram.

One particular hierarchy is called a Minimum Spanning Tree (MST) and as been presented in article [9]. An MST is hierarchy also call a tree that spans or covers all the nodes in the original graph while minimizing the total edge weight. Due to its definition one key property of the MST is its connectivity. In fact, the MST connects all vertices of the original graph, ensuring that there is a path between any pair of nodes within the tree. The other main property of an MST is to minimize the sum of the edge weights. Each edge in the hierarchy is assigned a weight, and the sum of these weights is as small as possible while still ensuring connectivity. Several algorithms can be used to construct a MST. Two well-known methods are Prim's algorithm [16] and Kruskal's algorithm [13].

4.2.1.2 Definition of Ultrametric and Ultrametric fitting

Every edge-weighted graph has a MST and, therefore, every edge-weighted graph can be considered as a hierarchy or tree [5]. In addition, in article [12], a distance called ultrametric distance has been introduced. The

ultrametric distance is a function $d : V * V \rightarrow \mathbb{R}$ where the triangle inequality is replaced by the ultrametric one: $(\forall(x, y, z) \in V^3)$, $d(x, y) \leq \max d(x, z), d(z, y)$. But, the key concept in the construction of ultrametric distances is the two-way relationship between ultrametric distances and hierarchical structures. Ultrametric distances can be used to construct hierarchical representations and, conversely, hierarchical structures can be expressed by ultrametric distances. Given a set of data points and their ultrametric distances, you can build a hierarchy. The process usually begins by creating a cluster for each data point. The clusters are then successively merged on the basis of the ultrametric distances between them. As distances increase, clusters are merged at higher levels in the hierarchy. Conversely, given a hierarchical structure, you can calculate ultrametric distances that capture the hierarchical relationships within the data. To do this, you start with the hierarchy and define the ultrametric distance between two data points as the height at which their respective clusters merge in the hierarchy. Figure 4.7 shows the ultrametric distances and an MST.



(a) Tree/Dendrogram (b) Ultrametric

Figure 4.7: Example of tree and its ultrametric distances, inspired from article [3]

By employing the ultrametric distance, an optimization problem known as ultrametric fitting can be formulated. In fact, as explained in article [3], the computation of a subdominant ultrametric leads to the establishment of a constrained optimization problem. The foundation of the subdominant ultrametric relies on a min/max operator, as the ultrametric distance can also be defined as [3]:

$$\forall(x, y) \in V^2, d_u(x, y) = \min_{P \in P_{xy}} \max_{e \in P} w(e) \tag{4.1}$$

where P_{xy} denotes the set of all paths between the vertices x and y of G.

By formulating a loss function, we can address this optimization problem and consequently identify the most suitable ultrametric distance for a given

dataset and thus used this ultrametric fitting method as a clustering one or a classification one.

4.2.2 Watershed Hierarchies

Exploring the notion of ultrametric fitting brought the fact that any graph can be considered as a hierarchy. In the previous case, we mentioned MSTs, but there are many other hierarchies such as partition hierarchies or minimum hierarchies. The article [4] links and explains these different hierarchies. One of them is called watershed hierarchies and extend the concept of watershed transformations by incorporating hierarchical structures. In a traditional watershed transformation, regions are partitioned based on gradients or euclidean distance. However, these methods might oversimplify complex structures and fail to capture some detail. Watershed hierarchies overcome this limitation by taking into account weighted markers where the importance of a marker can be given by extinction values [20] such as the dynamics [10]. Extinction value of an object or region indicates the level at which it is merged with another region during the hierarchy construction process. Essentially, it represents the difference in attributes or characteristics that initiates the merging of two adjacent regions. Where as the dynamics measure a contrast. The dynamic of a path is the difference in altitude between the points of the highest and lowest altitude of this path.

Thus the hierarchical watershed extends the concept by considering several levels of flooding. For example, instead of stopping the flooding process at a single level, it continues to flood the image at different levels of intensity by using the information implied by markers. The idea of modifying the hierarchy using a marker associated with the data is particularly interesting, especially in cases where the data is imbalanced. Specific markers can be strategically defined to solve the problem of imbalance. For example, in the image field, where small objects are disproportionate to the background, markers can be used to prioritise the segmentation of these objects. In doing so, the hierarchical watershed can ensure that objects are correctly delineated, even if they are smaller than the background. This approach could be used by directly modifying the watershed algorithm to better handle imbalanced datasets.

4.3 Watershed and Imbalanced Dataset

After exploring the existing literature on watershed transformations and recognizing their potential for dealing with imbalanced data, we are going to analyze how these algorithms can positively contribute to solving

imbalanced classification problems. As we have seen in previous sections, existing imbalanced methods, often involve clustering and classification techniques. Therefore, our focus will be on evaluating the efficiency of watershed-based methods in this context. Specifically, we will examine the potential of semi-supervised watershed, watershed cut and iterated watershed methods. By studying these approaches, we aim to determine their applicability and benefits in improving classification results for imbalanced dataset.

4.3.1 Semi-Supervised Watershed for Imbalanced Data

To begin our investigation, we launched a comparative study between usual machine learning methods and the semi-supervised Watershed algorithm. We focused on evaluating their performance on randomly selected imbalanced datasets. In particular, we created a set of six imbalanced datasets, as shown in table 2.1. The main objective was to evaluate the prediction scores generated by different classifiers in this context. The classifiers studied included Random Forest, Support Vector Machine (SVM), Adaboost, k-Nearest Neighbors (KNN) and the Semi-Supervised Watershed algorithm. Given the imbalanced nature of the datasets, our evaluation focused on the performance measures described in chapter 2, which include precision, recall, F1 score, AUC, geometric mean (G-mean) and negative predictive value (NPV). Using a 10-fold cross-validation strategy, we divided the datasets into training and test sets, the latter representing 25% of the dataset. The results of this comparative analysis are presented in Figures 4.1, 4.2, 4.3, 4.4, 4.5, and 4.6. These figures illustrate the performance of each classifier across the different evaluation metrics.

Precision	**Paw**	**Pima**	**Segment0**	**Vowel0**	**Yeast1**	**Glass1**
RF	60,1 ± 6,6	55,6 ± 4,8	97,9 ± 1,3	96,96 ± 4,1	73,1 ±20,3	71,5 ± 10,5
SVC	63,7 ± 25	64,6 ± 9,7	97,8 ± 1,7	94,3 ± 3,2	75,3 ± 13,2	69,8 ± 12,7
KNN	63,9 ± 12,7	57 ± 5,8	91,5 ± 1,9	87,5 ± 6,1	65,3± 12,8	66 ± 12,5
AdaBoost	60,2 ± 6,6	55,6 ± 4,8	97,9 ± 1,3	96,6 ± 4,1	73,1 ± 20,3	71,5 ± 10,5
Semi-Supervised Watershed	55,4 ± 8,4	40,7 ± 5,7	66,2 ± 4,0	91,5 ± 3,6	48,2 ± 8,4	43,2 ± 19,6

Table 4.1: Precision scores obtained for comparing Semi-Supervised Watershed with usual Machine Learning methods on Imbalanced datasets. Highlight in blue the best score obtained for a given dataset.

Recall	Paw	Pima	Segment0	Vowel0	Yeast1	Glass1
RF	42,4 ± 9,7	48,5 ± 4,8	88,9 ± 3,2	92,5 ± 4,0	30,8 ± 13,8	44,7 ± 12,3
SVM	20,4 ± 10,3	36,6 ± 4	88,8 ± 4,7	94,2 ± 4,5	37,7 ± 7,3	34,2 ± 6,8
KNN	34,4 ± 9,0	43 ± 5,4	80,9 ± 5,2	92,9 ± 4,9	32,3 ± 4,6	50,0 ± 9,2
AdaBoost	26,0 ± 9	36,9 ± 5	90,1 ± 2	89,8 ± 4,5	40 ± 11,3	41,1 ± 11
Semi-Supervised Watershed	48 ± 7,8	38,4 ± 5,7	81,2 ± 5,3	98,2 ± 2,2	48,5 ± 9,1	60 ± 11,8

Table 4.2: Recall scores obtained for comparing Semi-Supervised Watershed with usual Machine Learning methods on Imbalanced datasets. Highlight in blue the best score obtained for a given dataset.

F1	Paw	Pima	Segment0	Vowel0	Yeast1	Glass1
RF	49 ± 7,2	51,6 ± 3,7	93,1 ± 1,8	94,4 ± 2,8	41,6 ± 15	53,6 ± 8,8
SVM	29,5 ± 12,5	46,1 ± 2,3	93 ± 2,3	94,2 ± 2,4	49,5 ± 7	45 ± 5,4
KNN	43,7 ± 7,9	48,7 ± 4,2	85,7 ± 2,8	90 ± 3,9	42,5 ± 4,7	55,4 ± 5,8
AdaBoost	31,2 ± 9,3	44,8 ± 4,6	92,1 ± 1,5	92,7 ± 3,3	46,2 ± 8,4	48,9 ± 7,2
Semi-Supervised Watershed	50,8±5,6	39,4 ± 5,4	72,8 ± 3,8	94,7 ± 2,3	47,8 ± 7,2	47,6 ± 8,2

Table 4.3: F1 scores obtained for comparing Semi-Supervised Watershed with usual Machine Learning methods on Imbalanced datasets. Highlight in blue the best score obtained for a given dataset.

AUC	Paw	Pima	Segment0	Vowel0	Yeast1	Glass1
RF	4,1 ± 1,4	21 ± 4,3	0,3 ± 0,2	0,4 ± 0,4	1,3 ± 1,4	10,6 ± 5
SVM	1,5 ±1,6	11,8 ± 5,6	0,3 ± 0,3	0,6 ± 0,3	1,5 ± 1	9,1 ± 5,5
KNN	3,0 ±1,5	17,8 ± 4,3	1,3 ± 0,3	1,4 ± 0,7	2,2 ± 1,5	16,6 ± 11,7
AdaBoost	5,4 ± 1,7	14,6 ± 4,6	0,9 ± 0,2	0,4 ± 0,4	3,4 ± 1,5	13,4 ± 9,8
Semi-Supervised Watershed	5,8 ± 1,9	30,1 ± 4,5	6,9 ± 1,2	0,9 ± 0,4	5,9 ± 1,8	52 ± 20,6

Table 4.4: AUC scores obtained for comparing Semi-Supervised Watershed with usual Machine Learning methods on Imbalanced datasets. Highlight in blue the best score obtained for a given dataset.

Gmean	Paw	Pima	Segment0	Vowel0	Yeast1	Glass1
RF	63,3 ± 7,2	61,8 ± 2,9	94,1 ± 1,7	96 ± 2,1	53,4 ± 13,2	62,5 ± 7,4
SVM	41,7 ± 16	56,6 ± 1,9	94 ± 2,4	96,8 ± 2,3	60,7 ± 5,6	55,4 ± 4,5
KNN	57,2 ± 7,3	59,3 ± 3,4	89,3 ± 2,8	95,7 ± 2,5	56,1 ± 3,9	63,8 ± 4,6
AdaBoost	48,7 ± 8,9	55,9 ± 3,8	94,5 ± 1,1	94,5 ± 2,4	61,5 ± 8,4	58,7 ± 5,8
Semi-Supervised Watershed	67 ± 5,3	51,6 ± 4,6	86,9 ± 2,9	98,6 ± 1,1	67,2 ± 6,4	51,9 ± 8,5

Table 4.5: GMean scores obtained for comparing Semi-Supervised Watershed with usual Machine Learning methods on Imbalanced datasets. Highlight in blue the best score obtained for a given dataset.

NPV	Paw	Pima	Segment0	Vowel0	Yeast1	Glass1
RF	92,1 ± 1,2	74,1 ± 1,7	98,2 ± 0,2	99,2 ± 0,4	92,9 ± 1,3	75,2 ± 3,7
SVM	89,7 ± 1,1	72,2 ± 0,8	98,2 ± 0,7	99,4 ± 0,4	93,5 ± 0,7	71,8 ± 1,7
KNN	91,2 ± 1,1	72,9 ± 1,9	96,9 ± 0,8	99,3 ± 0,85	93 ± 0,4	75,5 ± 2,5
AdaBoost	90 ± 1,1	71,6 ± 1,6	98,4 ± 0,3	99,0 ± 0,4	93,7 ± 1,1	73,2 ± 2,5
Semi-Supervised Watershed	92,1 ± 1,2	74,1 ± 1,7	98,2 ± 0,5	99,2 ± 0,4	92,9 ± 1,3	75,2 ± 3,7

Table 4.6: NPV scores obtained for comparing Semi-Supervised Watershed with usual Machine Learning methods on Imbalanced datasets. Highlight in blue the best score obtained for a given dataset.

Our analysis reveals a remarkable observation: the semi-supervised Watershed algorithm improves prediction performance for specific evaluation metrics. In particular, we consistently note that the AUC score tends to be higher for the semi-supervised Watershed algorithm than for other conventional machine learning methods. While it is important to recognize that the semi-supervised Watershed algorithm is not perfect, these results suggest that the incorporation of connectivity information can indeed improve prediction results on imbalanced dataset. The semi-supervised watershed algorithm appears to be a good candidate for algorithm modification and cost-sensitive learning methods. Nevertheless, with regard to imbalanced data, it should be noted that the most common techniques for addressing such imbalances often involve sampling methods. Given that watershed methods can also be applied to clustering tasks, it is of interest to investigate the behavior of Watershed Cuts and Iterated Watershed algorithms in the context of imbalanced data. In fact, those methods could then be used to find minority data clusters in a unsupervised manner and then used to generate new data points by taking care of the data distribution. The motivation for this exploration lies in the potential transformation of these algorithms into oversampling algorithms.

4.3.2 Watershed Cuts and Iterated Watershed for Imbalanced Data

We perform a comparative analysis of the Watershed Cuts and Iterated Watershed methods with the clustering methods commonly used in oversampling techniques: KMeans [11] and DBSCAN [17]. To facilitate this examination, we use the same set of 6 imbalanced datasets as before. The aim is to compare the prediction performance of these clustering methods. Given that clustering approaches operate in an unsupervised manner, we establish a procedure for generating predictions using these methods. This procedure consists of two steps. First, we apply clustering to the training and testing part. Next, we assume that all data points in the same cluster belong to the same class. This allows us to assign labels to unlabeled

data points, i.e. test data points, on the basis of the majority label within the cluster to which they belong. By employing this methodology, we can effectively compare the various clustering techniques. The results of this comparative analysis are presented in tables 4.7, 4.8, 4.9, 4.10, 4.11 and 4.12. These tables summarize the prediction scores of each method according to different evaluation metrics.

Precision	Paw	Pima	Segment0	Vowel0	Yeast1	Glass1
Kmeans	63,7 ± 16,5	43,2 ± 5,2	84,2 ± 3,9	87,7 ± 10,1	49,5 ± 11,5	48,3 ± 7,6
DBSCAN	26,6 ± 7	38,1 ± 3,6	41,3 ± 6,5	50,9 ± 14,3	53,2 ± 26,5	59,5 ± 13,7
Watershed Cuts	34 ± 4,2	54,5 ± 17	75 ± 1,4	68,4 ± 9,8	42,8 ± 11,4	61 ± 37,3
Iterated Watershed	40,8 ± 10,5	42,1 ± 1,2	83,1 ± 3,8	33,1 ± 5	36,4 ± 15,2	42,7 ± 7,1

Table 4.7: Precision scores obtained for comparing Watershed Cuts and Iterated Watershed with usual clustering methods on Imbalanced datasets. Highlight in blue the best score obtained for a given dataset.

Recall	Paw	Pima	Segment0	Vowel0	Yeast1	Glass1
Kmeans	42,8 ± 7,6	44,8 ± 0	73,2 ± 1,2	71,7 ± 10,1	43,1 ± 11,5	46,3 ± 8,4
DBSCAN	29,2 ± 6,2	26,1 ± 3,7	23,2 ± 3,7	33,6 ± 6,8	14,6 ± 6,4	31,1 ± 9,8
Watershed Cuts	58,0 ± 7,8	6,0 ± 1,5	87,2 ± 7,9	89,9 ± 7,6	43,1 ± 9,9	6,3 ± 4,6
Iterated Watershed	44,0 ± 10,0	46,3 ± 6	60,4 ± 4,3	71,8 ± 10,6	25,4 ± 13,8	39,5 ± 12,1

Table 4.8: Recall scores obtained for comparing Watershed Cuts and Iterated Watershed with usual clustering methods on Imbalanced datasets. Highlight in blue the best score obtained for a given dataset.

F1	Paw	Pima	Segment0	Vowel0	Yeast1	Glass1
Kmeans	50,2 ± 7,5	43,8 ± 2,7	78,2 ± 1	78,1 ± 7,3	45,5 ± 10,2	47 ± 7,6
DBSCAN	27,6 ± 6,3	30,6 ± 1,4	29,7 ± 4,7	39,5 ± 6,8	21,9 ± 8,8	40 ± 11
Watershed Cuts	42,5 ± 3,9	10,8 ± 2,8	80,4 ± 2,6	77,3 ± 7,9	42,7 ± 10,1	10,9 ± 7,3
Iterated Watershed	41,8 ± 9	43,9 ± 3,3	69,9 ± 4,2	45,2 ± 6,6	29,4 ± 13,9	40,6 ± 9,6

Table 4.9: F1 scores obtained for comparing Watershed Cuts and Iterated Watershed with usual clustering methods on Imbalanced datasets. Highlight in blue the best score obtained for a given dataset.

AUC	Paw	Pima	Segment0	Vowel0	Yeast1	Glass1
Kmeans	6 ± 2,6	32,4 ± 6,8	2,3 ± 0,7	1,2 ± 1,1	5,0 ± 1,7	27,1 ± 6,3
DBSCAN	16 ± 1,1	23,6 ± 6,8	5,5 ± 0,6	3,7 ± 2	1,7 ± 1,5	12 ± 6,5
Watershed Cuts	14,3 ± 0,6	2,8 ± 1,2	4,8 ± 0,8	4,4 ± 1,8	6,5 ± 1,8	2,6 ± 2,7
Iterated Watershed	7,7 ± 0,9	34 ± 2,8	2 ± 0,4	14,8 ± 2	4,3 ± 1,3	37,7 ± 9,9

Table 4.10: AUC scores obtained for comparing Watershed Cuts and Iterated Watershed with usual clustering methods on Imbalanced datasets. Highlight in blue the best score obtained for a given dataset.

Gmean	Paw	Pima	Segment0	Vowel0	Yeast1	Glass1
Kmeans	63,8 ± 5,5	54,9 ± 2,8	84,5 ± 0,4	84 ± 6	63,4 ± 8,8	56,4 ± 6,8
DBSCAN	50,4 ± 5,7	44,4 ± 1,2	46,7 ± 3,9	56,6 ± 5,6	35,6 ± 12,9	54 ± 7,1
Watershed Cuts	69,4 ± 4	23,9 ± 3,2	91 ± 3,8	92,6 ± 4,2	63 ± 8	16 ± 13,9
Iterated Watershed	62,6 ± 7,4	55,1 ± 2,4	76,9 ± 2,9	78,0 ± 6,3	45,8 ± 18	47,4 ± 6,4

Table 4.11: GMean scores obtained for comparing Watershed Cuts and Iterated Watershed with usual clustering methods on Imbalanced datasets. Highlight in blue the best score obtained for a given dataset.

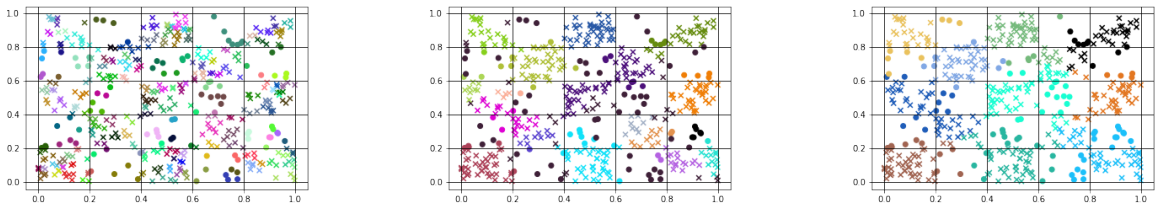
NPV	Paw	Pima	Segment0	Vowel0	Yeast1	Glass1
Kmeans	92,2 ± 1	69,4 ± 2,1	95,6 ± 0,2	97,2 ± 1	93,9 ± 1,2	71,2 ± 4,2
DBSCAN	89,7 ± 1	65,8 ± 0,9	88,1 ± 0,6	93,5 ± 0,6	91,3 ± 0,5	71,2 ± 2,8
Watershed Cuts	93,3 ± 1	65,9 ± 0,6	97,8 ± 1,3	98,9 ± 0,8	93,8 ± 1,1	65,1 ± 0,9
Iterated Watershed	91,9 ± 1,4	69,7 ± 1,5	93,7 ± 0,7	96,8 ± 1,2	92,2 ± 1,3	64,9 ± 3,6

Table 4.12: NPV scores obtained for comparing Watershed Cuts and Iterated Watershed with usual clustering methods on Imbalanced datasets. Highlight in blue the best score obtained for a given dataset.

We find that the Watershed Cuts method improves predictions for some specific scores. For example, most of the time, for recall or NPV scores, the Watershed Cuts method scores higher than the other usual clustering methods. However, the iterated watershed method has less impact. Except in special cases, the results of the iterated watershed method can be compared with those of KMeans. We therefore decided to consider only Watershed Cuts as an interesting method for developing a new oversampling method. In addition, we decided to compare the clusters obtained by the Watershed Cuts method with those obtained by the KMeans and DBSCAN methods. In fact, we want to understand whether, apart from connectivity aspects, there are other differences between these clustering methods. To do this, we visualized the clusters obtained by the Watershed Cuts, KMeans and DBSCAN methods on 2D datasets. For example, we obtained the figure 4.8 on chessboard examples 2.1. In this and all other visualizations, we observe that Watershed Cuts builds very small clusters compared to those obtained with KMeans and DBSCAN. Another important thing to note is that the Watershed Cuts parameter, the number of neighbor in the graph, has no influence on the shape of the clusters. This makes the watershed clusters very stable compared to those obtained with the DBSCSAN or KMeans methods.

4.4 Conclusion

Exploring the various aspects that watershed transformations can incarnate seems to present significant utility when it comes to predicting imbalanced data and potentially helping in the detection of ACS in SCD



(a) Watershed Cuts Clustering: *n_nei*:10

(b) DBSCAN Clustering: *eps*:0.05, *min_s*:5

(c) KMeans Clustering: *nb_clust*:50

Figure 4.8: Imbalanced Chessboard clustered by different methods: Watershed Cuts, DBSCAN and KMeans. Crosses correspond to majority data points, whereas circles correspond to minority ones.

patients. The flexibility of watershed methods becomes evident as they can function both as oversampling techniques, using Watershed Cut, but also as an algorithmic level thanks to the hierarchies. Furthermore, our analysis has demonstrated that Watershed Cut is an effective strategy for dealing with imbalanced datasets. This is attributed not only to the distinct shape of the clusters it generates, but also to the favorable prediction scores it obtains directly on datasets compared to other clustering methods. With this in mind, we first focus on designing a new oversampling method called WSSMOTE, based on the principle of Watershed Cuts.

References

[1] S. Beucher. "The Watershed Transformation applied to image segmentation". In: *Signal and Image Processing in Microscopy and Microanalysis* 6 (1992).

[2] A. Challa et al. "Watersheds for Semi-Supervised Classification". In: *IEEE Signal Processing Letters, Institute of Electrical and Electronics Engineers* (2019).

[3] G. Chierchia and B. Perret. "Ultrametric Fitting by Gradient Descent". In: *Neural Information Processing Systems* (2019).

[4] J. Cousty, L. Najman, and B. Perret. "Constructive Links between Some Morphological Hierarchies on Edge-Weighted Graphs". In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing* 7883 (2013).

[5] J. Cousty et al. "Watershed Cuts: Minimum Spanning Forests and the Drop of Water Principle". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2009).

[6] J. Cousty et al. "Watershed Cuts: Minimum Spanning Forests and the Drop of Water Principle". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31 (2009).

[7] J. Cousty et al. "Watershed Cuts: Thinnings, Shortest Path Forests, and Topological Watersheds". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (2010).

[8] A. X. Falcao, J. Stolfi, and R. de Alencar Lotufo. “The Image Foresting Transform: Theory, Algorithms and Applications”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (2004).

[9] J. C. Gower and G. J. S. Ross. “Minimum Spanning Trees and Single Linkage Cluster Analysis”. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 18 (1969), pp. 54–64.

[10] M. Grimaud. “New measure of contrast: the dynamics”. In: *Optics and Photonics* (1992).

[11] X. Jin and J. Han. “K-Means Clustering”. In: *Encyclopedia of Machine Learning* (2007).

[12] M. Krasner. “Nombres semi-réels et espaces ultramétriques”. In: *Comptes rendus hebdomadaires des séances de l’Académie des sciences* 219 (1944), pp. 433–435.

[13] J. B. Kruskal. “On the shortest spanning subtree of a graph and the traveling salesman problem”. In: *Proceedings of the American Mathematical Society* 7 (1956).

[14] F. Meyer and S. Beucher. “Morphological Segmentation”. In: *Journal of Visual Communication and Image Representation* 1 (1990).

[15] J. C. Platt. “Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods”. In: *Advances in Lage margin Classifiers* (1999).

[16] R.C. Prim. “Shortest connection networks And some generalizations”. In: *Bell System Technical Journal* 36 (1957).

[17] E. Schubert et al. “DBSCAN revisited, revisited: why and how you should (still) use DBSCAN”. In: *ACM Transactions on Database Systems (TODS)* 42.3 (2017), pp. 1–21.

[18] J. Serra. *Image Analysis and Mathematical Morphology*. 1982.

[19] S. Soor et al. “Iterated Watersheds, A Connected Variation of K-Means for Clustering GIS Data”. In: *IEEE Transactions on Emerging Topics in Computing* 9 (2021).

[20] C. Vachier and F. Meyer. “Extinction value: A new Measurement of persistence”. In: *IEEE Workshop on NonLinear Signal and Image Processing* (1995).

[21] L. Vincent and P. Soille. “Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations”. In: *IEEE Transactions on Pattern Analysis and Machine Learning* 13 (1991).

[22] Z. Wu et al. “A Comprehensive Survey on Graph Neural Network”. In: *IEEE Transactions on Neural Network* 32 (2020).

[23] T. Zhao et al. “Graph Data Augmentation for Graph Machine Learning: A survey”. In: *ACM SIGKDD Explorations Newsletter* 24 (2022).

Chapter 5

WSSMOTE: a Novel Oversampling Method

The Watershed Cuts method appears to be a promising way of dealing with imbalanced datasets, due to its distinctive cluster shapes and favorable performance directly on the datasets compared to other clustering methods. Given that oversampling methods are the most widely used approaches for dealing with imbalanced datasets, and that they have produced abundant results, we opted for a novel oversampling strategy. We developed a new oversampling method called "WSSMOTE", based on the principles of Watershed Cuts. In this chapter, we will look at the details of the algorithm's implementation and its implications for the detection of ACS. We will explain in detail how the WSSMOTE method was developed and integrated into the pipeline. In addition, we will discuss the effects and tangible results that the WSSMOTE method has had on improving the PPV and performance of ACS detection.

5.1 Algorithm Implementation

Our novel method called WSSMOTE is an oversampling method rooted in the watershed clustering algorithm. Using results derived from other oversampling methods, WSSMOTE has devised two distinct strategies. The first approach takes advantage of the watershed clustering algorithm's ability to generate compact, data-driven micro-clusters. This precision in defining micro-clusters limits the production of minority class data points within majority clusters. By contrast, the second strategy draws on the results presented in the article [5]. This article suggests that clustering-based oversampling methods are more efficient when they incorporate extended clusters. These larger clusters facilitate the addition of new data points at significant distances from each other. Consequently, formulating a methodology for merging clusters into larger clusters, often referred to as "super-clusters", is a key task. Both strategies are explained in more

detail:

- **Watershed Clustering Strategy**: Watershed clustering strategy: This strategy was developed in response to the challenges described in Figure 2.3, particularly with regard to the problems of disjunction (Figure 2.3a) and overlap (Figure 2.3c). In datasets where fidelity to the data distribution is essential, it is imperative to define small clusters. This ensures that minority data points are not added at too large a distance, preventing their generation within majority clusters. To achieve this, we use watershed clustering to define compact regions, and then generate data points within these regions. Building on the polynomial fit interpolation instances demonstrated by Gazzah in article [3], two schemes for incorporating minority data points emerge. The first is to use two random data points from the clusters ("mesh" option). The second involves using the average data points from the clusters plus another random data point from the same cluster ("star" option). These options produce distinct distributions of the final data.

- **Super-Cluster Strategy**: As explained for the first strategy, it has been observed that the clusters formed by the watershed algorithm are generally small, which can lead to insufficient information in some scenarios. Take the example of figure 2.3b, where each of the three red triangles on the left is treated as a separate group according to the watershed algorithm. Consequently, applying the watershed clustering strategy results in three isolated groups, which amounts to duplicating the red triangles without enriching the information. In contrast, another approach is to merge these three groups into a single one before generating new minority data points. This approach improves the capture of crucial information and, consequently, future predictions, a notion also formulated in the work of Kovacs, in article [5]. This idea motivated the development of the super-cluster strategy. The procedure consists of initiating cluster generation via watershed clustering, followed by the construction of a region adjacency graph (RAG). In the RAG, each vertex represents a cluster and the edges signify the connections between clusters, weighted by Euclidean distances. Clusters are then concatenated with their k-nearest neighbors, based on this global information. This aggregation results in larger clusters that preserve the fundamental distribution of the data. The final step is to add data points to the clusters using the "mesh" option.

The WSSMOTE algorithm can be accessed through its GitHub repository available at the following URL: https://github.com/yamnao/WSSMOTE. For a detailed understanding of the algorithm's pseudocode, please refer

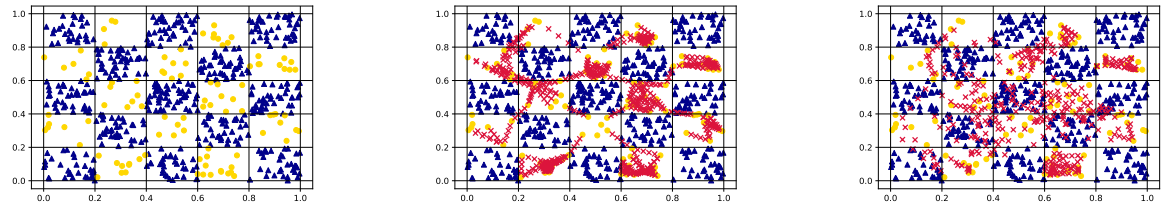
to Algorithm 4. It is important to note that the clusters generated by the watershed algorithm exclusively involve minority data points (line 2, algorithm 4). In fact, according to the literature [1, 2] and various experiments, the use of exclusively minority data points to build clusters yields better results than the combination of minority and majority data points. To visually illustrate the application of WSSMOTE on imbalanced datasets, please refer to figures 5.1 and 5.2. These figures provide visualizations of both strategies of the effects of WSSMOTE on datasets.

Algorithm 4: WSSMOTE

Data:
- imbalanced data D and its labels L
- nb_add: percentage of data points to be added
- strategy choice: choice between 'star', 'mesh', 'concat_k'
- k: parameter k, number of concatenate clusters

```
1  nb_to_add: (nb min data pts - nb maj data pts)*nb_add ;
2  graph, edge_weights: Generate the KNN graph on minority data points;
3  Clusters: Watershed Clustering(graph, edge_weights) ;
4  if strategy choice == 'star' then
5      for C in Clusters do
6          X_mean: Calculate the mean of all the data pts in C ;
7          D_C: Select random data points in C;
8          Generate new data points between D_C and X_mean equidistantly ;
9  if strategy choice == 'mesh' then
10     for C in Clusters do
11         D_C: Select random data points in C;
12         Generate new data points between two D_C data points ;
13 if strategy choice == 'concat_k' then
14     ClustersConcat: Concatenate Clusters using Region Adjacency Graph and
         parameter k;
15     for C' in ClustersConcat do
16         D'_C: Select random data points in C';
17         Generate new data points between two D'_C data points ;
```

Result: *Data* D', *labels* L'

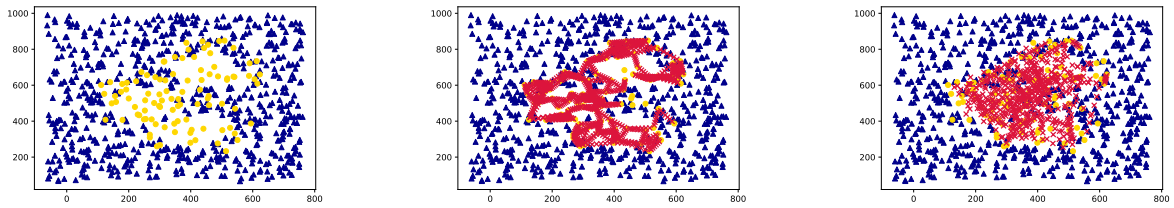


(a) ChessBoard without oversampling

(b) ChessBoard with WSSMOTE oversampling: parameter “mesh”

(c) ChessBoard with WSSMOTE oversampling: parameter “concat_5”

Figure 5.1: Application of WSSMOTE on ChessBoard dataset. Blue triangles correspond to majority data points, yellow circles to minority ones and red crosses to data points added by WSSMOTE method.



(a) Paw without oversampling

(b) Paw with WSSMOTE oversampling: “mesh”

(c) Paw with WSSMOTE oversampling: “concat_5”

Figure 5.2: Application of WSSMOTE on Paw dataset. Blue triangles correspond to majority data points, yellow circles to minority ones and red crosses to data points added by WSSMOTE method.

5.2 Results

Having described and implemented our watershed oversampling approach, the next step is to perform tests and comparative analysis with existing oversampling methods, based on the algorithm and methodology described in Chapter 3. More specifically, we integrate WSSMOTE into the existing smote_variants pipeline. This integration guarantees a consistent evaluation framework. It should be noted that this pipeline uses standard semi-supervised methods such as SVM (with kernal and linear strategy), RandomForest, KNearestNeighbors, AdaBoost and MLPClassifier in combination with the tested sampling methods. These semi-supervised methods have been rigorously tested with a range of parameters, using a grid search approach. There are two key aspects to our testing: firstly, an examination of WSSMOTE’s performance in detecting ACS and, secondly, an evaluation on randomly selected datasets.

5.2.1 Results on PRESEV Dataset

We begin by testing WSSMOTE for the prediction of acute chest syndrome (ACS) in patients with sickle cell disease. This testing is carried out using the PRESEVC dataset and our main objective, as outlined in the previous chapter 1, is the maximization of the negative predictive value (NPV) score, and of the positive predictive value (PPV) score.

The results, classified according to their NPV score, are summarized in table 5.1. These results are based on the comparison of four distinct machine learning methods, each associated with different oversampling techniques. Thus, as shown in Table 5.1, the most optimal combinations for the PRESEVC dataset involve combining WSSMOTE with the linear support vector machine (SVM) classifier.

Ranking	**Oversampling**	**Classifier**
1	WSSMOTE	SVM
2	Gaussian_SMOTE	SVM
3	ROSE	SVM

Table 5.1: Combinations of oversampling and machine learning classifiers that provide the best NPV scores in ACS prediction. These results are obtained using the smote_variants pipeline [6].

In order to provide a global perspective, we compare the results obtained using the combination of WSSMOTE and SVM with the results obtained in previous evaluations using basic machine learning methods. In addition, we join these results with those obtained by integrating Gaussian SMOTE with the SVM classifier. The full results are presented in table 5.2.

Prediction Method	PRESEVC	**PPV (%)**	**NPV (%)**
SVM	Train	35,9 ± 25.8	99,4 ± 1.6
	Test	22.6 ± 15.8	95.8 ± 2
Gaussian SMOTE + SVM	Train	23.1 ± 1.1	96.5 ± 0.8
	Test	24.6 ± 4.2	96.9 ± 2.9
WSSMOTE + SVM	Train	27.7 ± 1.9	96.6 ± 0.4
	Test	28.9 ± 3.1	96.6 ± 2.5

Table 5.2: PPV and NPV scores obtained on PRESEVC using WSSMOTE and comparison with previsous results.

Analysis of the results clearly demonstrates the improvement brought about by the WSSMOTE method. In particular, the PPV score shows a 4.5% improvement over the results obtained with the Gaussian SMOTE method. This improvement is significant. In addition, the implementation of WSSMOTE significantly reduces overfitting. For the PPV score, overfitting falls from 13.3% to just 1.2%, while for the NPV score, it falls from 3.6% to 0.1%. This reduction in overfitting demonstrates the effectiveness

of WSSMOTE to fit the data distribution. WSSMOTE's influence also extends to cross-validation variability. Before oversampling, cross-validation variability was around 20.8%. After the application of WSSMOTE, this variability is considerably reduced, to just 2.5%. This underlines the stability and reliability of the WSSMOTE method. In summary, the use of WSSMOTE undeniably improves the prediction accuracy of ACS in patients with sickle cell disease. This translates into better identification of high-risk patients, facilitating more targeted monitoring and care. In addition, the method enables low-risk patients to be identified with greater accuracy, which may lead to shorter hospital stays. It should be noted that the improved predictive accuracy and reduced variability achieved by WSSMOTE also engenders a high level of confidence in the reproducibility of the method. This raises the prospect of potential application to future patients, reinforcing the overall benefits of the approach.

5.2.2 Results on Randomly Selected Dataset

Encouraged by the promising results obtained in the PRESEV study, we are extending our research to other imbalanced datasets in order to carry out a comprehensive comparison of prediction performance. To this end, we are using the smote_variant pipeline, evaluating the results of four separate machine learning methods and 50 oversampling techniques, each with 20 potential parameter combinations. The cumulative results are presented in table 5.3, where WSSMOTE's ranking is side by side with the other 50 oversampling methods. This ranking is based on the different scores studied for each imbalanced dataset. This evaluation offers a clear perspective on the effectiveness of WSSMOTE in various scenarios.

	accuracy rank	**sensitivity rank**	**specificity rank**	**ppv rank**	**npv rank**	**gacc rank**	**f1 rank**	**auc rank**
yeast1	48	1	50	49	1	16	38	10
ecoli1	34	1	47	46	1	10	16	8
harbeman	27	2	43	24	1	6	11	6
wisconsin	11	3	33	32	4	8	8	12
vehicle1	25	2	46	46	1	19	13	1
glass1	22	2	45	44	2	6	10	11
subcl35	20	29	14	6	26	3	27	5

Table 5.3: Comparison of WSSMOTE with 50 other oversampling methods using the smote pipeline. Each number corresponds to the rank of WSSMOTE for a specific score and an imbalanced dataset. The best rank is 1, the worst is 50.

In addition, we undertake a parallel analysis involving others oversampling methods such as Gaussian SMOTE method [7]. The results of this comparative evaluation for Gaussian SMOTE, given each score and each imbalanced dataset, are summarized in table 5.4. This comparative analysis highlights the performance of Gaussian SMOTE compared to all over-

sampling methods. These detailed tables provide a complete picture, facilitating an informed understanding of how WSSMOTE perform against a range of oversampling methods in various datasets and scoring measures.

	accuracy rank	**sensibility rank**	**specificity rank**	**ppv rank**	**npv rank**	**gacc rank**	**f1 rank**	**auc rank**
yeast1	8	2	9	6	2	43	42	9
ecoli1	3	2	3	2	2	17	5	3
haberman	37	1	7	3	41	12	10	7
wisconsin	42	30	19	27	33	35	42	2
vehicle1	43	1	3	2	33	42	42	47
glass1	31	2	1	14	44	42	42	37
subcl35	2	1	37	8	4	39	2	15

Table 5.4: Comparison of Gaussian SMOTE with 50 other oversampling methods using the smote pipeline. Each number corresponds to the rank of Gaussian SMOTE for a specific score and an imbalanced dataset. The best rank is 1, the worst is 50.

In particular, the results presented highlight the beneficial impact of the WSSMOTE method, which consistently improves NPV and sensitivity scores in the majority of the various imbalanced datasets examined. This performance trend positions WSSMOTE as a valuable method, capable of solving classification challenges. However, it is important to recognize a fundamental principle recognized in the literature, articulated in various articles such as those referenced in [8, 4]. There is no universal silver bullet in the field of oversampling methods, and this also applies to WSSMOTE. Comparative analysis reveals that Gaussian SMOTE sometimes outperforms WSSMOTE, illustrating the variability in the efficacy of oversampling methods in different contexts. This suggests that the selection of an appropriate oversampling method should be guided by the specific characteristics of the dataset and the objectives of the classification problem. Although WSSMOTE offers performance improvements that are admirable in many cases, it is prudent to recognize that no single method can universally dominate all scenarios.

5.3 Conclusion

This new oversampling method, WSSMOTE, has demonstrated its ability to improve the detection of ACS in patients with sickle cell disease (SCD). The marked improvement, with a PPV increase of around 4% compared to other oversampling methods, represents a significant advance towards the accurate identification of ACS patients, thus improving the quality of patient care and hospital management. Moreover, WSSMOTE's influence also reduce the overfitting from 13.3% to just 1.2% which is promising for the reproducibility and reliability of future studies.

More generally, the applicability of WSSMOTE is not limited to ACS

prediction. Its effectiveness extends to various imbalanced classification challenges. However, it is essential to stress that, like all oversampling methods, WSSMOTE is not a universal silver bullet. The effectiveness of any oversampling method, including WSSMOTE, depends on the complexity and unique characteristics of the imbalanced dataset in question. While oversampling strives to mimic the structure of the dataset, imbalanced datasets can present immense variations.

However, the incorporation of connectivity considerations, illustrated by WSSMOTE, appears to be a powerful strategy in the field of imbalanced data processing. In addition, the PPV of ACS prediction can be further improved, as it is higher with the use of the usual SVM. For this reason, we are considering a new algorithm that focuses on connectivity considerations, distancing itself from oversampling methods.

References

[1] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap. "DBSMOTE: Density-Based Synthetic Minority Over-sampling TEchnique". In: *Applied Intelligence* 36 (2012).

[2] D. Cieslak, N. Chawla, and A. Striegel. "Combating imbalance in network intrusion datasets". In: *2006 IEEE International Conference on Granular Computing* (2006), pp. 732–737.

[3] S. Gazzah and N. Amara. "New Oversampling Approaches Based on Polynomial Fitting for Imbalanced DataSets". In: *2008 The Eighth IAPR International Workshop on Document Analysis Systems* (2008).

[4] A. B. Hassanat et al. "Stop Oversampling for Class Imbalance Learning: A Critical Review". In: *Digital Object Identifier* (2022).

[5] G. Kovacs. "An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets". In: *Applied Soft Computing Journal* 83 (2019).

[6] G. Kovacs. "Smote-variants: a python implementation of 85 minority oversampling techniques". In: *Neurocomputing* (2019).

[7] H. Lee, J. Kim, and S. Kim. "Gaussian-Based SMOTE Algorithm for Solving Skewed Class Distributions". In: *The International Journal of Fuzzy Logic and Intelligent Systems* (2017).

[8] B. Santos et al. "Synthetic Over Sampling Methods for Handling Class Imbalanced Problems: A review". In: *IOP Conference Series: Earth and Environmental Science* 58 (2017).

Chapter 6

Exploring Metric Learning Theory & Application to Imbalanced Datasets

The fundamental objective of metric learning, also known as distance metric learning, involves learning a distance metric that brings data points with identical labels closer together, while moving those with different labels further apart, as shown in figure 6.1. In other words, metric learning techniques aim to identify a space in which data points belonging to distinct classes are clearly separated. Many existing prediction methods are rooted in the principles of metric learning, covering techniques such as clustering approaches like KMeans [8], dimension reduction methods like principal component analysis (PCA) [14], and supervised techniques like support vector machines [10]. In this section, we focus on metric learning techniques designed for supervised classification problem, with particular emphasis on those designed to handle imbalanced datasets. We also include these metric learning methods into the smote_variants pipeline in order to perform a complete comparisons between metric learning methods and oversampling ones.

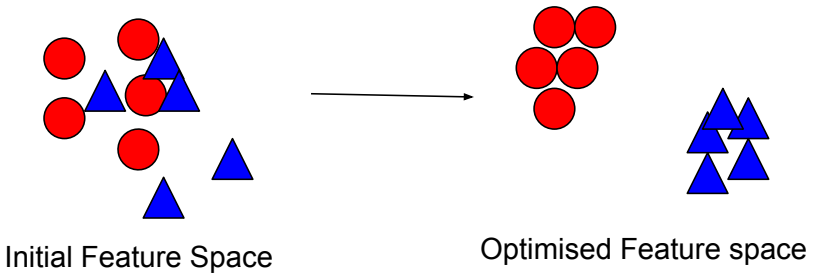


Figure 6.1: Impact of the metric learning method on the feature space.

6.1 Metric Learning for Balanced Dataset

To understand how a metric learning algorithm works, it is essential to understand what a metric space is.

Metric Space – A metric space [4] is a set of data points whose elements are linked by a distance.

Definition 1 (Metric) *Let M a set. d is defined as a metric, also called distance, if:*

- $d : M * M \to \mathbb{R}$ *is a function,*
- $\forall\ x \in M,\ d(x, x) = 0$
- $\forall\ x,y \in M,$ *and* $x \neq y,\ d(x, y) > 0,$
- $\forall\ x,y \in M,\ d(x, y) = d(y, x)$
- $\forall\ x,\ y,\ z \in M,\ d(x, z) \leq d(x, y) + d(y, z)$ ***triangle inequality***

The most familiar example of a metric space is the 2D Euclidean space. In this space, M is equal to $\mathbb{R}^2$, so that its elements X_1 and X_2 can be decomposed into $X_1 = (x_1, y_1)$ and $X_2 = (x_2, y_2)$. The associated distance is the Euclidean distance, defined as

$$d((x_1, x_2), (y_1, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

To arrive at an optimal metric space, it is necessary to address an optimization problem that has been incorporated into a metric learning algorithm. It's important to note that throughout this chapter, we'll maintain the assumption of a fixed number of features, allowing the space to be modified only through the distance function. We will now describe some of these optimization problem in metric learning:

- **Large Margin Nearest Neighbor (LMNN) [13]** – LMNN is a known metric learning algorithm and is based on the Mahalanobis distance [9].

  Definition 2 (Mahalanobis Distance) *Let $A \in \mathbb{R}^{dxd}$ be a positive semi definite matrix and d the dimension of the set. Thus, A and can be decomposed as $A = L^T L$ with $L \in \mathbb{R}^{r*d}$ and r the rank of A.*
  The Mahalanobis distance between two data points x_i, x_j in $\mathbb{R}^d$ is defined as $D_A(x_i, x_j) = (x_i - x_j)^T A(x_i - x_j)$

  The Mahalanobis distance is a generalization of the Euclidean distance. The goal of the LMNN algorithm is to separate the different classes. In other words, LMNN wants to move data points of the same class closer together and data points of different classes further apart. For this purpose, two functions have been defined: l_{push} and l_{pull}. The aim of the first is to move two neighbouring data points of different classes away from each other, while the second moves two neighbouring data points closer together, as shown in figure 6.2.

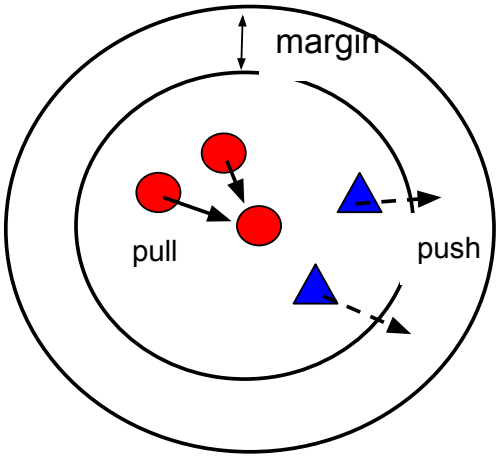


Figure 6.2: Illustration of *push* and *pull* functions developed in the LMNN algorithm.

Thus, the optimization problem is formulated as follow:

$$\min_{A \geq 0} F(A) = \frac{1}{n^2}[(1-\mu) l_{pull} + \mu l_{push}]$$
$$l_{pull} = \sum_{(i,j) \in N} ||A(x_i - x_j)||^2$$
$$l_{push} = \sum_{(i,j) \in N} [\sum_{z} (1 - y_{i,j})[1 + ||A(x_i - x_j)||^2 - ||A(x_i - x_z)||^2]_+$$

where:

μ = a parameter tuned by cross validation
n = number of samples
z_+ = max(0, z)
$y_{i,j}$ = 1 if x_i and x_j from the same class, else 0
N = two neighbouring data points
x = data point

By optimizing the matrix A, we can improve distance metrics, creating class separation and therefore improving predictive results.

- **Information Theoric Metric Learning (ITML)** [3] – ITML is also a metric learning algorithm based on the Mahalanobis distance. The objective of ITML is to stay "closer" to a predefined matrix. For this purpose, the algorithm is designed through the relative entropy, which is used to measure the distance between two Mahalanobis functions A_0 and A.

$$KL(p(x, A_0)||p(x, A)) = \int p(x, A_0) \log \frac{p(x, A_0)}{p(x, A)} dx$$

Thus the ITML goal is to minimize this entropy with two constraints:

$d_A(x_i, x_j) \leq u$ with x_i and x_j two data points from the same class

$d_A(x_i, x_j) \geq u$ with x_i and x_j two data points from the different classes

This optimisation problem can be solved using matrix methods and is most often presented as follows:

$$KL(p(x, A_0)||p(x, A)) = trace(AA_0^{-1}) - logdet(AA^{-1}) - n$$

where: n = number of samples

ITML aims to improve the performance of machine learning algorithms by adapting the distance metric to the specific problem. It can be particularly useful in cases where data distributions are complex or traditional distance metrics fail to effectively capture the underlying relationships between data. The ITML algorithm is faster than the LMNN algorithm and a really powerful technique, but it may also require careful parameter tuning.

- **Geometric Mean Metric Learning (GMML)** [15] – GMML is also a metric learning algorithm. GMML aims to find a metric that focuses on the geometric properties of the data distribution, making it well suited to tasks such as classification, clustering and search. The central idea of GMML is to learn a transformation of the original feature space that maximizes the geometric mean of pairwise distances between similar instances while minimizing the geometric mean of pairwise distances between dissimilar instances. GMML is to some extent based on the LMNN and ITLMN algorithms. In fact, in both methods, the "push" and "pull" loss functions are performed symmetrically. On the contrary, in GMML, the aim is to treat the two functions differently by solving the following optimization problem:

$$\min_{A \geq 0} F(A) = trace(A \sum (x_i - x_j)^T (x_i - x_j)) + trace(A^{-1} \sum (x_i - x_j)^T (x_i - x_j))$$

GMML's geometric mean objective addresses the limitations of the traditional Euclidean distance, particularly when dealing with imbalanced classes or non-linear data distributions. By focusing on the geometric properties of the data, GMML aims to create a measure that better reflects the structure of the data.

These three methods (LMNN, GMML and ITML) are accessible in Python via the website provided: http://contrib.scikit-learn.org/metric-learn/. These implementations use a gradient descent algorithm to address their respective optimization problems, as detailed in article [11].

6.2 Metric Learning for Imbalanced Datasets

LMNN, ITML and GMML are generally applied to balanced datasets, as their original formulations may not deal effectively with imbalanced datasets, leading to bias against minority classes. Aware of this limitation, researchers have developed alternative algorithms to improve the performance of these methods on imbalanced datasets. Many of these approaches involve incorporating class-specific weights directly into the loss function.

For example, an algorithm proposed in the article [5] is based on the ITML algorithm. This approach deals with imbalanced datasets by introducing class-specific weights that influence the learning process. Similarly, another algorithm presented in the article [7] extends the LMNN method. This method is known as IML. The IML algorithm modifies the standard LMNN approach by considering each class individually and assigning different weights to them. The objective is to prevent bias towards majority classes and improve the performance of the algorithm on imbalanced data. The loss function of the IML algorithm can be described as follows:

$$\begin{aligned}\min_{A\geq 0} F(A) = \min_{A\geq 0} \mu &\sum_{(x_i,x_j)\in Sim^+} l_1(A,x_i,x_j) \\ &+ (1-\mu) \sum_{(x_i,x_j)\in Sim^-} l_1(A,x_i,x_j) \\ &+ \lambda \sum_{(x_i,x_j)\in Dis^+} l_2(A,x_i,x_j) \\ &+ (1-\lambda) \sum_{(x_i,x_j)\in Dis^-} l_2(A,x_i,x_j)\end{aligned}$$

where:

Sim_+	= two data points from the majority class
Sim_-	= two data points from the minority class
Dis_+	= the first on from the majority and the second from the minority
Dis_-	= the first one from the minority and the second from the majority
$l_1(A,x_i,x_j)$	$= [d_A(x_i,x_j)-1]_+$
$l_2(A,x_i,x_j)$	$= [1+m-d_A(x_i,x_j)]_+$
m	= margin
μ	= tuning parameter to define
λ	= tuning parameter to define

By incorporating class-specific weights and refining the loss function, algorithms such as IML aim to reduce the challenges posed by imbalanced datasets in metric learning. These modifications enable methods to better adapt to data distribution and improve performance, particularly for tasks involving imbalanced classes.

In addition, other metric learning algorithms adapted to imbalanced datasets have been developed without modifying class weights. A notable example is the iterative metric learning method (ITML) proposed in article [12]. This method takes a different approach to the challenges of imbalanced data while employing the LMNN algorithm. The ITML method focuses on small regions of the dataset that contain an approximately equal number of minority and majority data points. This strategy aims to ensure that the learning process is not biased in favor of one class due to imbalanced proportions. The approach involves several iterative steps, and at each stage a subset of the data is selected on the basis of this criterion of equal class representation. The ITML algorithm can be described as follows:

1. **Select Balanced Subsets**: At each iteration, subsets of the dataset are chosen to maintain a balanced representation of minority and majority classes.
2. **Apply LMNN**: The LMNN algorithm is then applied to each of these balanced subsets.
3. **Iterative Refinement**: The algorithm repeats the above steps iteratively, updating the distance metric with each iteration. This process allows the distance metric to converge towards a representation that better suits the imbalanced data.

By focusing on balanced subsets and repeatedly applying the LMNN algorithm, the ITML method aims to adapt the distance metric to imbalanced data.

6.3 Metric Learning & Oversampling Comparisons

IML [7] shows promising results, but it should be noted that none of the above-mentioned algorithms have been compared with each other on randomly selected imbalanced real-world datasets. Furthermore, they have not been thoroughly compared or integrated with oversampling methods. To fill this gap, we decided to incorporate these metric learning algorithms into the smote_variant pipeline. The aim of this integration is to comprehensively evaluate the performance of these algorithms against each other and against existing oversampling techniques. The combination of metric learning algorithms and oversampling methods, as part of the smote_variant pipeline, aims to explore the potential of these algorithms to improve the performance of real-world datasets.

To achieve this, we undertook a series of modifications to the existing pipeline. A new class has been introduced, called “MetricLearning”, which serves as a framework for the integration of diverse metric learn-

ing methods. The main objective is to enable the integration of metric learning methods that can adapt and transform both the training and testing parts of the dataset. By exploiting the resources of libraries such as http://contrib.scikit-learn.org/metric-learn/ and https://github.com/LeoGautheron, we have successfully implemented four metric learning algorithms: GMML [15], LMNN [13], IML [7], and ITML [3].

As part of our pipeline changes, the basic structure of the smote pipeline has been modified. In particular, the "evaluation" part of the algorithm has undergone changes to accommodate these updates, as highlighted by the pseudo-code 5. It's important to note that the most difficult aspect lies not in the direct modifications to the code, but in designing an effective strategy for storing the results and parallelizing the code while incorporating the newly added metric learning methods. To facilitate testing, we have introduced "empty" oversampling and metric learning methods. This allows us to run tests without metric learning and/or oversampling methods.

To fully evaluate the impact of metric learning methods on imbalanced datasets, we started our investigation using the "No_SMOTE" oversampling method, which essentially corresponds to no oversampling at all. This approach facilitates a direct comparison between machine learning methods improved by metric learning and their unmodified equivalents, whitout metric learning. The results of this initial comparison are presented in table 6.1, and are based on the imbalanced datasets mentioned earlier in chapter 2. The evaluation is carried out using various scoring measures, as described in chapter 2. Machine learning methods considered for this comparison include SVM [10], KNN [2], RF [1] and AdaBoost [6]. The main objective here is to evaluate how the integration of metric learning techniques influences the performance of these machine learning algorithms on imbalanced datasets. These results lay the foundations for understanding the potential benefits that metric learning can offer to improve classification results in imbalanced classification problem.

Algorithm 5: Modified version of the main part of smote_variants for metric learning methods

Data:

- imbalanced data D and its labels L
- l_classifier: list of usual machine learning classifier
- l_oversampler: list of oversampling methods
- l_oversampler_params: list of parameters for each oversampling method
- **l_metric_learning: list of metric learning methods**
- **l_metric_learning_params: list of parameters for each metric learning method**

```
Creation of folds using D and L;
/* Sampling part                                                    */
for each fold do
    for each metric learning method do
        for each parameters corresponding to the metric learning method do
        fold_over: Apply metric learning method with the corresponding parameter on the fold dataset;
        for each oversampling method do
            for each parameters corresponding to the oversampling method do
            fold_over: Apply oversampling method with the corresponding parameter to the fold_over dataset;
            Save the new folding in the right place with the paramater and the oversampling used;
/* Evaluation part                                                  */
for each fold_over do
    for each classifier do
        scores: Compile score using the selected classifier;
        Save scores in the right place with all the information corresponding;
Modify the scores obtained into an organized table;
Result: Table containing all results
```

	ecoli1	glass1	haberman	paw	subcl35	vowel0	wisconsin
Acc	IML	GMML	GMML	Without	ITML	Without	Without
Sens	LMNN	Without	GMML	ITML	Without	Without	Without
Spec	Without	ITML	Without	Without	GMML	Without	IML
PPV	Without	ITML	GMML	Without	ITML	Without	IML
NPV	Without	Without	IML	ITML	Without	Without	ITML
GAcc	Without	ITML	GMML	ITML	Without	Without	Without
F1	IML	IML	GMML	ITML	Without	Without	Without
AUC	IML	IML	IML	IML	Without	Without	Without

Table 6.1: Comparison of data classification by combining metric learning methods with standard machine learning methods or using only standard machine learning methods (corresponding to "Without").

Analysis of the table 6.1 highlights the potential effectiveness of using metric learning approaches to improve prediction performance on imbalanced datasets in isolation. As expected, the IML approach outperforms LMNN when applied to imbalanced datasets, but, surprisingly, does not consistently outperform GMML or ITML - methods originally designed for balanced datasets. In addition, using metric learning alone does not produce entirely convincing results. In order to address this issue in a complete way, we carried out a multi-stage comparison involving three types of combinations:

- **Metric Learning and Usual Machine Learning Methods**: This combines metric learning techniques with conventional machine learning methods.

- **Oversampling and Usual Machine Learning Methods**: This combines oversampling methods with conventional machine learning methods.

- **Metric Learning, Oversampling and Usual Machine Learning Methods**: This combination incorporates both metric learning and oversampling techniques with conventional machine learning methods.

The results, presented in table 6.2, reveal that the most promising combinations for effectively predicting imbalanced datasets are the latter two options: using oversampling techniques in tandem with metric learning or conventional machine learning methods. In addition, it should be noted that our WSSMOTE method performs well, particularly when combined with the IML metric learning approach, or when used in combination with conventional machine learning methods.

	paw		pima		vehicle1	
	Metric Learning	**Oversampling**	**Metric Learning**	**Oversampling**	**Metric Learning**	**Oversampling**
Acc	GMML	Gazzah	Without	WSSMOTE	GMML	distance_SMOTE
Sens	IML	WSSMOTE	Without	Gazzah	Without	CCR
Spec	GMML	AUC	Without	WSSMOTE	Without	WSSMOTE
PPV	Without	Without	Without	WSSMOTE	Without	WSSMOTE
NPV	IML	WSSMOTE	Without	ROSE	Without	Gaussian_SMOTE
Gacc	IML	WSSMOTE	ITML	SMOTE_OUT	Without	WSSMOTE
F1	GMML	Gazzah	Without	Gaussian_SMOTE	GMML	distance_SMOTE
AUC	ITML	AHC	Without	WSSMOTE	LMNN	polynom_fit_SMOTE
	vowel0		**ecoli1**		**presev**	
	Metric Learning	**Oversampling**	**Metric Learning**	**Oversampling**	**Metric Learning**	**Oversampling**
Acc	Without	polynom_fit_SMOTE	Without	SMOTE_Cosine	ITML	CCR
Sens	Without	Gazzah	Without	polynom_fit_SMOTE	IML	WSSMOTE
Spec	Without	polynom_fit_SMOTE	ITML	WSSMOTE	LMNN	CCR
PPV	ITML	WSSMOTE	ITML	WSSMOTE	ITML	Without
NPV	Without	polynom_fit_SMOTE	Without	polynom_fit_SMOTE	Without	WSSMOTE
Gacc	Without	polynom_fit_SMOTE	Without	Gazzah	ITML	CBSO
F1	Without	polynom_fit_SMOTE	Without	SMOTE_Cosine	ITML	CBSO
AUC	Without	polynom_fit_SMOTE	ITML	SMOTE_TomekLinks	ITML	Gazzah

Table 6.2: Comparison of data classification using different combinations: metric learning alone, oversampling alone, or oversampling and metric learning.

6.4 Conclusions

The performance improvement resulting from the combination of metric learning methods and oversampling techniques is apparent. However, it should be noted that this improvement is not uniformly consistent, and there are cases where none of the metric learning methods tested proves efficient. This variability could be due to the difficulties associated with defining appropriate metric learning parameters, a task that often involves exhaustive exploration of multiple combinations. Another factor could be the uniform basis of the metric learning methods tested, which are all based on the Mahalanobis distance, a generalization of the Euclidean distance.

Given the encouraging results of the WSSMOTE approach, which is based on a connectivity criterion, we set out to develop a new metric learning method centered on ultrametric distance. This alternative distance metric introduces a connectivity-based criterion that can offer new capabilities for dealing with class imbalance and improving classification performance.

References

[1] Breiman. “Random Forests”. In: 45 (2001).

[2] T. Cover and P. Hart. “Nearest neighbor pattern classification”. In: *IEEE Transactions on Information Theory* 13 (1967).

[3] J. Davis et al. “Information-Theoretic Metric Learning”. In: *ICML* (2007).

[4] J. Dieudonné. *Foundations of Modern Analysis*. Ed. by Columbia University. 1960.

[5] L. Feng et al. “Learning a distance metric by balancing kl-divergence for imbalanced datasets”. In: *IEEE Transactions on Systems, Man and Cybernetics: Systems* (2018).

[6] Y. Freund and R. Schapire. “A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting”. In: (1995).

[7] L. Gautheron et al. “Metric Learning for Imbalanced Data with Generalization Guarantees”. In: *Pattern Recognition Letters* 133 (2020), pp. 298–304.

[8] X. Jin and J. Han. “K-Means Clustering”. In: *Encyclopedia of Machine Learning* (2007).

[9] P. C. Mahalanobis. “On the generalised distance in statistics”. In: *Proceedings of the National Institute of Sciences of India* 2 (1936).

[10] J. C. Platt. “Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods”. In: *Advances in Lage margin Classifiers* (1999).

[11] S. Ruder. “An overview of gradient descent optimization algorithms”. In: *Machine Learning* (2017).

[12] N. Wang et al. “Iterative metric learning for imbalance data classification”. In: *IJCAI* (2018).

[13] K. Weinberger and L. Saul. “Distance metric learning for large margin nearest neighbor classification”. In: *JMLR* 10 (2009).

[14] S. Wold, K. Esbensen, and P. Geladi. “Principal component analysis”. In: *Chemometrics and Intelligent Laboratory Systems* 2 (1987), pp. 37–52.

[15] P. Zadeh, R. Hosserni, and S. Sra. “Geometric Mean Metric Learning”. In: *ICML* (2016).

Chapter 7

WML: an Ultrametric Learning Method for Imbalanced Data

Our novel oversampling method called WSSMOTE, based on watershed transformation principles, has not only demonstrated its potential for improving PRESEV predictions, but also its efficacy in a range of real-world classification problems. These results highlight the potential and usefulness of integrating connectivity considerations when dealing with imbalanced datasets. In addition, our analysis has highlighted the limitations of conventional oversampling methods. These methods, which aim to balance the distribution of classes, often introduce biases and fail to generate the diversity needed to bridge information gaps between classes. To overcome these difficulties, another approach is to partition classes into coherent clusters by applying metric learning methods. As we highlighted in the previous chapter, metric learning can improve predictive performance of imbalanced datasets. Building on the promising results obtained with watershed transformations, we decided to exlore the field of metric learning using a graph-based approach. To this end, we present the formulation of a graph optimization method and subject it to an evaluation in the context of real-world imbalanced classification problems.

7.1 Algorithm Implementation

Our objective is to adapt the ultrametric fitting algorithm, accessible at https://github.com/PerretB/ultrametric-fitting, to the classification of imbalanced datasets. Our aim is to develop a novel algorithm capable of fitting and transforming both the training and testing part of the dataset. In this way, our optimization problem must modify the dataset and not only the graph as it was done previously, article [2]. A visual representation of the desired optimization problem is provided in Figure 7.1.

The initial step is to convert our dataset into a k-nearest neighbor graph. This graph serves as the basis for defining vertex pairs. Specifically, each

edge of the graph connects two data points: one as a source and the other as a target. We then assign a Euclidean distance to each of these edges. This distance assignment confers weights on the graph, enabling us to exploit its minimal spanning tree (MST) for the computation of ultrametric distances. We then use a simple neural network and a designated loss function to navigate an optimization problem, using the concept of ultrametric subdominance. In addition, as the graph weights are computed by linear operations, backtracking is performed using a standard gradient descent algorithm [4]. Consequently, our algorithm aims to improve classification performance by adapting the ultrametric fitting methodology to the domain of imbalanced datasets.

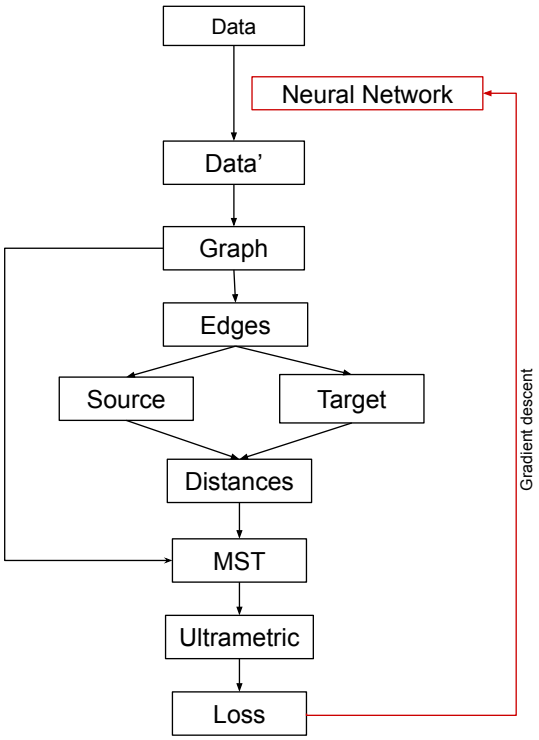


Figure 7.1: Schematic of ultrametric optimization problem for imbalanced dataset classification.

To formulate the optimization problem for predicting imbalanced datasets, we need to define an appropriate loss function. We have devigned two loss functions inspired by LMNN [5] and the push/pull theory. Let A and B denote the two classes, l represent the labels of the data points, and C be a constant. The first loss function is defined as follows:

$$L_1 = \sum_{\substack{(x,y)\in E/\\ l(x)=A,l(y)=A}} d_u(x,y) + \sum_{\substack{(x,y)\in E/\\ l(x)=B,l(y)=B}} d_u(x,y) - \sum_{\substack{(x,y)\in E/\\ l(x)=A,l(y)=B}} (d_u(x,y) - C)$$

To incorporate the imbalanced criteria, a second loss function is defined as follows. Here, nb_A represents the number of edges where both the source and the target belong to class A, nb_B is the number of edges of class B,

and nb_T is the total number of edges.

$$L_2 = \frac{nb_A}{nb_T} \sum_{\substack{(x,y)\in E/ \\ l(x)=A, l(y)=A}} d_u(x,y) + \frac{nb_B}{nb_T} \sum_{\substack{(x,y)\in E/ \\ l(x)=B, l(y)=B}} d_u(x,y) + \frac{nb_T - (nb_A + nb_B)}{nb_T} \sum_{\substack{(x,y)\in E/ \\ l(x)=B, l(y)=B}} (d_u(x,y) - C)$$

With these loss functions in place, we implemented the optimization algorithm, incorporating it into the smote_variants pipeline. The algorithm, named WML for the loss function L_1 and WMLI for the loss function L_2, follows the same structure as other algorithms in the pipeline i.e. with an fitting function for the training part and a transformation function for the testing and training parts. The pseudocode for the WML algorithm is presented in Algorithm 6. The WMLI algorithm is similar, with a modification in the loss function.

Using a neural network can be very complex, especially when it comes to defining the number of layers, non-linear functions and parameters. We tried different combinations, parameters and layers, and found that the one that gave the best results most of the time was a simple neural network with a single layer. It is this simple neural network that we have used in the following results and visualizations.

7.2 Results

7.2.1 Tests and Visualization

To validate our optimization algorithm, which aims to separate data points of different classes, we conducted tests on small 2D datasets that were manually created. The results of these tests are depicted in Figures 7.2, 7.3, and 7.4. We observe successful separation of the two classes in all cases. However, it is important to note that certain parameters, such as the number of epochs and the descent coefficient for the gradient descent algorithm, along with the number of neighbors used to build the graph, greatly impact the outcomes on these small datasets. Moreover, in these small-scale examples, we might not be able to discern a significant difference between WML and WMLI, even when dealing with imbalanced datasets.

Algorithm 6: Fitting part of the WML algorithm

Data:

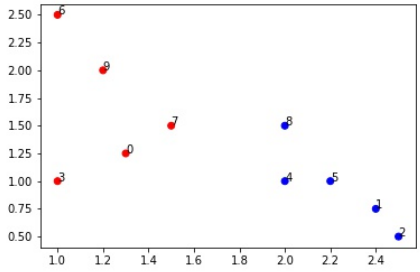
- Train_Data: imbalanced data train and
- L: training labels
- lr: learning rate
- epochs: number of epochs
- knn: number of nearest neighbor

Model: Define is a neural network model. Example: single linear layer with a PReLU activation function.;

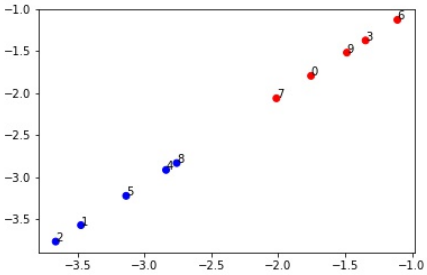
for $t \in epochs$ **do**

- Graph: Define the knn graph using the training data points and knn parameter;
- S, T: Train[sources], Train[targets];
- distances: $\sum_{i \in \text{nb features}} (S_i - T_i)^2$;
- M: Calculates an ultrametric distance from a given graph and daistances. It uses hierarchical clustering to determine the ultrametric distances.;
- Make_pairs: Generates groups and pairs of edges based on graph connectivity and class labels. It identifies edges connecting two instances from the minority class, two instances from the majority class, and instances from both classes. ;
- Loss: Calculates a loss value based on ultrametric distances and the identified groups and pairs. It computes distances between hierarchical clusters for the different edge groups and uses these distances to calculate the loss.;
- Train_Data: Backward the loss function using gradient descent algorithm;

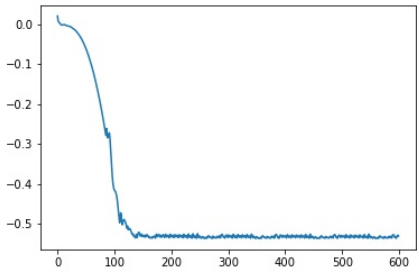
Result: Train_Data



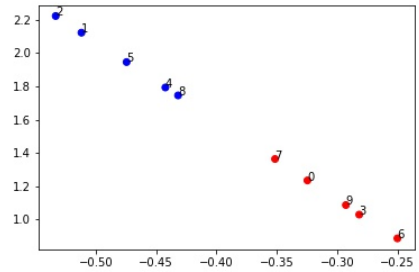
(a) Initial Example 1



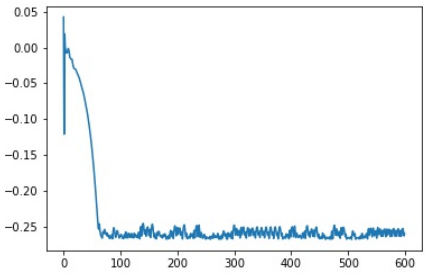
(b) Example 1 With WML



(c) Loss Function Example 1 With WML

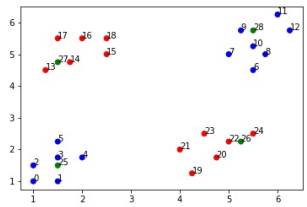


(d) Example 1 With WMLI

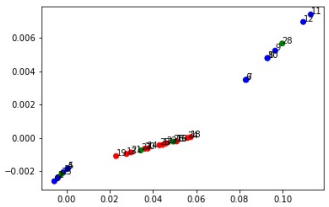


(e) Loss Function Example 1 With WMLI

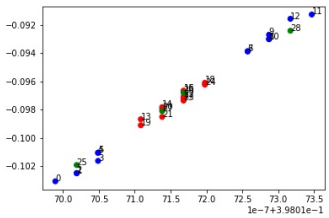
Figure 7.2: First small balanced example of class separation using the WML and WMLI algorithms. The blue data points represents the class A and red ones the class B.



(a) Initial Example 2

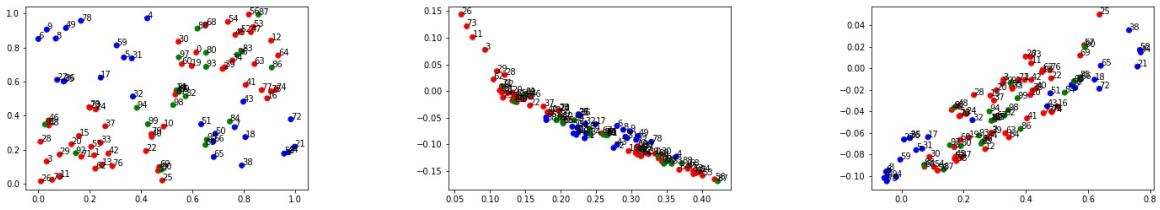


(b) Example 2 With WML



(c) Example 2 With WMLI

Figure 7.3: Second small balanced example of class separation using the WML and WMLI algorithms. The blue data points represents the class A, the red ones the class B and the green the testing data points, i.e. unlabelled ones.



(a) Initial Example 3 (b) Example 3 With WML (c) Example 3 With WMLI

Figure 7.4: First small imbalanced example of class separation using the WML and WMLI algorithms. The blue data points represents the class A, the red ones the class B and the green the testing data points, i.e. unlabelled ones.

Having observed promising results in our tests on small datasets, we proceeded to integrate both the WML and WMLI algorithms into the smote_variants pipeline. This integration will allow us to gain a better understanding of the impact of WML and WMLI on imbalanced datasets and their efficiency in improving classification results.

7.2.2 Performance of WML and WMLI on Random Imbalanced Datasets

In our evaluation, we first conducted a comparison between the WML and WMLI algorithms against other metric learning methods (GMML, ITML, LMNN, and IML) as well as traditional machine learning methods (SVM, RF, KNN, and AdaBoost). The tests were performed on imbalanced datasets, aiming to identify the most efficient metric learning method in terms of different scoring metrics. The summarized results of this comparison can be found in Table 7.1.

	Ecoli1	**glass1**	**haberman**	**paw**	**subcl35**	**vowel0**	**wisconsin**
Acc	IML	WMLI	GMML	Without	ITML	Without	Without
Sens	LMNN	Without	WML	ITML	Without	Without	Without
Spec	Without	ITML	Without	WMLI	GMML	Without	IML
PPV	Without	ITML	GMML	Without	ITML	Without	IML
NPV	Without	WMLI	IML	ITML	Without	Without	ITML
Gacc	Without	ITML	GMML	ITML	Without	Without	Without
F1	IML	WMLI	GMML	ITML	Without	Without	Without
AUC	IML	WMLI	WMLI	IML	Without	Without	WML

Table 7.1: Comparison of data classification by combining WML or WMLI and others metric learning methods with usual machine learning methods or using only standard machine learning methods (corresponding to “Without”).

The results obtained indicate that WML and WMLI contribute positively to the prediction performance of imbalanced datasets, even without the use of oversampling methods. However, as with oversampling techniques, it is clear that no metric learning method is a silver bullet. In some

cases, none of the metric learning methods offers significant improvements and, in these cases, traditional machine learning methods remain the best choice for predicting imbalanced datasets. This reinforces the importance of considering different approaches and experimentation to determine the most appropriate strategy for a given dataset and problem.

7.2.3 Performance of WML and WMLI in Combination with Oversampling on Random Imbalanced Datasets

Then, as the previous section shows that the combination of oversampling and metric learning improves prediction results, we decided to compare the combination of WML and WMLI with oversampling methods. In fact, using WML or WMLI alone gives mixed results, so perhaps combining with oversampling methods could help. At this point, we encountered an important issue in our algorithm while generating graphs from the dataset using the k-nearest neighbors (knn) method provided by the "sklearn.neighbors" library. This knn method uses distances to determine neighboring data points. However, a problem arises when two data points are extremely close, i.e., when their distance is less than 10^{-5}. In such cases, the algorithm treats them as identical and excludes one of them from being a neighbor. Although this might not be a major problem on its own, it becomes significant in our context. Since our goal is to bring data points closer together during optimization, some data points become very close or nearly identical. As a result, many data points are treated as duplicates by the algorithm, leading to an unconnected graph. This issue prevents us from generating minimum spanning trees (MSTs) and subsequently defining ultrametric distances.

To address this problem, we introduced a regularization term in the distance calculation between two data points. In the previous version, we calculated the distance between the source (S) and target (T) of an edge using the formula:

$$d = \sum_{i \in \text{nb features}} (S_i - T_i)^2$$

By adding a "regularization" term, the distance becomes:

$$d = \sum_{i \in \text{nb features}} (S_i - T_i)^2 + 10^{-4}$$

After introducing the regularization term to address the issue with extremely close data points, we conducted tests on imbalanced datasets to evaluate the performance of our methods. The results of the best combinations achieved on two imbalanced datasets are summarized in Table 7.2. Similar results and conclusions were observed for the other datasets as well.

	presev		vehicle1	
	Combination (ML + Over)	**Value**	**Combination (ML + Over)**	**Value**
Acc	polynom_fit + WML	82,5 ± 1,1	Cure_SMOTE + IML	74,7 ± 1
Sens	WSSMOTE + IML	100 ± 0	Random_SMOTE + IML	99,7 ± 0,7
Spec	Without + ITML	82,3 ± 1,1	WSSMOTE + WMLI	100 ± 0
PPV	polynom_fit + WML	40,8 ± 3,8	polynom_fit + WML	89,0 ± 2
NPV	Selected_SMOTE + WML	96,8 ± 4,4	ROSE + WML	48,9 ± 23
Gacc	polynom_fit + WML	65,5 ± 5,9	polynom_fit + WML	66 ± 3,2
F1	SPY + WML	0,8 ± 0,2	Random_SMOTE + IML	85,4 ± 0,3
AUC	polynom_fit + WML	73,2 ± 4,8	polynom_fit + WML	73,2 ± 3,9

Table 7.2: Comparison of imbalanced data predictions using different combinations: metric learning alone, oversampling alone, or oversampling and metric learning.

Based on the results, several important conclusions can be made from the experiments:

- **Impact of WML and WMLI**: WML and WMLI demonstrate their effectiveness in improving predictions on imbalanced datasets. However, it is crucial to note that, similar to other oversampling and metric learning methods, no single method is optimal across all classification problems. The effectiveness of these methods varies depending on the specific evaluation metrics and the characteristics of the imbalanced datasets being considered.
- **Complexity and Parameter Sensitivity**: The complexity of the WML and WMLI algorithms is notable, and their performance is heavily reliant on various parameters, including graph construction, nearest neighbor selection, and gradient descent settings. These parameters are interconnected and can greatly impact the results. Moreover, due to memory limitations, not all possible parameter combinations can be exhaustively tested.
- **High Variability**: The comparison between Tables 7.1 and 7.2 highlights the high variability in this type of preprocessing method. The choice of the most suitable preprocessing strategy for a particular imbalanced dataset is not straightforward and can be laborious. The optimal method might vary even within the same dataset, underscoring the need for careful experimentation and parameter tuning.

In short, although WML and WMLI demonstrate the potential to improve imbalanced predictions by taking advantage of connectivity, their efficiency depends on the specifics of the dataset, the evaluation measures chosen and the configuration of their parameters. This variability underlines the importance of conducting thorough experiments and considering multiple pre-processing strategies when dealing with imbalanced datasets.

7.2.4 Performance of WML and WMLI in PRESEV

However, since our primary dataset is PRESEV and we have observed that WML has the potential to improve predictions, we applied the techniques discussed in Chapter 1 to calculate results for PPV and NPV scores. The obtained results are presented in Table 7.3.

Prediction Method	**PRESEVC**	**PPV (%)**	**NPV (%)**
SVM	Train	35,9 ± 25.8	99,4 ± 1.6
	Test	22.6 ± 15.8	95.8 ± 2
WSSMOTE + SVM	Train	27.7 ± 1.9	96.6 ± 0.4
	Test	28.9 ± 3.1	96.6 ± 2.5
WML + Selected SMOTE + SVM	Train	21,5 ± 2,1	95,5 ± 1,5
	Test	21,4 ± 4,4	95,4 ± 4,5

Table 7.3: PPV and NPV scores obtained for the PRESEV dataset using WML method.

These findings indicate that while the WML method seems to enhance NPV and PPV scores individually, it is unable to simultaneously improve both NPV and PPV.

7.3 Conclusions

We have successfully integrated watershed transformations into metric learning methods via the WML and WMLI algorithms. However, while the combination of metric learning and oversampling can be advantageous in specific cases, it is important to note that there is no unique solution. This conclusion, which has been established for oversampling methods (cf. article [3]), can now be extended to various preprocessing techniques. Although preprocessing methods are largely used to address the challenges of imbalanced prediction, it might be worth considering the development of algorithms that work directly on imbalanced datasets without requiring any preliminary modifications. While one option might be to create a neural network algorithm based on watershed transformation, similar to the approach in the article [1], this could cause problems in cases of small datasets with limited features and samples, as the PRESEV one. Overfitting could become a significant problem in such cases. For this reason, we investigate various other hierarchical algorithms designed to deal directly with imbalanced datasets, which can be found in the appendix section.

References

[1] A. Challa et al. “Triplet-Watershed for Hyperspectral Image Classification”. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2021).

[2] G. Chierchia and B. Perret. “Ultrametric Fitting by Gradient Descent”. In: *Neural Information Processing Systems* (2019).

[3] A. B. Hassanat et al. “Stop Oversampling for Class Imbalance Learning: A Critical Review”. In: *Digital Object Identifier* (2022).

[4] S. Ruder. “An overview of gradient descent optimization algorithms”. In: *Machine Learning* (2017).

[5] K. Weinberger and L. Saul. “Distance metric learning for large margin nearest neighbor classification”. In: *JMLR* 10 (2009).

Conclusions & Perspectives

Acute Chest syndrome (ACS) is a severe and lethal complication of sickle cell disease, often occurring around 2.5 days after the patient is admitted to hospital. The two-part PRESEV study - PRESEV1 and PRESEV2 - aimed to investigate ACS and its prediction using biomarkers in emergency department patients. The main aims of these datasets were twofold: firstly, to predict patients likely to develop ACS with a high negative predictive value (NPV) in order to improve treatment and reduce the number of deaths; at the same time, to predict patients unlikely to develop ACS with a high positive predictive value (PPV) in order to improve the allocation of hospital resources and ensure possible follow-up at home. This dual approach aims to improve both patient care and the efficient use of healthcare resources.

At first, the PRESEV1 dataset was subjected to standard machine learning methods, which produced promising results. However, when these models were applied to the PRESEV2 dataset, problems of high variability and over-fitting emerged. The root cause was the significant class imbalance within PRESEV, which only included around 20% of ACS patients. To address these class imbalances, thousands of approaches, including oversampling and cost-sensitive algorithms, have been devised over time. These techniques aim to remedy the imbalanced in class distribution such as by increasing the representation of the minority class (oversampling) or by assigning different costs to the different classes (cost-sensitive learning).

Nevertheless, determining the optimal method has proven to be a challenging task, largely due to the lack of a comprehensive library or standardised approach. This dilemma is further exacerbated by the absence of identifiable models to guide method selection. For example, the imbalance ratios and the number of characteristics do not help to select the best imbalanced method. In response to this complex problem, the ImbPip pipeline was designed to provide integration of multiple methods, promoting systematic reproducibility and facilitating methodological comparisons. However, while I was working on these methodologies, a similar pipeline known as "smote_variant" was developed at the same time and subsequently published. In future, we intend to use the latter pipeline, smote_variant, in our future projects. This decision is based on its proven

effectiveness and reliability in addressing the challenge of class imbalance in datasets such as PRESEV2, with the ultimate aim of improving the robustness and accuracy of predictive models in our future research initiatives.

By applying a range of imbalance strategies to the PRESEV dataset using this last framework, it became clear that no single strategy improved PPV and NPV scores while reducing variability. This challenge prompted us to explore other approaches, which led to the integration of image segmentation concepts, in particular the watershed algorithm. The watershed algorithm, designed to detect patterns among larger backgrounds, showed promise when directly applied to imbalanced datasets. This inspired the development of WSSMOTE, an oversampling method based on the watershed algorithm. WSSMOTE has been significantly successful in the PRESEV dataset and on other real-world imbalanced dataset. Indeed, WSSMOTE significantly reduced the variability of overfitting from 13.3% when using the standard machine learning method to just 1.2%, while maintaining a high PPV value of almost 28%. In contrast, the standard oversampling method also reduced overfitting, but resulted in a drop in PPV to around 24%.

However, the application of WSSMOTE to other imbalanced datasets shows that the method is not a magic solution, as it faces similar problems to those encountered by existing imbalanced datasets. Nevertheless, the success of the watershed has encouraged the exploration of metric learning methods, particularly those based on optimization problems. Using graph theory, these approaches aim to optimize data projection for more robust clustering. We therefore designed two separate methods, WML and WMLI, based on different loss functions and a link between data points, graphs and ultrametric distances. Although we successfully demonstrated the effectiveness of the WML and WMLI methods on a wide range of imbalanced datasets, it should be noted that these methods did not provide any improvements in the case of the PRESEV dataset.

A significant limitation emerged from the dependence on graph structures, which are at the heart of the watershed algorithm. Exploring more advanced graph modeling could enhance the impact and predictive power of the watershed. Unfortunately, the relatively small size of our dataset application (only around 600 samples and 5 features) and the nature of the neural-network-based graph transformations limited this exploration, as over-fitting is highly probable.

In addition, by recognizing the limitations of existing methods, attention was to explore direct modifications of watershed (cf. Appendinx 9),in particular by going beyond the notion of a graph and tackling the notion

of a hierarchy. Several approaches were tested, such as modifying the hierarchy by elevation or marker propagation. But none produced convincing results, perhaps due to the simplicity of the methodology. However, the notion of modifying the watershed algorithm in place remained promising.

Another perspective is to exploit watershed algorithms directly, given their proven success with balanced datasets. By combining active learning and ensemble methods in a hierarchical framework, improvements could be made. This approach needs further investigation to ensure its viability.

In summary, the PRESEV study aimed to address the challenges posed by acute chest syndrome (ACS) in patients with sickle cell disease through the use of biomarkers and predictive modelling. The main objectives were to improve patient care by predicting the development of ACS with a high negative predictive value (NPV) and to optimise resource allocation with a high positive predictive value (PPV). Initially, the PRESEV1 dataset showed promise with standard machine learning methods, but when applied to PRESEV2, problems of high variability and overfitting emerged due to a large imbalance between classes. Despite thousands of methods developed to tackle this imbalanced problem, the selection of the optimal method remained difficult due to the lack of a standardised approach. The ImbPip pipeline was created to integrate several methods. While our watershed-based oversampling method, WSSMOTE, significantly improved overfitting and maintained a high PPV in the PRESEV dataset, it is not a universal solution for all imbalanced datasets.

This encouraged the exploration of metric learning methods based on optimisation problems, leading to the development of WML and WMLI. However, these methods did not provide any improvements in the PRESEV dataset, highlighting the need for an in-place algorithm that does not take into account the distribution of the data. Efforts to modify the catchment hierarchy directly have not produced convincing results, perhaps due to the simplicity of the methodology. However, the concept of modifying the watershed algorithm remains promising. Another avenue to explore is to exploit watershed algorithms directly, possibly through active and ensemble learning methods in a hierarchical framework.

In conclusion, the study of ACS in the PRESEV dataset and class imbalance have been punctuated by challenges and promising discoveries, indicating the ongoing need for innovative approaches and further research to improve predictive models for imbalanced datasets, as they have the potential to be really useful in the real world, for example for patient care in the future.

Appendices

Watershed Hierarchy & Marker Propagation for Imbalanced Data

Even if preprocessing methods are commonly used to deal with imbalanced datasets, they present several issues and are highly dependent on the distribution of the dataset. As we demonstrated earlier in chapter 5 and 7, despite our WSSMOTE algorithm showing promising results on the PRESEV dataset, it could not be successfully extended to other imbalanced datasets. To create a more universally applicable solution for different imbalanced datasets, we opted for the development of algorithm modifications.

Initially, we envisaged creating this algorithm directly on the graph, following the approach adopted for the two algorithms previously implemented. However, modifying the graph edges remains to adjusting the weights of the graph's edges.The concept of a "muddy watershed", where minima related to minority data points are reached faster than those related to majority data, seemed intriguing. However, after practical exploration, this idea proved to be closely aligned with the metric learning algorithm we had already developed. Essentially, changing speed is equivalent to changing distances.

Another idea was to explore the use of neural networks, as shown in the article [1]. Nevertheless, the size of our studied dataset (PRESEV) remained small, making this approach subject to overfitting. Consequently, we turned to our latest idea, by interpreterping a graph as a hierarchy. We decided to manipulate this hierarchy, with a specific focus on the concept of marker propagation. This chapter will focus into the development of this idea and the corresponding algorithm.

Concept Development & Algorithm Implementation

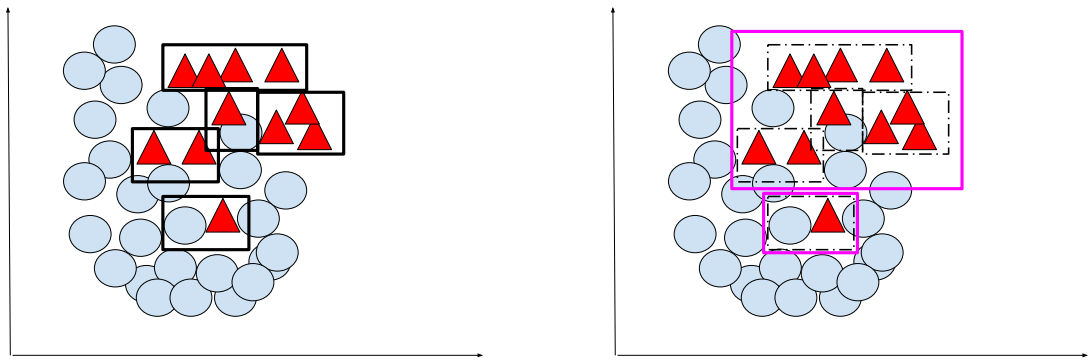
Our main objective is to design an in-place algorithm that not only exploits the data distribution, but also incorporates more information beyond the simple dependence on nearest neighbors and thus on the simple use of graph edge weights. To achieve this goal, we first considered using the concept of watershed hierarchy. A first understanding of this concept was

developed in section 4.2.

Before jumping into idea development, it is essential to recognize a fundamental distinction from other machine learning methods. This distinction results from the hierarchical, graph-oriented nature of our approach, which leads to a unique characteristic: the connection between the training (labeled data points) and testing (unlabeled data points) parts. Unlike conventional approaches, where training and predicting can be separate processes, in our context they are inseparably linked. The construction of the graph or hierarchy requires the simultaneous performance of the fitting and predicting tasks, as the graph and thus the hierarchy are build on the training and the testing data points, The primary graph or the primary hierarchy is build using all the data points.

Initial Approach: Watershed Hierarchy for Imbalanced Data

The initial approach we thought about is centered on the observation that data points belonging to the minority class often tend to be far apart due to their sparsity. Therefore, clustering techniques such as the watershed cuts algorithm are likely to create small groups strongly influenced by nearest neighbors and distances between data points. To address this problem, we have thought about designing a strategy that involves increasing the size of these clusters. In doing so, we aim to amplify the impact of minority data points both on the clusters themselves and on unlabeled data points, a visual representation of which is given in figure 5.



(a) Hypothetical Clustering obtained via Watershed Methods

(b) Hypothetical Refinement Clustering obtained via Watershed and Markers

Figure 5: Illustration of hypothetical clustering results using the watershed method. The purpose of this visual representation is to highlight that by using markers and fine-tuning the clusters, the influence of minority data points can be amplified, leading to a potential improvement in predictions. Red triangles represent minority data points, while blue circles represent majority data points. Rectangles reprensent the different clusters.

This concept of reinforcing the influence of selected clusters is similar

to the notion of watershed hierarchy by attribute. Similarly, in scenarios where the watershed algorithm is used for image segmentation, the resulting segmentation can sometimes be too complex. Using attributes or markers allows us to focus on specific regions or aspects, resulting in more meaningful and larger segmentation results, as illustrated in article [3].

A prototype of the algorithm we are exploring involves the following steps:

1. **Graph Construction**: Construction of an edge-weighted graph using labeled training data points and unlabeled test data points.

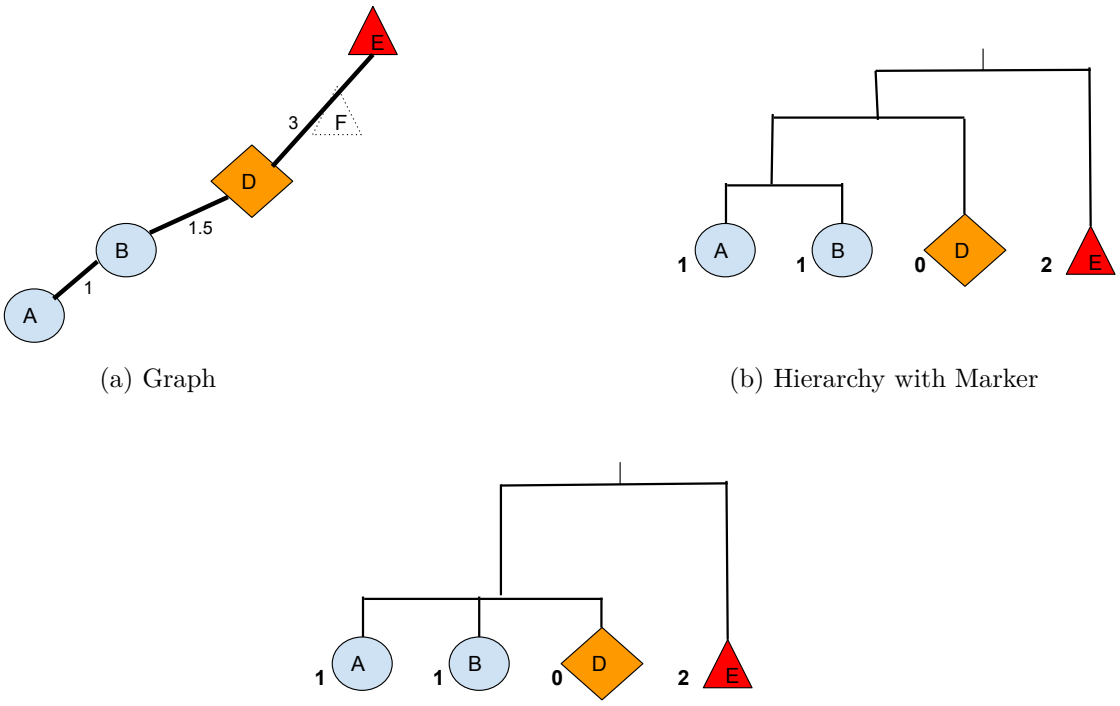
2. **Hierarchy and ultrametric distance**: Build a hierarchy and calculate ultrametric distance using graph structure and edge weights.

3. **Marker generation**: Generate markers based on the specific characteristics of data points. For example, in the figure bellows, markers are defined according to class proportions. Data points of the minority class are assigned markers (r_{min}) calculated as the ratio between the number of data points of the minority class in the training data and the total number of training data points. Whereas, data points of the majority class are assigned markers (r_{maj}) calculated as $1 - r_{min}$. For unlabeled data points, the marker is determined as the average of the markers among neighboring data points in the hierarchy.

4. **Marker propagation and ultrametric updating**: propagate these markers into the hierarchy, using the attribute-based watershed hierarchy algorithm available in Higra(cf. `https://higra.readthedocs.io`).

5. **Update graph edge weights and predictions**: use the generated ultrametric distances as new graph edge weights. These updated weights will serve as the basis for the application of the watershed cuts algorithm or watershed for semi-supervised one.

However, when we tested this algorithm on small datasets, we discovered that the problem we had initially identified was not the main challenge (cf. figure 6). Indeed, because unlabeled data points are already incorporated into the graph and hierarchy, modifying refinement using attributes may not produce significant changes, especially when unlabeled data points are already in a majority cluster. It is not always possible to modify the hierarchy. Changes impact more importantly the regions closest to the roots of the hierarchy rather than the leaves.



(a) Graph

(b) Hierarchy with Marker

(c) Watershed by Attribute

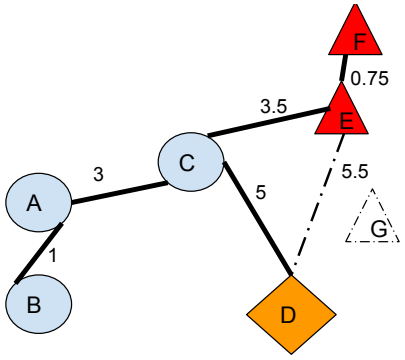
Figure 6: A hypothetical simulation demonstrating the limitations of watershed hierarchies by attribute when dealing with unlabeled data points at a lower level of the hierarchy. In this context, the majority class is represented by blue circles, minority data points by red triangles and unlabeled data points by orange squares. The dotted line triangles represent a fictive minority data points - and the lack of information introduced by the imbalanced structure of the data. The weights of the graph's edges are determined by Euclidean distances, while the markers are derived from the labels assigned to the data points.

Despite this limitation, the concept of using markers in the hierarchy remains attractive, as it allows the inclusion of more diverse information than simple Euclidean distances between data points. This observation inspired the development of another idea based on marker propagation.

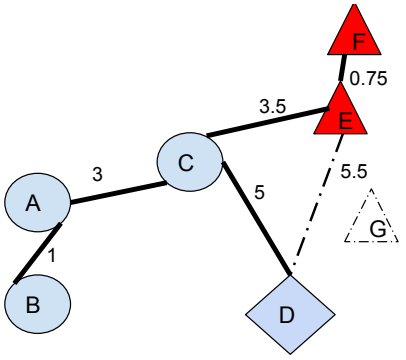
Rectified Approach: Marker Propagation for Imbalanced Data

When developing WSSMOTE and WML, we focused on nearest neighbors and distance-based considerations. However, we recognized the potential value of incorporating additional information such as point density and class distribution. Introducing attributes, or markers, into the hierarchy allowed us to explore these dimensions. In addition, this marker propagation approach means that we no longer depend exclusively on supervised clustering algorithms such as Watershed Cuts, but can instead build an in-place algorithm fully rooted in the structure of the hierarchy.

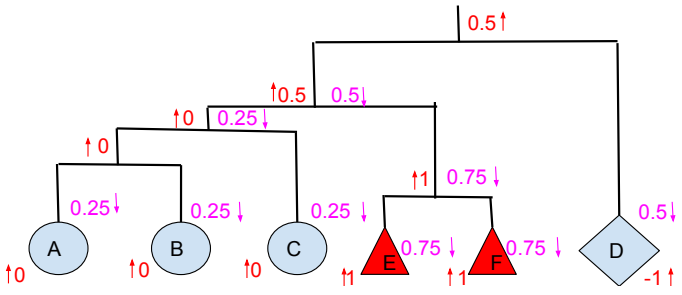
To illustrate the central concept of the algorithm we have formulated, we can consider a simple example as in Figure 7. We have chosen the most elementary form of marker - a label: 1 for the minority class, 0 for the majority class and -1 for unlabeled points. Using an average label propagation function, we expand these labels from leaves to roots and vice versa.The first expansion mirrors the accumulator principle used in the watershed hierarchy with attribute, disseminating information and creating a marker through the hierarchy. The second expansion, however, is key. It extends the prediction to not only be defined by the immediate region of the point, but also by the surrounding regions. This expansion recognizes that a point's labeling is influenced not only by its own region, but also by neighboring regions. Consequently, the pink results in the figure indicate probabilities that can be used for predictions.



(a) Graph



(b) Results with Watershed Cuts



(c) Results with Marker Propagation

Figure 7: A hypothetical simulation to illustrate the idea of marker propagation. In this context, the majority class is represented by blue circles, minority data points by red triangles and unlabeled data points by orange squares (which need to be predicted as a minority data points). The dotted line triangles represent a fictive minority data points - and the lack of information introduced by the imbalanced structure of the data. The weights of the graph's edges are determined by Euclidean distances, while the markers are derived from the labels assigned to the data points. Figure 7b shows that using Watershed Cuts, the unlabeled orange data point will wrongly be predicted as a majority data points. Where as the propagation marker method in Figure 7c considered the unlabeled data point as minority one.

The key idea is to propagate markers through the hierarchy, consid-

ering both upward and downward directions, and calculate probabilistic values for class prediction based on these markers. By propagating markers through the hierarchical structure of data points, taking into account both their immediate neighbors and their surrounding regions, the algorithm aims to create a more complete understanding of the relationships and influences between data points. This approach calculates probabilistic values that can be used for class prediction, taking into account not only the point's own attributes, but also the attributes of neighboring points. Instead, it exploits the hierarchical structure and additional information provided by markers to create a more nuanced and accurate prediction mechanism that incorporates both local and global contextual information.

Let's denote M as a marker. Below is the pseudocode representation of our algorithm (Algo. 7):

Algorithm 7: Marker Propagation

Data:

- H: hierarchy obtained using the complete set of data points
- M: markers

```
for Each leaves do
    Assign its marker;
for Leaves to Root do
    /* This step ensures that marker information is
       spread upwards in the hierarchy.                */
    Propagate the M using a label_propagation function;
for Root to Leaves do
    /* This step extends the marker information back down
       the hierarchy.                                   */
    Propagate the M using a label_propagation function;
```

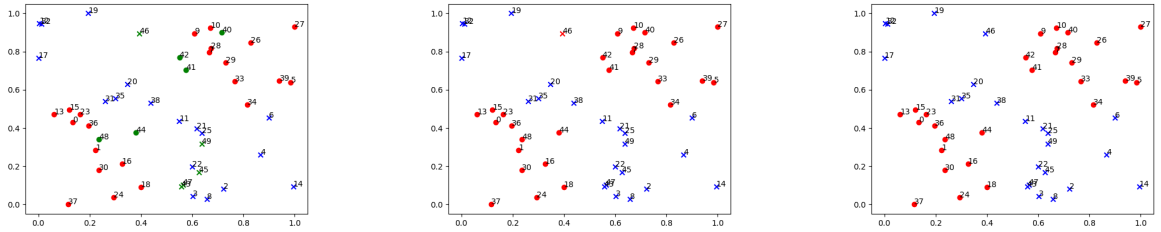
Result: Probabily output

Note that the M marker must actually produce probabilistic values between 0 and 1 to guarantee point class prediction at the end.

Results & Discussion

We have conducted a series of experiments to evaluate the performance of our algorithm on various small 2D datasets that are visually interpretable (Figures 8, 9 and 10). To comprehensively evaluate its impact, we used different types of markers. These markers were derived either from labels, similar to the initial example on which our algorithm was built, or based on point density, which takes into account factors such as distance and the number of neighboring points. Our evaluation consisted in comparing the

results generated by our algorithm with those produced by the Watershed Cuts method. The results were interesting: in some scenarios, our algorithm improved prediction accuracy. However, this trend was not constant in all cases, resulting in a mixed performance.

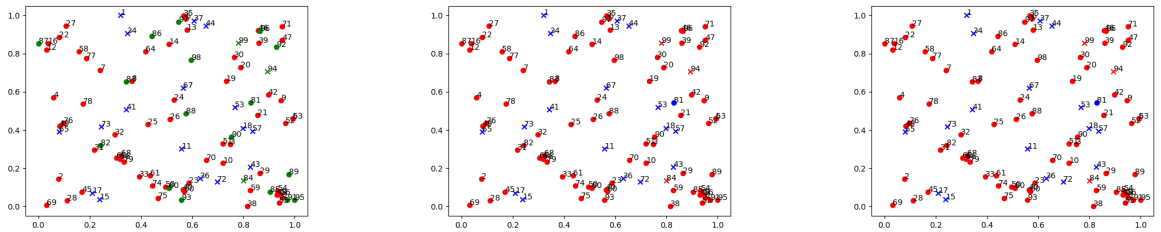


(a) Real Cases

(b) Predictions using Watershed Cut

(c) Predictions using Marker Propagation

Figure 8: Comparison between Watershed Cuts and Marker Propagation Algorithm in a Imbalanced Dataset - First Example. In green unlabeled data points. An improvement of the prediction can be noticed by the use of Marker Propagation Algorithm.

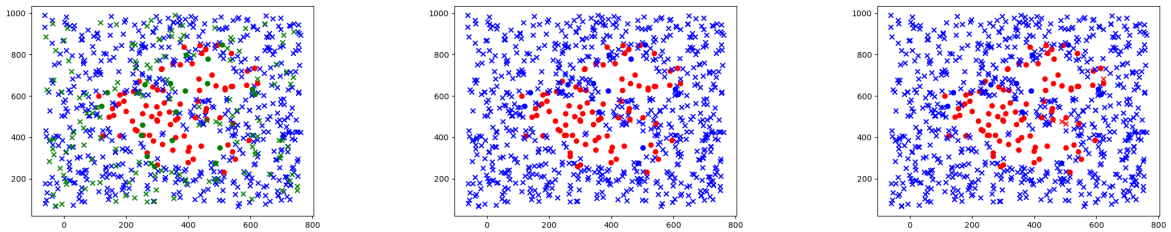


(a) Real Cases

(b) Predictions using Watershed Cut

(c) Predictions using Marker Propagation

Figure 9: Comparison between Watershed Cuts and Marker Propagation Algorithm in a Imbalanced Dataset - Second Example. In green unlabeled data points. Both methods gives the same results.



(a) Real Cases

(b) Predictions using Watershed Cut

(c) Predictions using Marker Propagation

Figure 10: Comparison between Watershed Cuts and Marker Propagation Algorithm in a Imbalanced Dataset - Paw Example. In green unlabeled data points. An improvement of the prediction can be noticed by the use of Marker Propagation Algorithm, but only in the majority class.

The variability in performance could probably be attributed to the relatively simplistic way in which our algorithm propagates markers through the data. To remedy this, we explored the possibility of refining regions by combining marker propagation and watershed hierarchy based on attribute. Unfortunately, this combination failed to deliver consistent improvements.

Thus, even if, the concept behind our algorithm is encouraging, there is still considerable potential for improvement. In particular, the implementation of a more complex probabilistic propagation and probabilistic process, inspired by the principles outlined in the article [2], could lead to better results.

In addition, we recognized that our current algorithm is limited by the use of a single marker. This limitation could potentially be avoided by adopting an ensemble-based approach, where multiple markers are used in combination to create a more comprehensive and nuanced prediction mechanism.

References

[1] A. Challa et al. "Triplet-Watershed for Hyperspectral Image Classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2021).

[2] G. Barbosa da Fonseca et al. "Fuzzy-marker-based segmentation using hierarchies". In: *Discrete Geometry and Mathematical Morphology: First International Joint Conference* (2021).

[3] D. S. Maia et al. "Recognizing hierarchical watersheds". In: *Discrete Geometry for Computer Imagery* (2019).