NOVA IMS Information Management School

DOCTORAL PROGRAMME

Information Management

The Role of Synthetic Data in Improving Supervised Learning Methods

The Case of Land Use/Land Cover Classification

João Pedro Martins Ribeiro da Fonseca

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor in Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

[this page should not be included in the digital version. Its purpose is only for the printed version]

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

The Role of Synthetic Data in Improving Supervised Learning Methods: The Case of Land Use/Land Cover Classification

by

João Pedro Martins Ribeiro da Fonseca

Doctoral Thesis presented as partial requirement for obtaining the PhD in Information Management

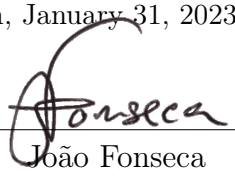
Supervisor: Professor Fernando Bação, PhD

January 2023

Statement of Integrity

I hereby declare having conducted this academic work with integrity. I confirm that I have not usedplagiarism or any form of undue use of information or falsification of results along the process leadingto its elaboration. I further declare that I have fully acknowledge the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Lisbon, January 31, 2023



João Fonseca

Dedication

To my parents, José Manuel and Maria de Lurdes Fonseca.

Acknowledgements

MIT Portugal - funding

Fernando Bacao - advisor

Georgios Douzas - discussions, tips and help on technical implementations throughout the research work

English professor from NOVA IMS - proofreading multiple chapters of this dissertation

My parents, José Manuel and Maria de Lurdes, my siblings, Hugo and Inês, my grandmother, Teresa, my girlfriend, Yasmina El Fassi and my brother-in-law, Hugo Nunes.

My friends, Pedro Rodrigues, Francisco Martins, Miguel Meco, Andrew Bell, Rafaela Henriques, Margarida Saragoça, Manvel Khudinyan, Francisco Braga, Miguel Portas, Mari Reyes, Ana Vieira, Ana Correia, Oumaima Derfoufi

My secondary school's biology teacher, Regina Lucena, who taught us life skills well beyond the courses' syllabi, and tremendously inspired me towards a research-oriented path (without me realising it).

Professor Marco Painho and my friends from the doctoral program, Vicente Tang, Maria Anastasiadou and Darina Vorobeva

Abstract

In remote sensing, Land Use/Land Cover (LULC) maps constitute important assets for various applications, promoting environmental sustainability and good resource management. Although, their production continues to be a challenging task. There are various factors that contribute towards the difficulty of generating accurate, timely updated LULC maps, both via automatic or photo-interpreted LULC mapping. Data preprocessing, being a crucial step for any Machine Learning task, is particularly important in the remote sensing domain due to the overwhelming amount of raw, unlabeled data continuously gathered from multiple remote sensing missions. However a significant part of the state-of-the-art focuses on scenarios with full access to labeled training data with relatively balanced class distributions. This thesis focuses on the challenges found in automatic LULC classification tasks, specifically in data preprocessing tasks. We focus on the development of novel Active Learning (AL) and imbalanced learning techniques, to improve ML performance in situations with limited training data and/or the existence of rare classes. We also show that much of the contributions presented are not only successful in remote sensing problems, but also in various other multidisciplinary classification problems. The work presented in this thesis used open access datasets to test the contributions made in imbalanced learning and AL. All the data pulling, preprocessing and experiments are made available at https://github.com/joaopfonseca/publications. The algorithmic implementations are made available in the Python package *ml-research* at https://github.com/joaopfonseca/ml-research.

Keywords: LULC classification; Active Learning; Imbalanced Learning; Synthetic Data; Oversampling;

Sustainable Development Goals (SDG):

Contents

List of Figures

List of Tables

1. Introduction

Accurate LULC maps constitute a unique resource for a variety of applications. Its applications range from environmental monitoring, land change detection, and natural hazard assessment to agriculture and water/wetland monitoring [1]. However, the production of LULC maps often require the involvement of multiple photo-interpreters with specialized skills, making the process expensive and time-consuming and unsuitable for operational LULC mapping over large areas [2]. Therefore, the manual production of LULC maps are often outdated by the time they are completed, limiting its value to the analysis past conditions in the area of interest.

An alternative method to address the limitations found in the photo-interpreted approach to produce LULC maps is automated mapping. This method uses remotely sensed data, especially multi and hyper-spectral images, to train ML algorithms to automatically detect the different LULC classes. To do this, the usage of recent supervised learning techniques seems to be a promising path to achieve reliable and updated maps [3]. However, the practical application of this approach is hampered by various limitations:

1. Human error. The quality of the classifiers produced is heavily dependent on the quality of its training data. In this case, the training dataset is extracted from manually labeled land cover patches using typically satellite imagery. The target LULC map's minimum mapping unit, as well as the quality of orthophotos and satellite images being used are some of the factors that may lead to label noise in the training data [4].

2. High dimensionality. The high dimensionality of multi and hyper-spectral images contain useful information to improve ML classification tasks. However, it also introduces an additional layer of complexity and redundancy in classification [5]. If the training data is not large enough, it may cause the classification task to be affected by the curse of dimensionality.

3. Class separability. Some LULC classes sometimes contain overlapping spectral signatures which makes the separation of the two classes particularly challenging in some cases [6].

4. Infrequent LULC classes. Depending on the region of study, some land cover classes may be more or less frequent when compared to other regions [7]. However, the accurate identification of rare classes is often equally or more important than the identification of the remaining classes. This problem is known, in ML community, as imbalanced learning. As an example, the classification of a desert region with 3 classes of interest, bare soil, urban and water, would be an imbalanced learning problem. In this case, a classifier that would predict bare soil for the entire area of study would have very high overall accuracy scores but would not be useful.

5. Scarcity of labeled data. The increasing number of remote sensing missions in the past decades are generating large amounts of high quality data. However, only a small portion of this data has some sort of LULC ground truth and can be used for supervised learning. In these scenarios the production of automated LULC maps is particularly challenging and involves the usage of techniques that are able to leverage information from both labeled and unlabeled data and simultaneously maximize the value of the data annotation process [8].

The latter two challenges, imbalanced learning (*i.e.*, infrequent LULC classes) and scarcity of labeled data, are the focus of this thesis. However, these two challenges may be addressed in various different ways.

The asymmetry frequently found on LULC class distributions affects the performance of ML classifiers. In this scenario, during the ML classifier's learning phase, the minority classes contribute less to the minimization of accuracy, the typical objective function, biasing the classifier towards the most frequent classes. The possible approaches to deal with imbalanced learning can be divided into three main groups [9]:

1. Cost-sensitive solutions. These methods use a cost matrix to adjust the misclassification cost to benefit the minority classes.
2. Algorithmic-level solutions. These methods introduce algorithmic solutions to improve the learning process on the minority classes.
3. Resampling solutions. They modify the training data to balance the class distribution by removing observations from the majority class(es) or adding observations belonging to the minority class(es).

The latter set of methods, resampling solutions, benefit from their simplicity. This type of approaches do not require domain knowledge to define a cost matrix and dispense the usage of specialized classifiers. These methods can be further divided into (1) Oversampling, where an algorithm generates artificial observations belonging to the minority class, (2) Undersampling, where an algorithm removes observations belonging to the majority class or (3) Hybrid approaches, which combine oversampling and undersampling together. In this thesis, we will focus on oversampling methods and generalize these as a corner case of augmentation methods for tabular data in later chapters. In Section 1.1 this topic is discussed to a greater extent.

The second challenge this thesis focuses on is the implementation of useful ML algorithms in environments with scarce availability of labeled data. There are three different types of techniques to address this problem, which fall in between supervised and unsupervised learning:

1. Semi-supervised Learning. Uses both labeled and unlabeled data in the training phase to improve the classifier's performance [10]. It pushes the decision boundaries of classifiers to regions with lower density of observations while maximizing the performance over the labeled training dataset [11].
2. Self-supervised Learning. Uses unlabeled data to perform secondary/pretext tasks to learn representations of the input space [12]. These methods typically use neural network architectures [13].
3. Active Learning. Iteratively samples the most informative/representative observation out of a pool of unlabeled data in order to be labeled and included into the training dataset [14]. This approach attempts to optimize the classifier's performance with as least data as possible.

The latter method, Active Learning, benefits from the possibility of including different techniques into its pipeline, including the former two. It is therefore one of the main focus of this thesis. In Section 1.2 this topic is discussed to a greater extent.

1.1. Data Augmentation

Data Augmentation methods expand the training dataset by introducing new and informative observations [15]. The production of artificial data may be done via the introduction of perturbations on the input [16], feature [17] or output space [15]. Data Augmentation methods may be divided into Heuristic and Neural Network-based approaches [18]. In addition, they may also be distinguished based on its data generation policy, whether local (considers a local/specific subset of the dataset) or global (considers the overall distribution of the training dataset). Figure 1.1 shows the general taxonomy of Heuristic Data Augmentation methods. Finding the appropriate Data Augmentation method generally depends on

the domain [17], although some studies discuss which methods are more appropriate according to the domain [18, 19, 20].

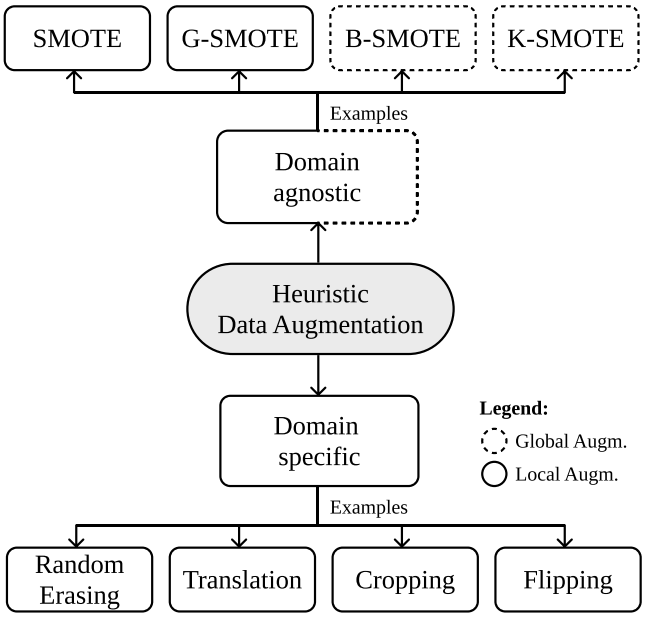


Figure 1.1.: Schema containing a general Heuristic Data Augmentation taxonomy.

Heuristic approaches attempt to generate new and relevant observations through the application a predefined procedure, usually incorporating some degree of randomness [21]. Since these methods typically occur in the input space, they require less data and computational power when compared to Neural Network methods. Neural Network approaches, on the other hand, map the original input space into a lower-dimensional representation, known as the feature space [17]. The generation of artificial data occurs in the feature space and is reconstructed into the input space. Although these methods allow the generation of less noisy data in high-dimensional contexts and more plausible artificial data, they are significantly more computationally intensive. Considering the scope of this thesis, the computational power available for this experiment and the breadth of datasets used in the different experimental procedures, we will focus on domain-agnostic heuristic data augmentation methods.

While some techniques may depend on the domain, others are domain-agnostic. For example, Random Erasing [16], Translation, Cropping and Flipping are image data-specific augmentation methods. Other methods, such as most of the variants of the Synthetic Minority Oversampling TEchnique (SMOTE) [22], may be considered domain agnostic. However, SMOTE methods were originally developed as oversamplers, whose goal is to balance the class frequencies of the target variable in the training dataset and address the class imbalance bias. Therefore, oversampling methods may be considered a subset of Data Augmentation. Data Augmentation strategies may follow varying augmentation strategies, which does not necessarily depend on the target class distribution. An example of the differences among general data augmentation and oversampling generation strategies is shown in Figure 1.2.

The simplest approach found in the literature is randomly duplicating existing training observations. As a non-informed data generation method, although simple to implement, it increases the risk of overfitting and generally performs worse than other informed heuristic methods [2].

The SMOTE method generates artificial data via the linear interpolation between a random observation and one of its k-nearest neighbors (also randomly selected) [22]. Although simple and effective, it also contains several limitations which motivated the development other variants, discussed below. Specifically, its selection mechanism does not consider the global structure of the dataset while its generation mechanism

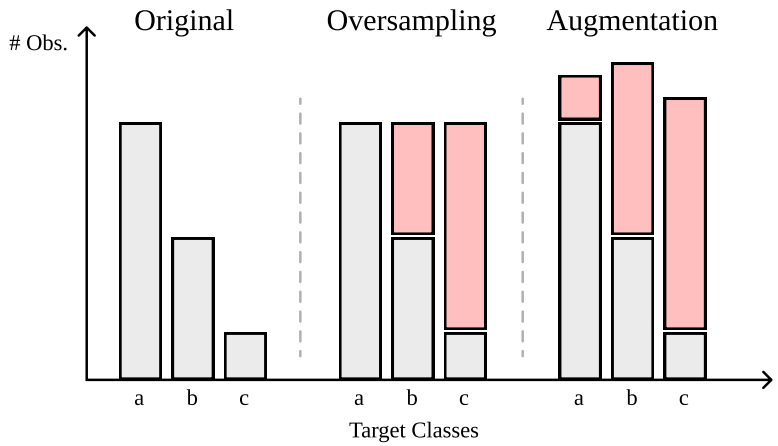


Figure 1.2.: Examples of data augmentation Strategies. The salmon-colored bars represent artificial data using the normal oversampling (center group) and an example of augmentation (right group) strategies.

introduces little variability into the training dataset [23]. Borderline-SMOTE (B-SMOTE) [24] improves the selection mechanism by attributing a larger importance to the observations closer to the decision boundaries. The selected observations are used to run the SMOTE method in order to produce better defined decision boundaries. A more recent improvement of the selection mechanism is K-means SMOTE (K-SMOTE) [25]. This method uses a clustering-based approach to overcome imbalances between and within classes, while considering the densities of each region of the input space.

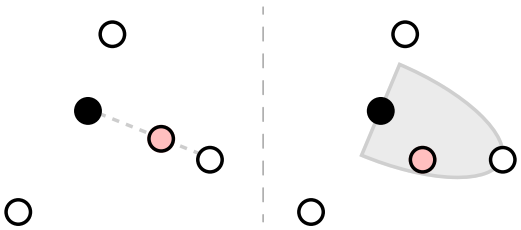


Figure 1.3.: Examples of data generation using SMOTE and G-SMOTE. In this example, both G-SMOTE's deformation and truncation parameters assume values around 0.5.

G-SMOTE [23] modifies SMOTE's generation mechanism. Instead of generating an observation as a linear combination between 2 others, it generates observations within an hypersphere defined using the selected observation as its center and one of its nearest neighbors as its boundary. The hypersphere contains two hyperparameters, the truncation and deformation factors, which limit the area of the hypersphere. The difference between SMOTE and G-SMOTE is shown in Figure 1.3.

1.2. Active Learning

Supervised ML algorithms typically perform well in contexts where labeled data is abundant and accessible. However, in a practical setting, finding this data is frequently a challenging task. Depending on the domain, collecting large volumes of data may not be feasible since the labeling of such data becomes labor and time intensive and may involve domain experts throughout the process [26]. AL maximizes a

classifier's performance while annotating as least observations as possible. It assumes that observations within the same dataset have a different contribution to the training of ML classifiers [27]. Consequently, the data annotation cost can be minimized via the annotation of the most valuable observations within an unlabeled input space. The goal is to iteratively maximize the classification performance of ML algorithms while minimizing the required amount of training data to reach a certain performance threshold [28]. It allows the implementation of ML classifiers with a good performance and minimal effort when compared to randomly selecting data or labeling the entire unlabeled dataset [29]. Therefore, it addresses the labeling problem in scenarios with a limited budget, time, or availability of labeled data.

AL methods may be divided into 2 different stages, initialization and iteration. Figure 1.4 shows a diagram that represents the typical AL initialization. Assuming the AL task is initialized without any previously labeled data, it is typically composed of 3 steps [30]:

1. Collection of an unlabeled dataset, where the procedure depends on the domain of application.
2. Selection of an initial data subset. Typically, when there is no a priori labeled dataset, the initial data subset is randomly picked from the unlabeled dataset.
3. Data labeling. The supervisor is presented with the data subset, where its goal is to label each observation. Some of the research refers to the supervisor as the oracle [31, 32].

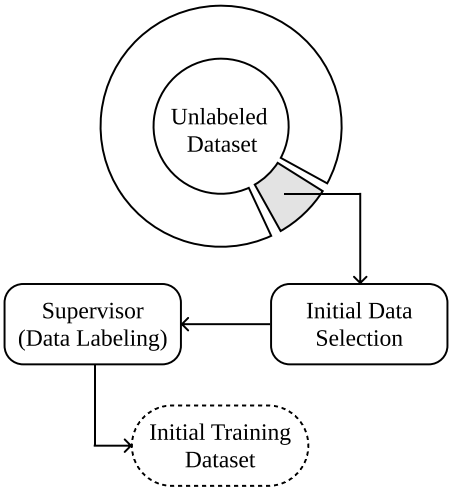


Figure 1.4.: Diagram depicting an AL initialization.

Once an initial training dataset is set up, the iterative process of AL takes place. An AL iteration is completed once a new batch of labeled data is added to the training dataset. A standard AL process is shown in Figure 1.5 and is composed of the following steps [33, 34]:

1. Setting up a classification algorithm and uncertainty criterion. The classifier is trained using the labeled dataset (*i.e.,* the Current Training Dataset), and is used to predict the class membership probabilities of the observations found in the unlabeled dataset. The class probabilities are passed into an Uncertainty Criterion, which will return the classification uncertainty of the classification algorithm for each unlabeled observation. The combination of the classifier, along with the uncertainty criterion is sometimes referred to as the Query/Acquisition function [35].
2. Selecting the top N observations. Since it is not possible to determine a priori whether the classifier's prediction is correct or not, the N observations with highest uncertainty may have been unknowingly correctly classified. However, regardless of the classification quality, these observations are expected to provide the most meaningful information to train the classifier in the next iteration.

3. Labeling the selected N observations and updating the current training dataset with the new training observations. The selected observations from the unlabeled dataset are presented to the supervisor, which is responsible for manually labeling the observations. The new (labeled) training observations are added to the training dataset and the iteration is completed.

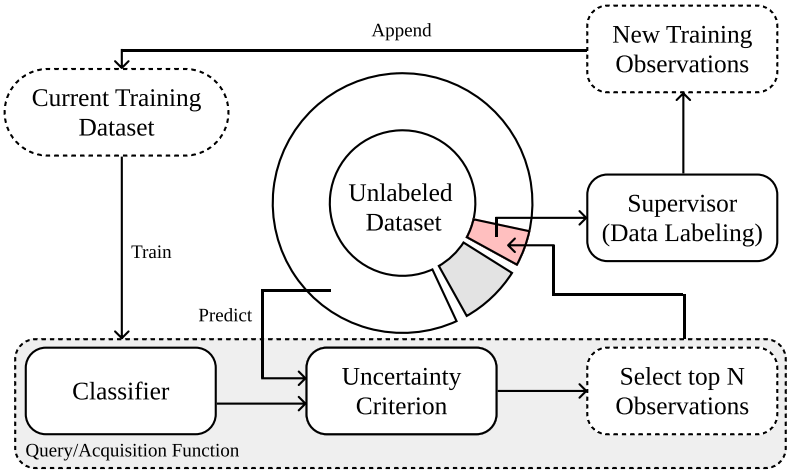


Figure 1.5.: Diagram depicting an AL iteration. In the first iteration, the training set collected during the initialization process becomes the "Current Training Dataset".

Two common challenges found in AL implementations is the consistency and efficiency of AL in practical scenarios [36]. On the one hand, the consistency problem refers to the high variance in performance (regarding classification and data selection) over different initializations (*i.e.,* different initial training datasets) of active learners. On the other hand, the efficiency problem refers to the maximization of the quality of the collected data over a run. Therefore, a good active learner is capable of having a consistent performance over different initializations while ensuring the production of high-performing classifiers with the least possible amount of data. There are various factors that may affect the consistency and efficiency of the AL framework: (1) Human error during data labeling [37], (2) Non-informative initial training dataset [38] and (3) Lack of an appropriate uncertainty criterion [35]. AL research has typically been focused on the specification of uncertainty criteria, as well as domain-specific applications. Query functions can be divided into two different categories [39, 40]:

1. Informative-based query strategies. These strategies use the classifier's output to assess the importance of each observation towards the performance of the classifier. These strategies focus on quantifying the class uncertainty of the unlabeled observations. Since these techniques do not account for the relationships between the unlabeled observations and treats each observation independently [41].

2. Representative-based query strategies. These strategies estimate the optimal set of observations that will optimize the classifier's performance. This strategy contains 3 main approaches: Density-based, Diversity-based and Exploration of graph structures. Although this method addresses the problem of sampling bias and redundant instance selection, these strategies typically require more observations in order to reach the desired classification performance [40].

Although there are significant contributions towards the development of more robust query functions and classifiers in AL, modifications to AL's basic structure is rarely explored. In [31] the authors introduce a loss prediction module in the AL framework to replace the uncertainty criterion. This model implements a second classifier to predict the expected loss of the unlabeled observations (using the actual losses collected during the training of the original classifier) and return the unlabeled observations with the highest expected loss. Although this contribution is specific to neural networks (and more specifically, to

deep neural networks), they were able to significantly improve the efficiency of data selection in AL. In [8] the authors propose the usage of semi-supervised learning during both the initialization of the AL and the iterative process as well. However, this method was proposed specifically for deep learning applications.

A query strategy/function encompasses all the steps prior to the data labeling within an AL iteration. They focus on finding the observations' informativeness, representativeness or both [39, 40]. Representative query strategies are generally less efficient in data selection than Informative query strategies [40]. However, recent research often use representative approaches alongside informative approaches [39, 42]. Representative query strategies are explored via 3 main approaches [40]:

1. Density-based, which select representative observations from high density regions. [43, 44, 45] used a density-based approach using clustering algorithms to select the observations closest to the centroid of each cluster.
2. Diversity-based, which select the N observations at each iteration that maximize the diversity in the training data. The diversity-based approach was developed to avoid the selection of redundant observations in batch-mode learning [46].
3. Graph-based, which find the most representative nodes and edges of a graph network [47]. Since these methods are specific to graph network data, they have a more limited applicability.

Informative query strategies, unlike representative query strategies, do not account for the structure of the unlabeled dataset. As a result, this type of strategy may lead to the inefficient selection of observations (*i.e.,* redundant observations with similar profiles) [40]. Research on more robust selection criteria attempts to address the efficiency problem. This is motivated by the importance of the selection criteria in AL's iterative process [35]. Specifically, Settles [48] observed that in some datasets informative query strategies fail to outperform the random selection of observations. Generally, the Random Selection query method is used as a baseline. This method disregards the class membership probabilities produced by the classifier and returns N random points from the dataset without following any specific criteria.

A frequently used query strategy is Uncertainty Sampling, originally proposed in [49]. Using this method, the estimation of an observation's uncertainty is based on the target class with the highest probability (p_a, according to the classifier) and the uncertainty is calculated as $1 - p_a$. However, since this method dismissed the classifier's predictions on the remaining labels, the Breaking Ties criterion was proposed to address this limitation for multiclass problems [50]. This method uses the two target classes with highest probability (p_a and p_b, according to the classifier) and the uncertainty is calculated as $p_a - p_b$ (in this case, the lower the output value, the higher the uncertainty). Recent variants of the Breaking Ties criterion, such as the Modified Breaking Ties, attempted to fix some limitations of the original method [51, 52].

Another common informative query strategy is the calculation of Shannon's Entropy. This metric measures the level of uncertainty based on the probabilities of a set of possible events. Its formula is given by $H(p) = -\sum_{i=0}^{n} p_i \log_2 p_i$, having p as the set of probabilities of all target classes. The application of the Entropy uncertainty criterion is also frequently applied in Deep Active Learning [32]. Other Entropy-based methods were also developed for more specific applications. For example, an ensemble querying approach known as Entropy Querying-by-Bagging uses the predictions of all estimators to find the maximum entropy of each observation [53].

The Query by Committee (QBC) strategy was developed to address ensemble classifiers. It is a disagreement based strategy attempts to maximize the information gain at each iteration by computing the disagreement of the predictions over the estimators that form the ensemble. The Entropy Querying-by-Bagging and Query-by-Boosting methods are also ensemble strategies. Query by boosting and bagging methods were found to achieve a good performance over various datasets [54], while the performance between the two strategies appears to differ significantly across various scenarios [55].

Other classifier-specific query strategies were also developed for different applications. However, these methods have the disadvantage of depending on the classifier being used. For example, Margin Sampling is a well studied strategy that uses a Support Vector Machine as its classifier in order to select the unlabeled observations closest to its decision boundaries [40]. Although, since this method is known to lead to the excessive selection of observations in dense regions [56], it was improved in various ways. In [56] the authors extend this strategy by applying the manifold-preserving graph reduction algorithm beyond the normal Margin Sampling method.

1.3. Research Questions

The main research questions (RQ) and goals of the work plan are the following:

1. What are the main research lines in data augmentation?
    - Development of a literature review to study existing data augmentation methods and the core fields where they are being used.
2. How can automatic LULC mapping achieve an increased and consistent quality?
    - Exploration of imbalanced learning methods in the context of LULC.
3. How to perform automated LULC mapping efficiently under limited ground-truth data availability?
    - Development of an improved active learning framework in the remote sensing domain using artificial data generation.
4. How to oversample data with both continuous and categorical features?
    - Development of an improved oversampling method to be used with mixed data types.

1.4. Main Objectives

This research aims to apply and develop new data preprocessing techniques to LULC classification, with a focus on AL and oversampling techniques. The main objective is to optimize ML classifiers' performance with minimal labeled data and/or imbalanced learning scenarios. This objective was decomposed into four research questions. We start by performing a literature review of data augmentation techniques (RQ1). Second, we apply a state-of-the-art oversampling method to understand its effect in LULC classification tasks (RQ2). Third, we modify the AL framework to include a generator and optimize its augmentation policy to further reduce the amount of required labeled data for ML classifiers to reach a satisfactory performance (RQ3). Finally, we propose a new oversampling method for datasets with both continuous and categorical features (RQ4).

Each contribution towards the different RQs is divided into different studies. All chapters refer to a different RQ with the exception Chapters 4 and 5, which refer to RQ3. The future steps necessary to complete the PhD plan are described in Chapter 7. The current structure of the thesis is shown in Table 1.1.

RQ1 aims to investigate and consolidate data augmentation methods. This was done by conducting a literature review to study the main areas of application of data augmentation methods. To do this, we created a database of research papers containing the author list, year of publication, journal/conference, keywords, title, abstract and number of citation of a large number of papers related to data augmentation. This study was done with Natural Language Processing (NLP) techniques to extract topics and knowledge

graphs based on keywords to understand the different relationships among studies. This approach resulted in a compilation of the most important and current research in the field, allowing both researchers and practitioners to use this work as a starting point for their own work. Finally, the development of this work represents a crucial step towards the identification of new opportunities within the field and consequently towards the following research question and objectives. The results of this work is available in Chapter 2.

The most relevant methods found in Chapter 2 will be used for posterior work. Specifically, we address RQ2 using a heuristic data augmentation method, K-means SMOTE [25], as described in Chapter 3. Since datasets produced for LULC classification often contain irrelevant, redundant, noisy and/or unreliable data, knowledge discovery is hindered and ultimately leads to the poor training of predictors. Consequently, data preprocessing becomes an important contribution to the quality of the predictors developed. In this case, we used K-means SMOTE to oversample minority regions belonging to the same land cover class. This was motivated by the challenges faced in producing automated LULC maps using a training dataset with rare classes. In this scenario, the spectral signature of a given class often depends on its geographical distribution and the time of the year the image was captured. Cluster-based oversamplers, such as K-means SMOTE, allow for a more accurate generation of minority samples, since it can identify and isolate variations in spectral signatures within a land cover class.

At a later stage, in RQ3, we addressed a different instance of the problem of rare land cover classes in the training dataset. Specifically, in a context of limited sample-collection budget, the collection of the most informative samples capable of optimally increasing the classification accuracy of a LULC map is of particular interest [33]. Active learning attempts to minimize the human-computer interaction involved in photo-interpretation by selecting the data points to include into the annotation process. Although, current state-of-the-art techniques are mostly focused on the improvement of the acquisition function. We study this problem via the modification of the typical AL framework. We focus on the usage of data augmentation techniques and an augmentation policy optimizer to improve the quality of the data generated. These modifications are presented in Chapters 4 and 5:

- Chapter 4 introduces the generator component into the typical AL framework.
- Chapter 5 introduces the augmentation policy optimizer, while generalizing the generator component for augmentation policies beyond oversampling strategies.

One of the main limitations uncovered in RQ1 (see Section 2) is the lack of oversampling methods applicable to datasets containing categorical features. In fact, only two oversamplers were found to be capable of oversampling data with categorical features, SMOTENC [22] and random oversampling. However, in a practical setting, datasets with mixed feature types are common but the methods available are outdated. RQ4 will be addressed through the modification of a State-of-the-art oversampling method by mixing its data generation mechanism with the one found in SMOTENC. However, this work is still in progress.

1.5. Methods

The methodology used for all contributions follow a similar approach, with exception to the work presented in Chapter 2. It is composed as follows:

1. Collection of a large number of (LULC, multidisciplinary or image) classification datasets.
2. Identification of related literature and limitations to be addressed.
3. Design and implementation of contributions.
4. Definition and implementation of experimental settings.

5. Analysis of results and statistical significance testing.
6. Publish results.

The methodological approach to the proposed doctoral work is depicted in Figure 1.6. All of the work presented was developed while ensuring full reproducibility. Contributions at the algorithm-level are implemented in the Python open-source package ML-Research.

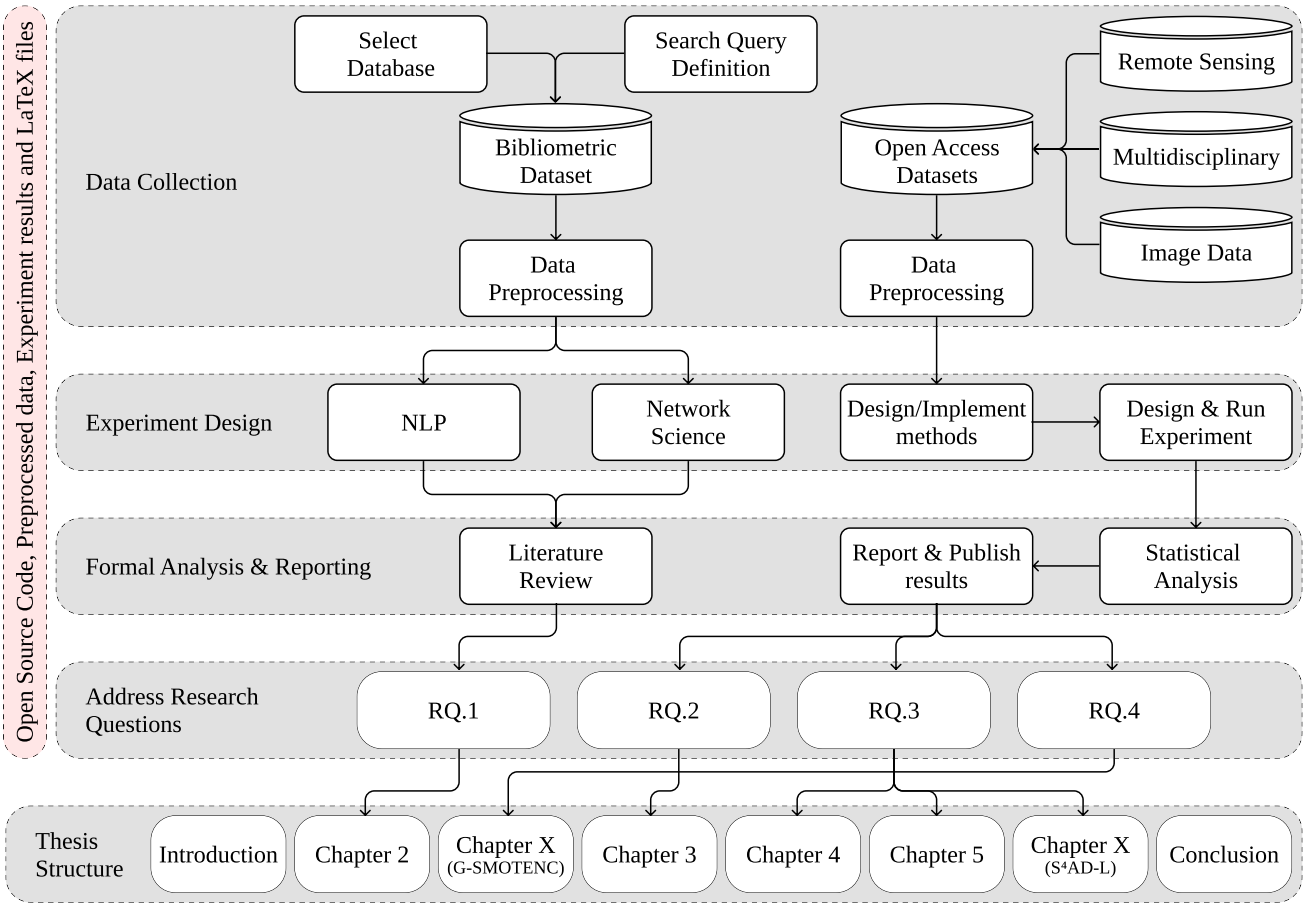


Figure 1.6.: Planned structure and methodological approach of the doctoral work.

1.6. Path of Research

The work developed is organized in different papers of related subjects, namely AL and imbalanced learning. Every submitted or published research output is available at this GitHub repository, along with the LaTeX scripts, data, source code (for data pulling, preprocessing, experiments and analysis), experiments' raw outputs and analysis outputs (figures, tables and diagrams). The current stage of each of the studies is presented in Table 1.1.

Chapter	RQ	Study Name	Current stage
2	1	Research Trends and Applications of Data Augmentation Algorithms	Under Review (Initial submission)
3	2	Improving Imbalanced Land Cover Classification with K-means SMOTE: Detecting and Oversampling Distinctive Minority Spectral Signatures	Published in the journal Information
4	3	Increasing the Effectiveness of Active Learning: Introducing Artificial Data Generation in Active Learning for Land Use/Land Cover Classification	Published in the journal Remote Sensing
5	3	Improving Active Learning Performance Through the Use of Data Augmentation	Under Review (2^{nd} round)
n/a	3	S^4AD-Learning: Self-supervised Semi-supervised Active Deep Learning	Work in Progress
n/a	4	Geometric SMOTENC: A geometrically enhanced drop-in replacement for SMOTENC	Work in Progress

Table 1.1.: Current stage of the planned studies in the scope of the doctoral program and PhD grant.

2. Research Trends and Applications of Data Augmentation Algorithms

Submitted as Joao Fonseca, Fernando Bacao, to a Q1 Journal, 2022

In the Machine Learning research community, there is a consensus regarding the relationship between model complexity and the required amount of data and computation power. In real world applications, these computational requirements are not always available, motivating research on regularization methods. In addition, current and past research have shown that simpler classification algorithms can reach state-of-the-art performance on computer vision tasks given a robust method to artificially augment the training dataset. Because of this, data augmentation techniques became a popular research topic in recent years. However, existing data augmentation methods are generally less transferable than other regularization methods. In this paper we identify the main areas of application of data augmentation algorithms, the types of algorithms used, significant research trends, their progression over time and research gaps in data augmentation literature. To do this, the related literature was collected through the Scopus database. Its analysis was done following network science, text mining and exploratory analysis approaches. We expect readers to understand the potential of data augmentation, as well as identify future research directions and open questions within data augmentation research.

Keywords: Data Augmentation; Generative Adversarial Networks; Regularization Methods; Overfitting

2.1. Introduction

The performance of Machine Learning models is highly dependent on the quality of the training dataset used [57, 58]. Specifically, the presence of imbalanced and/or small datasets, target labels incorrectly assigned, outliers and high dimensional input spaces reduce the prospects of a successful machine learning model implementation [58, 59, 60]. Even though the performance of any classifier is affected by the size of its training dataset, deep learning models have a particularly inconsistent performance over unseen datasets even when trained with large datasets [61, 62]. Conversely, deep learning models are capable of quickly adapting (and overfitting) to the training dataset, including when it contains label and/or complete pixel noise [62, 63]. Although the performance of these models can be improved through regularization methods, they are still incapable of correcting label noise in the training dataset [63].

Regardless of the machine learning model used, when the training set contains significant limitations (regarding overall quality and size), the model's performance on unseen data is generally going to be affected. Specifically, when the training data is not representative of the true population, or the model is over-parametrized, it becomes particularly prone to overfitting [64]. There are different strategies to reduce overfitting, known as regularization methods [18]. Identifying the appropriate regularization

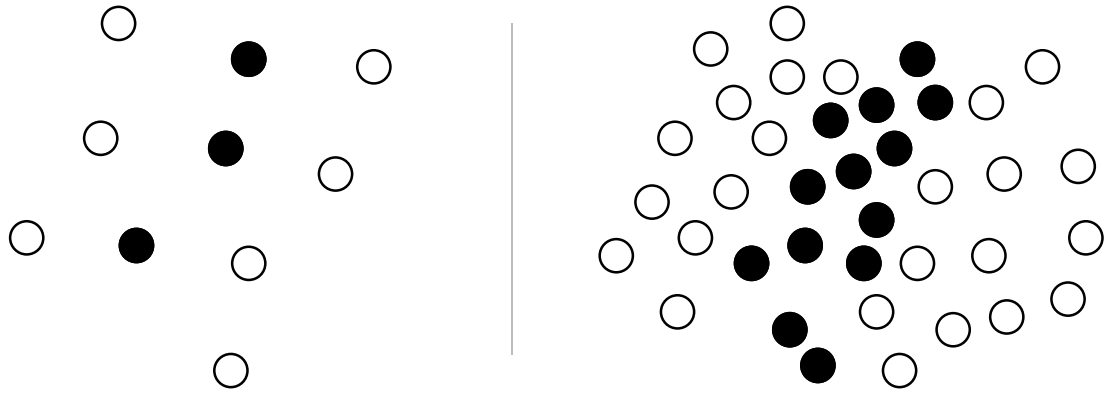


Figure 2.1.: Example of data augmentation in a 2-dimensional binary classification setting. The left pane contains the original dataset, where the amount data is scarce and the gap among the two classes are wide, allowing for greater classification variability. The right pane contains the augmented dataset, where the gap among the two classes is narrower and the decision boundaries become easier to define.

methods varies according to the use case [65]. While some methods can only be applied on specific classifiers, data types or domains, others may be applied at the data level, independently from the classification problem. For example, methods such as dropout/dilution, batch normalization and transfer learning/domain adaptation are mostly applied on neural network architectures. Pruning is applied on decision trees. Early stopping can be used on learners trained iteratively, making it a broader method.

Data augmentation techniques are used to increase the size (and hopefully the diversity) of data in a training dataset through the production of artificial observations [20, 66]. They are frequently used as regularization techniques for various types of problems and classifiers, since it is applied at the data level [15]. Figure 2.1 shows an example of data augmentation, where the decision boundaries become clearer after the original dataset is augmented. Data Augmentation methods can be divided into heuristic and Deep Learning approaches [18, 67]. Within these approaches, they may be either domain specific or contextually independent. For example, although both Synthetic Minority Oversampling Technique (SMOTE) [22] and Kernel Filters are heuristic approaches, SMOTE may be used regardless of the context, while Kernel Filters are specific to image data augmentation. The different types of Data Augmentation methods are defined at a higher detail in Section 2.2.

In 2011, Jürgen Schmidhuber's group showed that a MLP ensemble architecture can achieve state-of-the-art performance on computer vision benchmarks given strong enough data augmentation [68, 69]. Although the state-of-the-art improved since then, two recent papers developed by Google Brain and Facebook research teams support Schmidhuber's group's findings. Specifically, in [70, 71] the authors discuss two similar MLP ensemble architectures, showing that the proposed model attains a comparable performance to convolutional neural networks and attention-based networks. Another recent study also discusses a related MLP architecture with similar findings, while suggesting that the strong performance of computer vision models may be attributable mainly to the inducive bias produced by the patch embedding and the carefully-curated set of training augmentations [72].

Research on data augmentation methods gained significant popularity in recent years. As such, there were some efforts in the past to establish a taxonomy and distinction of the different types of data augmentations methods [18]. To the best of our knowledge, there is no analysis on data augmentation research as a whole, as well as domains of application and future directions. In this paper we focus on current and past research trends of data augmentation methods, its different applications and use cases.

This is done with an extensive analysis of the title, keywords and abstract of a large set of literature related to data augmentation, collected through the Scopus database using Natural Language Processing and Network Science techniques. The analysis contains 3 phases. We started by performing an exploratory data analysis to identify the most significant publications, journals and conferences within the field of data augmentation. Then, we analysed the articles' author keywords by constructing a network and extracting and identifying communities of keywords. Finally we used a text mining approach to extract additional applications and methods using the articles' abstracts, as well as validate the findings discussed with the keyword analysis.

The rest of this paper is structured as follows: Section 2.2 describes the main methods and approaches used in data augmentation, Section 2.3 describes the procedures defined throughout the different analyses. Section 2.4 presents and discusses the findings drawn from the analyses, as well as research gaps and open questions in data augmentation research. Section 2.5 summarizes the main findings discussed throughout the study.

2.2. Data Augmentation Methods

Based on the literature found, a Data Augmentation method may be characterized based on 3 criteria. The more common division is done between Heuristic and Deep Learning approaches [18]. Within these, several approaches have been developed to produce artificial observations at the input [16], feature [17], or output space [15]. Finally, we also distinguish them based on whether their generation mechanism considers local (*i.e.,* considers partial/specific information within the dataset) or global (*i.e.,* it's based on the overall distribution/structure of the dataset) information of the original dataset. Figure 2.2 depicts the concept map with the different subdivisions of characteristics of data augmentation methods. In this section, the analysis of the different types of data augmentation will be based on their architectural approach. However, all the methods mentioned may be divided using any of the definitions mentioned.

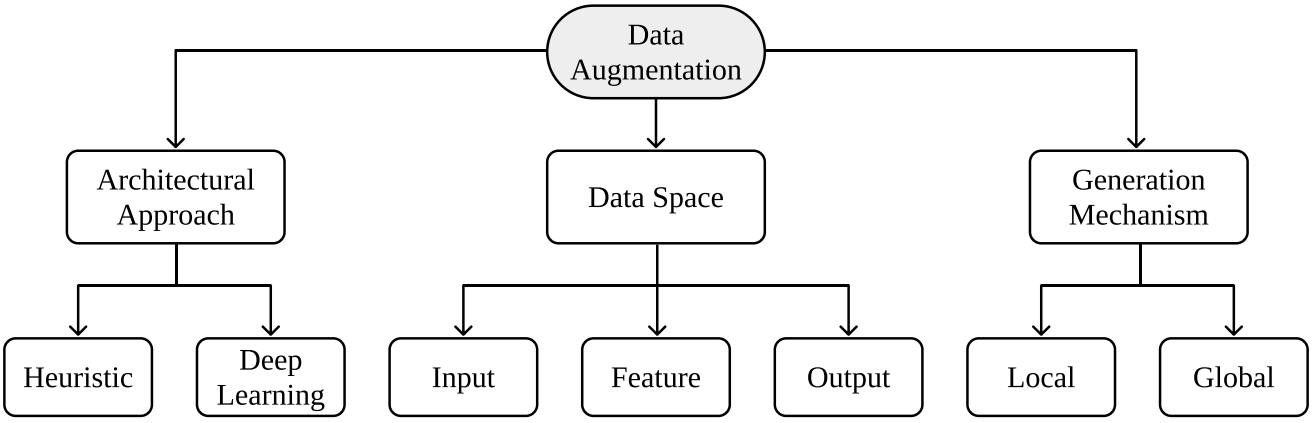


Figure 2.2.: Data Augmentation concept map.

Heuristic methods use the information found in the input space to generate new, relevant, non-duplicated observations by applying a predefined set of rules, while incorporating a degree of randomness in the generation process. Since data augmentation occurs in the input space, these are cost-effective approaches to data augmentation. For this reason, heuristic methods are simpler to implement and are particularly appealing for low dimensional classification problems, especially when the computational power available is limited.

Some Deep Learning methods, on the other hand, attempt to map the original input space into a lower-dimensional representation, known as feature space [17]. The generation of artificial observations occurs at the feature space level, before being reconstructed to the original input space. This is commonly done with Convolutional Neural Networks (CNN) and auto-encoder architectures [18]. Since data augmentation is performed in the feature space, this type of approach is particularly useful for high-dimensional data types and results in more plausible synthetic observations [17]. However, deep learning approaches require more computational power than heuristic approaches and the resulting feature space is difficult to interpret.

The difference in classification performance of the two perspectives is still unclear. Wen et al. [73] evaluate the impact of data augmentation on time series for various classification and forecasting tasks. Although they found that both heuristic and deep learning approaches improved the results over the various experiments, there was no direct comparison among the different methods. Wong et al. [20] compared both input and feature space data augmentation methods for image data classification performance over the MNIST dataset. They found that input space augmentation can lead to better classification performance if plausible transforms on the data are known. However, in [17] the authors discuss that the effectiveness of each data augmentation method generally depends on the domain. The lack of research on effective, domain-agnostic data augmentation methods appears to be a current research gap.

2.2.1. Heuristic Approaches

Various heuristic approaches depend on the data type. For example, image data augmentation may be done via translation, cropping or random erasing [16], among others [18]. However, these techniques depend on the context and may not be applicable to other data types such as time-series [19, 73] or tabular data. In this subsection we will focus on domain-agnostic data augmentation methods.

Heuristic approaches may be applied at the input or feature space. The appropriate method to be applied also depends on the machine learning goal. Specifically, heuristic methods are commonly used to address classification problems where the frequency of the different target classes vary significantly, a problem known as Imbalanced Learning [74]. In this context, the dataset contains one or multiple rare classes, which become more difficult to predict. This happens because during the learning phase, classifiers are trained in order to maximize one or few performance metrics. Although, a poor choice of performance metrics (*i.e.*, a metric insensitive to class imbalance, such as overall accuracy) might lead to a poor estimation of the model's actual performance, since minority classes contribute less to the learning phase and to the estimation of the performance metric [30].

The problem of Imbalanced Learning is frequently addressed with oversampling algorithms [75]. Oversampling methods generate artificial observations in order to balance class distributions using contextual information based on the original dataset. These methods apply linear or geometric interpolations between a random observation and one of its neighbors to generate a new observation.

SMOTE [22] is one of the most popular oversampling methods. It generates an artificial observation along a line segment between a randomly selected minority class observation and one of its nearest-neighbors. Since it was first proposed, modifications at the neighbors parent observations selection and data generation mechanisms of the original algorithm were proposed. Borderline-SMOTE [24] is an example of a modification of SMOTE's data selection mechanism. Instead of selecting any random minority class observation and one of its neighbors, the algorithm focuses in the minority class observations closest to the decision boundary. Geometric-SMOTE [23] proposes a modification of the data generation mechanism. Instead of generating data within a line segment, it generates data within a hyper-spheroid between two parent observations.

Contrary to Deep Learning approaches, heuristic approaches can be applied at the input space without the need of learning a feature space. This allows the implementation of heuristic data augmentation with less computational power and technical complexity. In addition, in contexts of limited data availability

(*i.e.*, small datasets), deep learning approaches are not appropriate since the amount of parameters to be tuned during the learning phase often exceeds the number of observations in the dataset, making it over-parametrized. The augmentation of small datasets using heuristic approaches is explored in an Active Learning context in [76]. The authors found that significantly smaller amounts of curated data using Active Learning, along with heuristic data augmentation methods, achieved a classification performance comparable to classifiers trained over the full dataset.

2.2.2. Deep Learning Approaches

Different deep learning approaches and architectures have been developed for various domains. Deep learning data augmentation methods may be developed via augmentation at the feature space (which involves learning a feature space) [17] or via a combination of a set number of observations into a neural network in order to output a non-linear, non-geometric combination of input observations [77]. Other domain-specific deep learning methods also exist, such as style transferring techniques (specific to image data) [77, 78].

Data augmentation at the feature space is especially useful when dealing with high-dimensional datasets with complex and/or discontinuous distributions. For example, many heuristic data augmentation techniques cannot be applied in handwritten digits classification problems since they may change the true label of the generated image and generate noisy data. In this situation, performing data augmentation at the input space may introduce noise [79], since the data is also subjected to the curse of dimensionality (see Figure 2.3a). This allows transformations of known observations in a lower dimensional space to generate new, non-noisy artificial observations projected in the input space, as shown in Figures 2.3b and 2.3c.

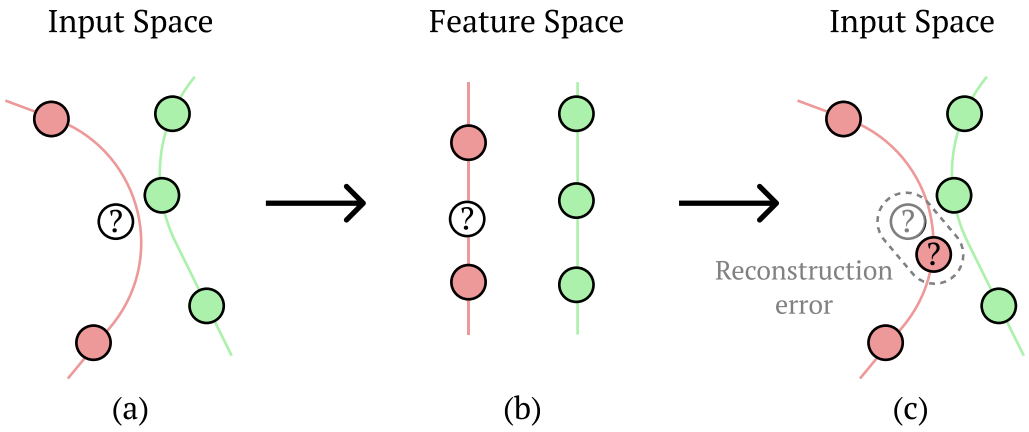


Figure 2.3.: Example of a sparse input space and its corresponding feature space. Learning a manifold feature space facilitates the generation of non-noisy artificial data. In the original input space (a) the unseen/artificial observation marked with "?" is closer to a green observation and to the learnt red manifold space, which may lead to a noisy observation. When projected or generated in the feature space (b), the observation will better match the nearest manifold measure and avoid noisy data in the input space (c). Adapted from [80].

The utilization of autoencoders [81] is particularly useful to perform feature space data augmentation [18]. Autoencoders are composed by an encoder and a decoder, which map the input space to and from the feature space, respectively. The autoencoder is trained by minimizing the difference between the original observation and the reconstructed observation (see Figure 2.3c). Once the training phase is completed, heuristic methods are applied in the feature space [17].

Generative Adversarial Network (GAN) [82] architectures is a deep learning approach frequently used as a data augmentation method. It involves a generator and a discriminator. Although many different architectures have been proposed, the generator in the vanilla GAN algorithm may be seen as a decoder, which is trained based on the gradients calculated via the discriminator. The discriminator attempts to distinguish true and generated (fake) observations in order to assess the quality of the data produced by the generator. The problem is better formulated as a minimax decision rule, where the generator attempts to fool the discriminator by producing observations that are difficult to classify as generated.

When the size of the training dataset is not sufficiently large to employ deep learning approaches and other related datasets or unlabeled datasets are available, one technique that may also be used is transfer learning. In this context, the data augmentation model is trained on a secondary model and is later adjusted to the training dataset. This allows the usage of deep learning models in few-shot learning environments [80].

2.3. Methodology

In this section we describe the procedures defined for the literature collection, data preprocessing and literature analysis. The analysis of the literature was developed with 3 different approaches. Throughout the analyses, data preprocessing and hyperparameter tuning was developed iteratively. The procedure adopted in this manuscript is shown in Figure 2.4.

The literature collection procedure is described in Subsection 2.3.1. The data and text preprocessing is described in Subsection 2.3.2. The exploratory data analysis described in Subsection 2.3.3 was done to understand which manuscripts, journals and conferences are most significant within the field of Data Augmentation. The manuscripts' keywords were used to construct a network of keywords (described in Subsection 2.3.4) and study the different communities of keywords found in the network. The topic modelling and parameter tuning is described in Subsection 2.3.5.

2.3.1. Literature Collection

The focus of this literature analysis is to understand the different algorithms, domains and/or tasks that employ data augmentation techniques. Therefore, we search for documents containing the keyword "data augmentation" in the search query. The results were then limited to conference papers and journal articles written in English that were published in the past 15 years. Due to the large amount of results found, using solely the Scopus database was found to be sufficient. One of the goals during the search query design was to come up with a simple and unbiased query. The resulting query is shown below:

```
KEY ("data augmentation") AND (LIMIT-TO (LANGUAGE, "English"))
AND (LIMIT-TO (DOCTYPE, "cp") OR LIMIT-TO (DOCTYPE, "ar"))
AND (LIMIT-TO (PUBYEAR, 2021) OR LIMIT-TO (PUBYEAR, 2020)
 OR  LIMIT-TO (PUBYEAR, 2019) OR LIMIT-TO (PUBYEAR, 2018)
 OR  LIMIT-TO (PUBYEAR, 2017) OR LIMIT-TO (PUBYEAR, 2016)
 OR  LIMIT-TO (PUBYEAR, 2015) OR LIMIT-TO (PUBYEAR, 2014)
 OR  LIMIT-TO (PUBYEAR, 2013) OR LIMIT-TO (PUBYEAR, 2012)
 OR  LIMIT-TO (PUBYEAR, 2011) OR LIMIT-TO (PUBYEAR, 2010)
 OR  LIMIT-TO (PUBYEAR, 2009) OR LIMIT-TO (PUBYEAR, 2008)
 OR  LIMIT-TO (PUBYEAR, 2007) OR LIMIT-TO (PUBYEAR, 2006))
```

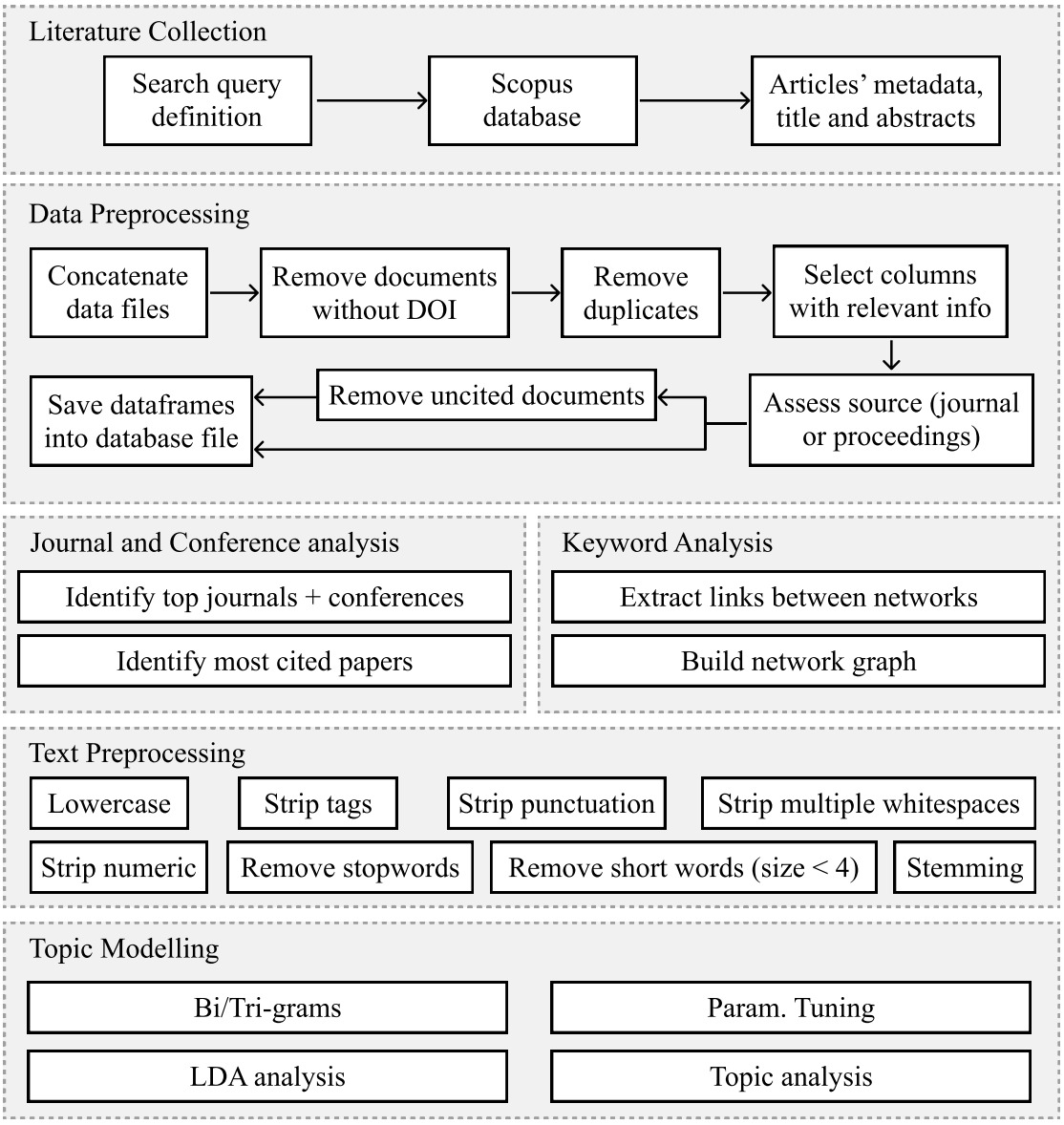


Figure 2.4.: Diagram of the proposed literature analysis approach.

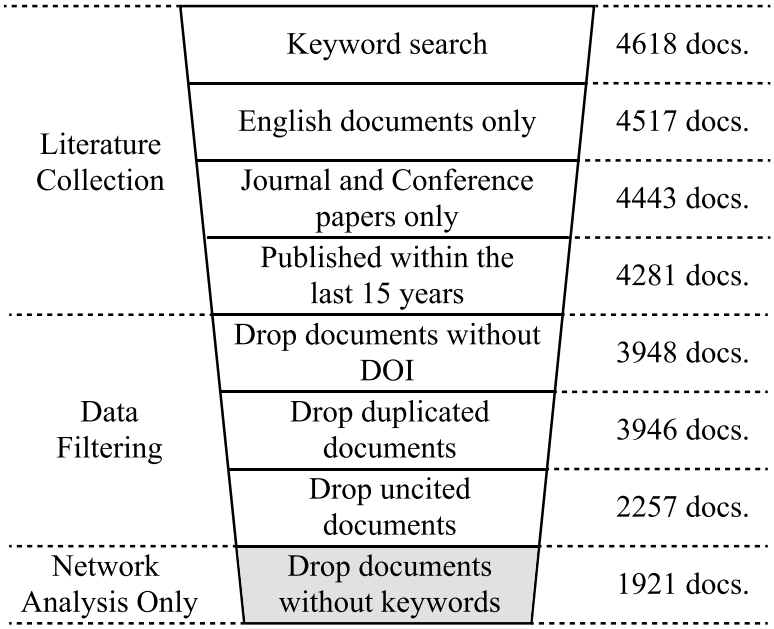


Figure 2.5.: Data filtering pipeline.

The search query resulted in 4281 documents. The resulting data selection/filtering pipeline is shown in Figure 2.5. Due to the limitations in the Scopus data export (maximum 2000 documents per export), the data was split in four different time periods and exported separately: 2006 until 2018, 2019, 2020 and 2021, which produced four CSV files.

2.3.2. Data Preprocessing

The data preprocessing stage and amount of documents dropped is represented in Figure 2.5. The data was first concatenated into a single data frame. During this process, we found that one of the exported references had a corrupted line, which caused the loss of one additional document. Since the DOI can be used as a unique identifier for intellectual property [83], references without a DOI were disregarded from further analysis, while the ones with the same identifiers are removed (*i.e.*, only one of the repeating entries is kept).

This dataset was kept to perform the analysis described in Subsection 2.3.3. However, further preprocessing was done for the remaining parts of the literature analysis. References without any citations were excluded for the keyword network and topic modelling analyses. Finally, only the documents containing keywords in Scopus' database were used to prepare the network analysis.

2.3.3. Journal and Conference analysis

The exploratory analysis developed on the preprocessed dataset was targeted towards the identification of the most significant works, journals and conferences. We used the citation count as a proxy to understand the impact of a specific manuscript within the research community.

The identification of the most significant conferences and journals is done by sorting each type of publication according to the number of citations per document. Conferences and journals with less than 10 papers published in the area are not considered in this analysis.

2.3.4. Keyword Analysis

The analysis of keywords is expected to uncover general trends in data augmentation research and its applications. The keyword "data augmentation" was removed since it would link with all other keywords. Keywords are connected based on their co-occurrence in each research paper to form the edges of the network. It consists of an undirected graph whose weights are based on the total citation count for the papers containing a given keyword pair and is calculated as weight $= \log(\text{citations}) + 1$ to avoid a potential bias caused by highly cited research articles. The size of the nodes were determined with a logarithmic transformation of each node's page rank.

Keyword combinations showing up in only one document are removed from further analysis. The keyword network is then analysed using Python and the communities were found using the greedy modularity maximization algorithm proposed in [84]. The results of the analysis and community detection were ported to Gephi to produce the final visualizations.

2.3.5. Topic Modelling

The extraction of topics was done using the publication's abstracts. The words were tokenized and all tags, special characters, punctuation, multiple white spaces, numeric values, stop words and words with size smaller than 4 were removed. Finally, we enriched the corpus by constructing bi-grams and tri-grams.

We used a Latent Dirichlet Allocation (LDA) model [85] to infer the topics present in our research domain. The tuning of the parameters was done through experimentation and qualitative interpretation of the results achieved. Additionally, the coherence score curve was also used as a reference for parameter tuning and the choice of parameters, which are described in Table 2.1.

Model	Hyperparameter	Value
LDA	Num Topics	8
	Chunk Size	2000
	Passes	20
	Alpha	0.1
	ETA	auto

Table 2.1.: Hyperparameters used.

2.3.6. Software Implementation

The analysis and modelling was developed using the Python programming language, along with the Scikit-Learn [86], Gensim [87], and Networkx [88] libraries. The final network analysis and visualization

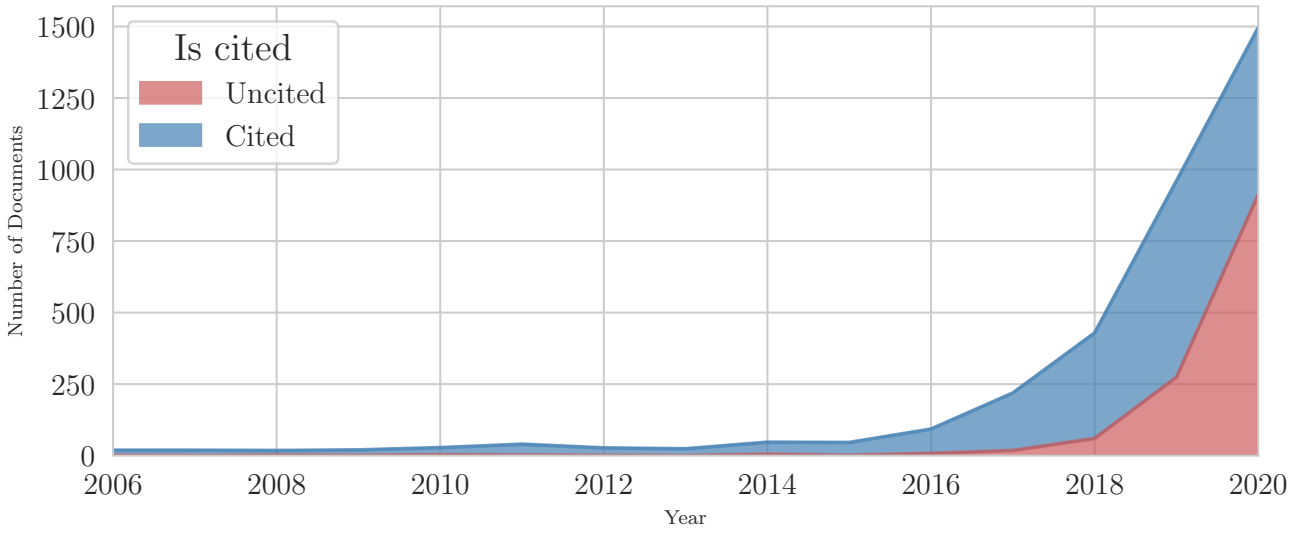


Figure 2.6.: Annual number of publications containing the keyword "data augmentation".

was done with Gephi [89]. All functions, algorithms, analyses and results are provided in the GitHub repository of the project.

2.4. Results & Discussion

The popularity of research in data generation has grown significantly in the past 5 years, as shown in Figure 2.6. Despite the significant amount of uncited publications, out of the ones published in 2020, 39% have already been used in other works. Although most of the research developed before 2016 was used in other works, the amount of cited research increased significantly after that period.

2.4.1. Journal and Conference Analysis

The initial exploration of the bibliometric data allows us to assess which journals focused in data augmentation more intensely over the past years, as shown in Table 2.2. Most of the top journals belong to technical fields, predominantly from Statistics, Remote Sensing, Medical Imaging and other domains of applications such as agriculture. In addition, all these journals have a high impact in their respective fields (based on Scimago Journal & Country Rankings).

Citation-wise, the publications coming from conference proceedings tend to have a comparable impact in the research community, as shown in Table 2.3. The most relevant conferences are positioned in the computer science and information management fields. Research developed in other areas of application, such as computer vision, speech recognition, acoustic modelling, natural language processing and signal processing have more activity in the form of conference proceedings publications. Conversely, the domains most frequent in journal publications are not as active on conference proceedings publications.

The papers with the highest citation count are listed in Table 2.4. We found that much of the research focused on improving deep learning classification, segmentation or object detection without a focus on a particular domain of application. Other papers centered in the application of data augmentation methods for biomedical image classification and segmentation, sound and speech recognition and remote sensing.

Source title	Publications	Citations	Average
Journal of the American Statistical Association	11	538	48.91
IEEE Geoscience and Remote Sensing Letters	19	552	29.05
Neurocomputing	35	808	23.09
Expert Systems with Applications	14	283	20.21
Medical Image Analysis	15	288	19.20
Neural Networks	10	190	19.00
Journal of Computational and Graphical Statistics	23	433	18.83
Computers and Electronics in Agriculture	15	219	14.60
Biometrics	13	163	12.54
IEEE Transactions on Medical Imaging	10	123	12.30

Table 2.2.: Top journals focusing on data augmentation techniques, sorted by citations per document.

2.4.2. Keyword Analysis

The keyword network shown in Figure 2.7 revealed 8 main communities of keywords, and 13 other small communities. The different communities are distinguished by the type of algorithms used and/or the domain of application. The main distinctive factor for the larger communities are the types of generative models used, while the smaller communities are distinguished according to the domain of application. The most significant findings we found from this analysis are:

1. The community marked with pink-colored nodes is characterized by the usage of neural network-based data augmentation methods in convolutional neural networks. The keyword "deep learning" is positioned as a central node (although not labelled in the figure to maintain readability). Other relevant keywords are related to machine/deep learning frameworks, deep learning classifiers and data augmentation algorithms, such as "tensorflow", "keras", "convolutional neural network" and "generative adversarial networks". Domain specific keywords are also present:
    - Medical keywords located in this community cover a variety of applications. Relevant sub communities are ["hand writing", "parkinson's disease (pd)", "transfer learning"], ["breast cancer", "computer-aided detection"], ["melanoma", "skin cancer", "image processing", "googlenet"], ["chest x-ray", "computer-aided diagnosis", "tuberculosis", "segmentation"] and ["brain", "mri", "multiple sclerosis"].
    - Remote sensing keywords are typically related to classification and object detection tasks. Relevant sub communities are ["object detection", "aerial image", "drone", "generative adversarial network", "semantic segmentation"], ["attributed scattering center (asc)", "synthetic aperture radar (sar)", "convolutional neural network (cnn)"], ["remote sensing", "road extraction", "transfer learning", "generative adversarial network"]. Keywords such as "hyperspectral imaging" and "weather classification" are also scattered around the community.
    - Facial recognition research is also represented in few sub communities: ["micro expression recognition", "small training data", "convolutional neural network (cnn)", "local binary pattern-three orthogonal planes (lbp-top)"] and ["training data augmentation", "sequence-to-sequence speech synthesis", "sequence-to-sequence speech recognition"].
    - Fault detection studies also used data augmentation to deal with imbalanced datasets: ["fault diagnosis", "imbalanced data", "gan"]

Source title	# Pubs.	Cited	Avg
Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition	49	2111	43.08
Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	372	14946	40.18
Procedia Computer Science	13	288	22.15
International Conference on Information and Knowledge Management, Proceedings	10	180	18.00
IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops	23	314	13.65
ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings	95	1153	12.14
Proceedings - International Symposium on Biomedical Imaging	30	346	11.53
Proceedings of the International Conference on Document Analysis and Recognition, ICDAR	17	158	9.29
Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR	13	113	8.69
2019 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019 - Proceedings	12	84	7.00

Table 2.3.: Top conferences focusing on data augmentation techniques, sorted by citations per document.

- Data augmentation was also associated to regularization methods and feature extraction tasks, based on the presence of the sub communities [“overfitting”, “dropout” and “cnn”] and [“feature extraction”, “cnn”, “svm”].

2. The community marked with blue-colored nodes is characterized by the usage of Markov Chain-based algorithms. The keywords “markov chain”, “data augmentation algorithm” and “monte carlo” appear as central nodes. No application-specific sub-community was found.

3. The community marked with green-colored nodes is characterized by the usage of Markov Chain and Bayesian-based algorithms. The keywords “bayesian inference”, “markov chain monte carlo”, “mcmc”, “bayesian analysis”, “missing data” and “em algorithm” (expectation maximization algorithm). Application-specific keywords may be found sparsely distributed across the community, all of them related to biological applications. Specifically, the sub community [“ecological health”, “stressor-response”, “biological monitoring”, “bayesian methods”] and the keyword “camera trapping” were found in this community.

4. The community marked with orange-colored nodes is characterized by keywords specific to big data and data warehousing applications. The network is composed of the keywords “big data”, “data lake”, “olap”, “map reduce”, “cmm”, “data warehouse”, “augmentation” and “dm”.

5. The remaining communities consist mostly of data augmentation methods applied to specific domains. Specifically, the usage of temporal-dynamic neural network architectures with “eeg (electroencephalogram)”, music information retrieval applications (e.g., “chord recognition”), speech/

Authors	Title	Year	Cited
Ronneberger O., Fischer P., Brox T.	U-net: Convolutional networks for biomedical image segmentation	2015	13597
Chatfield K., Simonyan K., Vedaldi A., Zisserman A.	Return of the devil in the details: Delving deep into convolutional nets	2014	1885
Snyder D., Garcia-Romero D., Sell G., Povey D., Khudanpur S.	X-Vectors: Robust DNN Embeddings for Speaker Recognition	2018	636
Shorten C., Khoshgoftaar T.M.	A survey on Image Data Augmentation for Deep Learning	2019	590
Salamon J., Bello J.P.	Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification	2017	505
Eitel A., Springenberg J.T., Spinello L., Riedmiller M., Burgard W.	Multimodal deep learning for robust RGB-D object recognition	2015	352
Ding J., Chen B., Liu H., Huang M.	Convolutional Neural Network with Data Augmentation for SAR Target Recognition	2016	319
Wong S.C., Gatt A., Stamatescu V., McDonnell M.D.	Understanding Data Augmentation for Classification: When to Warp?	2016	302
Frid-Adar M., Diamant I., Klang E., Amitai M., Goldberger J., Greenspan H.	GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification	2018	296
Bilen H., Vedaldi A.	Weakly Supervised Deep Detection Networks	2016	287

Table 2.4.: Top papers using data augmentation techniques, sorted by citation count.

speaker recognition and embedding, time series forecasting of diabetes and natural language processing and text classification.

2.4.3. Topic Analysis

The LDA topic extraction resulted in 8 different topics, whose distribution of topics is shown in Figure 2.8. The main topics within which most articles were included is topic 5, which is defined by the main theoretical keywords related to image data augmentation. Rather, the secondary topic is more useful for this analysis. It is found based on the topic likelihood of each document, excluding the dominant topic. Documents belonging to the same group across primary, secondary and/or tertiary topics had a likelihood of zero of belonging to any other topic.

The topics found in the bibliometric data are shown in Table 2.5. A few topics seem to overlap each other, although they are generally distinguishable. The primary domains of application of data augmentation methods differ for each different topic identified:

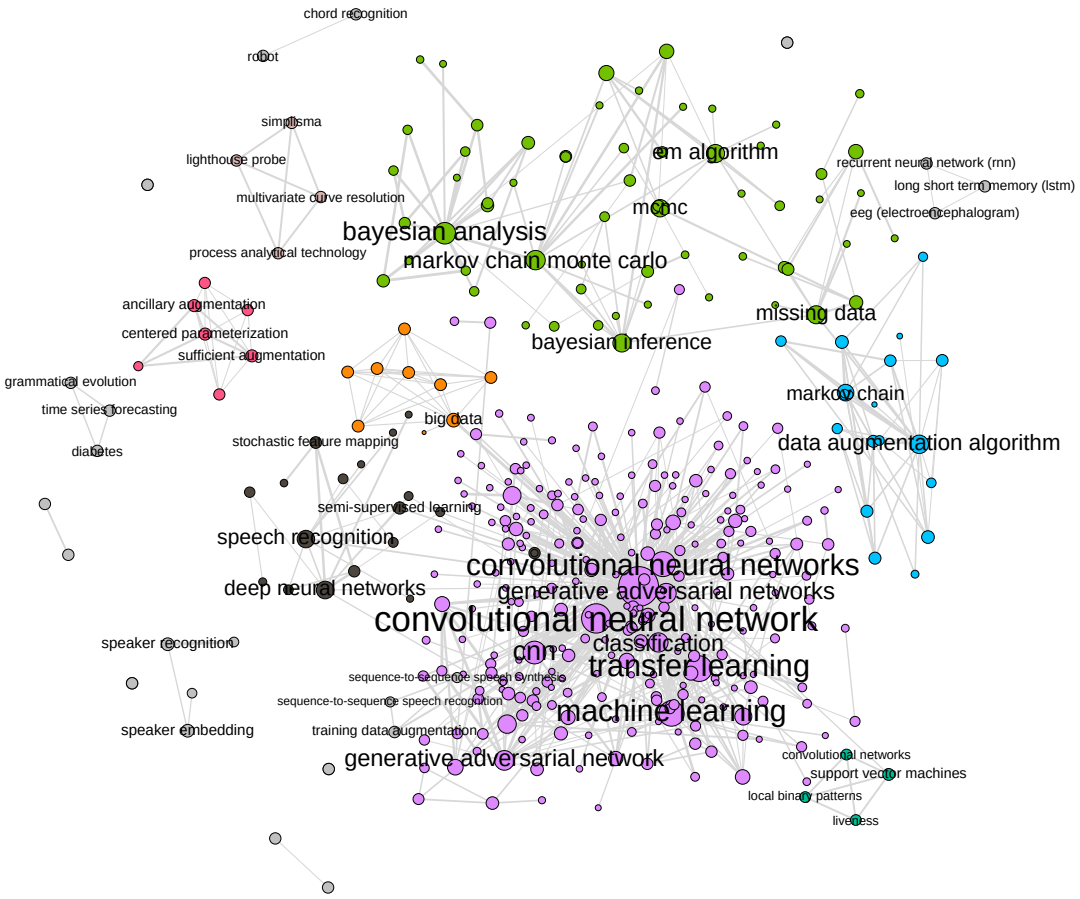


Figure 2.7.: Keyword network.

1. Documents in Topic 1 frequently use the word “yolov”, which refers to the YOLOvX family of deep learning object detection models [90], where X refers to the version of the model used (the most recent version is 5). Another relevant keyword is “style_transfer”, which refers to a specific technique of data augmentation.

   This topic has two primary domains of application. The keywords “pest” and “coffe” refer to data augmentation on agriculture research. The keywords “biomed”, “histolog” and “nodul” refer to biomedical applications such as pulmonary nodule detection and histology image classification. Within these topics, a few domain-specific data augmentation algorithms were proposed. For example, in [91] the authors propose a style-transfer data augmentation method for histology image classification.

2. Documents in Topic 2 are primarily associated to the study of applications that include image data augmentation. The dominant keyword, “hyperspectr_imag”, refers to the application of data augmentation on hyperspectral images, commonly used in remote sensing and medicine. Other classification tasks include license plate detection (“licens_plate”), inpainting (“inpaint”), background subtraction (“illumin_chang”) and cloud shadow detection/segmentation (“shadow”).

3. Documents in topic 3 refer to the application of data augmentation to deal with censored data (a condition in which the value of an observation is only partially known) and/or supervised tasks on data structured as graphs. Other domains of application involve chest x-rays classification (“cxr”), epidemiology (“risk_factor”) and few audio/music information retrieval (“sourc_separ”) articles.

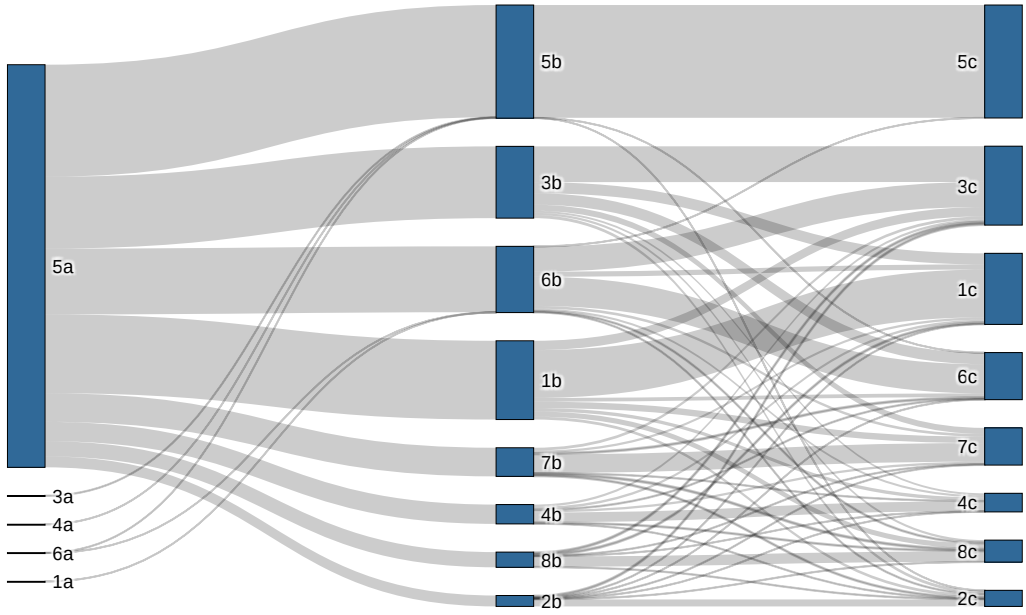


Figure 2.8.: Distribution of documents over the different topics found. The left column represent the primary topics, the middle column represents the secondary topics and the right columns represents the tertiary topics.

4. Documents in topic 4 refer to the application of data augmentation methods on object detection tasks. Specifically fire and smoke, pedestrians and crowd counting. Other applications within this topic are focused on speech recognition and angiography segmentation/classification.
5. Documents in topic 5 are focused on image segmentation and classification methods where data augmentation algorithms are involved. It includes common keywords present in a large set of articles. These articles are mainly focused on the development of different convolutional neural network architectures ("cnn") and neural network-based data augmentation methods.
6. Documents in topic 6 are focused on Bayesian-based algorithms and Markov Chain algorithms. This topic includes data augmentation on regression tasks and misclassification detection.
7. Documents in topic 7 covers the application of data augmentation into various domains. Specifically, music information retrieval ("music"), fish/marine organisms recognition, gender bias, speech recognition, random erasing
8. Documents in topic 8 contains remote sensing and biomedicine as the primary research domains. The keywords "drone" and "aircraft" refer to the sources of data collected for remote sensing work, whereas "pneumonia" and "chest_rai_imag" refers to biomedicine research topics/image data.

The per-year popularity of the different topics is shown in Figure 2.9. Since 2015, topic 5 gained more research momentum, whereas topic 6 lost much of its relative popularity within the field. In the past

Topic	Representative Paper	Papers	Words
1	GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification	440	yolov, pest, style_transfer, coffe, thermal, biomed, scene_text, histolog, nodul, visibl
2	CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training	61	hyperspectr_imag, licens_plate, command, inpaint, illumin_chang, upper, restor, ann, foreign, shadow
3	A survey on Image Data Augmentation for Deep Learning	401	censor, markov_chain, node, team, tree, cxr, risk_factor, mass, largest, sourc_separ
4	Return of the devil in the details: Delving deep into convolutional nets	108	smoke, pedestrian, transcrib, crowd, children_speech, intent, adult, auxiliari_variabl, speech, angiographi
5	U-net: Convolutional networks for biomedical image segmentation	632	imag, detect, gener, dataset, classif, sampl, network, cnn, featur, augment
6	Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification	370	tea, multivari, markov_chain_mont_carlo, bayesian, regress, misclassif, procedur, famili, illustr, mcmc
7	Weakly Supervised Deep Detection Networks	160	music, fish, marin, gender, vocal, random_eras, low_qualiti, crowd, prune, bengali
8	An Efficient Deep Learning Approach to Pneumonia Classification in Healthcare	85	drone, gait, aircraft, gestur_recognit, pneumonia, chest_rai_imag, covid, walk, onset, hidden_layer

Table 2.5.: Description of the main topics found in the literature.

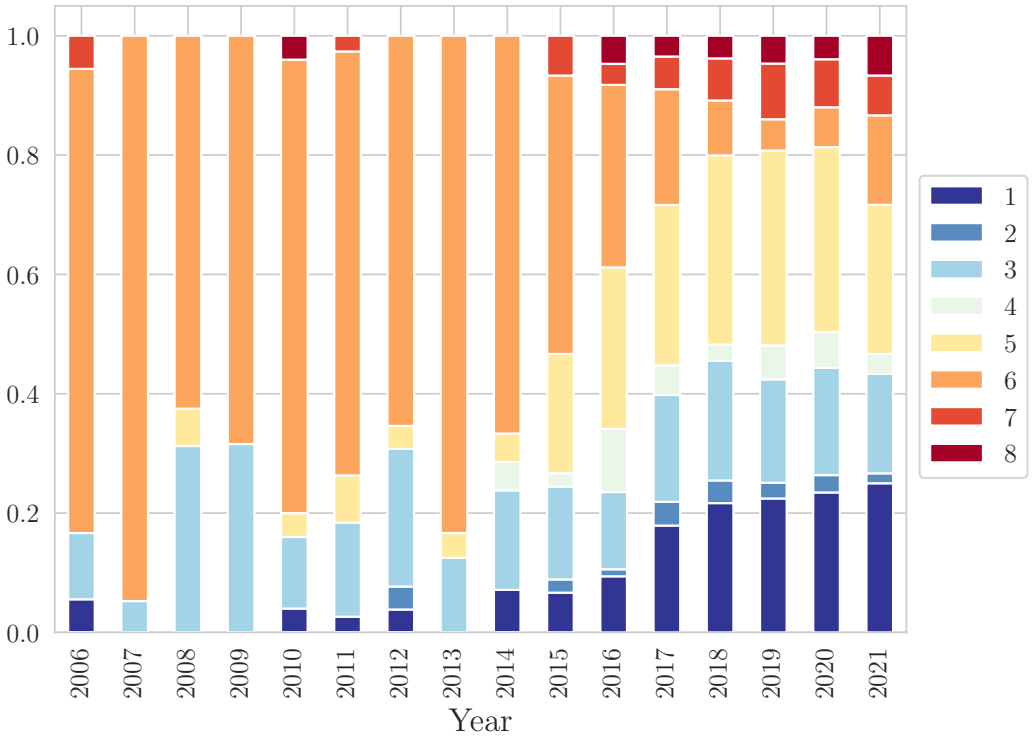


Figure 2.9.: Topic frequency per year.

5 years topics 8 and 3 have become steady research streams while topic 1 saw a significant growth in popularity.

2.4.4. Research Gap Discussion

Data augmentation mechanisms are often used as regularization methods for deep learning classifiers. The study of data augmentation mechanisms in ensembles of simple classifiers have achieved state-of-the-art performance not only 10 years ago [68, 69], but also when compared to modern deep learning architectures [70, 71, 92]. However, the implementation of different data augmentation methods shows a promising path to improve the performance of simple classifiers (and/or recent ensemble architectures) and requires further research.

A research application that was not frequently found in the literature was small dataset augmentation. This is particularly useful for any complex problem when the amount of labeled data available to use as training data is scarce, which limits the usage of classification algorithms and especially deep learning algorithms. In this context, techniques such as Active Learning can be used to annotate a small amount of data, while maximizing the classification performance [33]. However, classifiers may not be capable of generalizing with small training datasets and the ability to reproduce and augment the labelled data available can further reduce annotation cost and allow the usage of data intensive classifiers.

Another limitation found in the literature relates to the problem of initialization on network-based data augmentation methods. The same data augmentation algorithm trained with different initialization settings (different random seed or training subset) may lead to different model parameters and quality of the trained classifier.

The rapid development of data augmentation algorithms raises additional open questions on how the data used and store for model training. Specifically, the lower data storage and processing power available

to the general public (*i.e.*, organizations and individuals) is a limitation for producing state-of-the-art classifiers. Another problem arises from data privacy concerns, since the usage of user data to train machine learning models typically involve the storage of such information. However, if data augmentation algorithms were continuously updated and capable of producing reliable data on an as-needed basis, not only would storage requirements decrease, but it would also become possible to work with fully artificial data, without the need to store as much data. This would also facilitate the sharing of datasets (in the form of an algorithm) without compromising sensitive data.

2.5. Conclusion

Depending on the domain of application, data augmentation research differs in the format of publication. On the one hand, domains like Statistics, Remote Sensing and Medical Imaging seem more active on journal publications, typically in journals with high impact factor. On the other hand, research developed in the domains of computer vision, speech recognition, acoustic modelling, natural language processing and signal processing seem to attribute higher importance to conference papers. Many of the influential papers we found were focused on deep learning methods for classification, segmentation, sound and speech recognition and remote sensing.

We analysed the different communities of keywords formed using document keywords, as well as topic analysis using a LDA analysis over the document's abstracts. We found various distinctive areas of research, both regarding the data augmentation methods used and the domain of application. We found that in recent years research on augmentation methods using Bayesian-based algorithms, as well as Markov Chain algorithms reduced its popularity, whereas data augmentation methods based on neural networks and deep learning classifiers have increased its popularity.

Data augmentation is most commonly applied/studied in the realm of computer vision for tasks like image classification, segmentation, object detection, inpainting and background subtraction tasks, even though it may be applied to many other data structures. It is frequently used in studies within the domains of biomedicine, agriculture, speech recognition, acoustic modelling, remote sensing and computational creativity. It is also used alongside other data preprocessing techniques, such as feature extraction and dimensionality reduction.

Although data augmentation is a vibrant area of research, there are still significant gaps to be addressed. Data augmentation methods are increasingly used as regularization methods for deep learning. Although, recent research shows that the same can be done for simpler classifier configurations in order to achieve a classification performance comparable to that of state-of-the-art deep learning, which require further confirmation, as well as the development of less computational intensive data augmentation methods. Other less popular topics, such as small data augmentation, appear to have a relevant practical importance and require further research. In addition, other limitations of data augmentation algorithms should be addressed. One problem commonly found in the literature is the impact the weights initialization and training set used have in the quality of the trained algorithm. In the future, using data augmentation methods as a source of artificial datasets can address a variety of concerns, such as data privacy, sharing and storage. Finally, exploring data augmentation algorithms to complement or replace techniques such as Active Learning may reduce the cost of data collection, although it is yet to be explored.

3. Improving Imbalanced Land Cover Classification with K-means SMOTE: Detecting and Oversampling Distinctive Minority Spectral Signatures

Published as Joao Fonseca, Georgios Douzas, Fernando Bacao, in Information Journal, 2021

Land cover maps are a critical tool to support informed policy development, planning, and resource management decisions. With significant upsides, the automatic production of Land Use/Land Cover maps has been a topic of interest for the remote sensing community for several years, but it is still fraught with technical challenges. One such challenge is the imbalanced nature of most remotely sensed data. The asymmetric class distribution impacts negatively the performance of classifiers and adds a new source of error to the production of these maps. In this paper, we address the imbalanced learning problem, by using K-means and the Synthetic Minority Oversampling TEchnique (SMOTE) as an improved oversampling algorithm. K-Means SMOTE improves the quality of newly created artificial data by addressing both the between-class imbalance, as traditional oversamplers do, but also the within-class imbalance, avoiding the generation of noisy data while effectively overcoming data imbalance. The performance of K-means SMOTE is compared to three popular oversampling methods (Random Oversampling, SMOTE and Borderline-SMOTE) using seven remote sensing benchmark datasets, three classifiers (Logistic Regression, K-Nearest Neighbors and Random Forest Classifier) and three evaluation metrics using a 5-fold cross-validation approach with 3 different initialization seeds. The statistical analysis of the results show that the proposed method consistently outperforms the remaining oversamplers producing higher quality land cover classifications. These results suggest that LULC data can benefit significantly from the use of more sophisticated oversamplers as spectral signatures for the same class can vary according to geographical distribution.

Keywords: LULC Classification; Imbalanced Learning; Oversampling; Data Augmentation; Clustering

3.1. Introduction

The increasing amount of remote sensing missions granted the access to dense time series (TS) data at a global level and provides up-to-date, accurate land cover information [93]. This information is often materialized through Land Use and Land Cover (LULC) maps. While Land Cover maps define the biophysical cover found on the surface of the earth, Land Use maps define how it is used by humans [94]. Both Land Use and Land Cover maps constitute an essential asset for various purposes, such as land cover change detection, urban planning, environmental monitoring and natural hazard assessment [1]. However, the timely production of accurate and updated LULC maps is still a challenge within the remote

sensing community [95]. LULC maps are produced based on two main approaches: photo-interpreted by the human eye, or automatic mapping using remotely sensed data and classification algorithms.

While photo-interpreted LULC maps rely on human operators and can be more reliable, they also present some significant disadvantages. The most important disadvantage is the cost of production, in fact photo-interpretation consumes significant resources, both in terms of money and time. Because of that, they are not frequently updated and not suitable for operational mapping over large areas. Finally, there is also the issue of overlooking rare or small-area classes, due to factors such as the minimum mapping unit being used.

Automatic mapping with classification algorithms based on machine-learning (ML) have been extensively researched and used to speed up and reduce the costs of the production process [1, 75, 96]. Improvements in classification algorithms are sure to have significant impact in the efficiency with which remote sensing imagery is used. Several challenges have been identified in order to improve automatic classification:

1. Improve the ability to handle high-dimensional datasets, in cases such as Multi-spectral TS composites high-dimensionality increases the complexity of the problem and creates a strain on computational power [5].
2. Improve class separability, as the production of an accurate LULC map can be hindered by the existence of classes with similar spectral signatures, making these classes difficult to distinguish [6].
3. Resilience to mislabelled LULC patches, as the use of photo-interpreted training data poses a threat to the quality of any LULC map produced with this strategy, since factors such as the minimum mapping unit tend to cause the overlooking of small-area LULC patches and generates noisy training data that may reduce the prediction accuracy of a classifier [4].
4. Dealing with rare land cover classes, due to the varying levels of area coverage for each class. In this case using a purely random sampling strategy will amount to a dataset with a roughly proportional class distribution as the one on the multi/hyperspectral image. On the other hand, the acquisition of training datasets containing balanced class frequencies is often unfeasible. This causes an asymmetry in class distribution, where some classes are frequent in the training dataset, while others have little expression [97, 98].

The latter challenge is known, in machine learning, as the imbalanced learning problem [74]. It is defined as a skewed distribution of instances found in a dataset among classes in both binary and multi-class problems [99]. This asymmetry in class distribution negatively impacts the performance of classifiers, especially in multi-class problems. The problem comes from the fact that during the learning phase, classifiers are optimized to maximize an objective function, with overall accuracy being the most common one [100]. This means that instances belonging to minority classes contribute less to the optimization process, translating into a bias towards majority classes. As an example, a trivial classifier can achieve 99% overall accuracy on a binary dataset where 1% of the instances belong to the minority class if it classifies all instances as belonging to the majority class. This is an especially significant issue in the automatic classification of LULC maps, as the distribution of the different land-use classes tends to be highly imbalanced. Therefore, improvements in the ability to deal with imbalanced datasets will translate into important progress in the automatic classification of LULC maps.

There are three different types of approaches to deal with the class imbalance problem [9, 75]:

1. Cost-sensitive solutions. Introduces a cost matrix to the learning phase with misclassification costs attributed to each class. Minority classes will have a higher cost than majority classes, forcing the algorithm to be more flexible and adapt better to predict minority classes.
2. Algorithmic level solutions. Specific classifiers are modified to reinforce the learning on minority classes. Consists on the creation or adaptation of classifiers.

3. Resampling solutions. Rebalances the dataset's class distribution by removing majority class instances and/or generating artificial minority instances. This can be seen as an external approach, where the intervention occurs before the learning phase, benefitting from versatility and independency from the classifier used.

Since resampling strategies represent a set of methods that are detached from classifiers by operating at the data level, they allow the use of any off the shelf algorithm, without the need for any type of changes or adaptions to the algorithm. Specifically, in the case of oversampling (defined below), the user is able to balance the dataset's class distribution by without the loss of information, which is not the case with undersampling techniques. This is a significant advantage especially considering that most users in remote sensing are not expert machine learning engineers.

Within resampling approaches there are three subgroups of approaches [9, 75, 101]:

1. Undersampling methods, which rebalance class distribution by removing instances from the majority classes.
2. Oversampling methods, which rebalance datasets by generating new artificial instances belonging to the minority classes.
3. Hybrid methods, which are a combination of both oversampling and undersampling, resulting in the removal of instances in the majority classes and the generation of artificial instances in the minority classes.

Resampling methods can be further distinguished between non-informed and heuristic (i.e., informed) resampling techniques [9, 101, 102]. The former consist of methods that duplicate/remove a random selection of data points to set class distributions to user-specified levels, and are therefore a simpler approach to the problem. The latter consists of more sophisticated approaches that aim to perform over/undersampling based on the points' contextual information within their data space.

The imbalanced learning problem is not new in machine learning but its relevancy has been growing, as attested by [103]. The problem has also been addressed in the context of remote sensing [2]. In this paper, we propose the application of a recent oversampler based on SMOTE [22], the K-means SMOTE [25] oversampler, to address the imbalanced learning problem in a multiclass context for LULC classification using various remote sensing datasets. Specifically, we use seven land use datasets commonly used in research literature, that vary among agricultural and urban land use. The K-means SMOTE algorithm couples two different procedures in the generation of artificial data. The algorithm starts by grouping the instances into clusters by using the K-means algorithm; next, the generation of the artificial data is done using the smote algorithm, taking into consideration the distribution of majority/minority cases in each individual cluster. The idea of starting with a clustering procedure before the data generation phase is important in remote sensing because the spectral signature of the different classes can change significantly based on the geographical area in which it is represented. In other words, the spectral signature of a specific class can vary greatly depending on the geography, meaning that often we will be facing within-class imbalance [104].

In fact, we can decompose class imbalance into two different types: between-class imbalance and within-class imbalance [25, 105]. While the first refers to the overall asymmetry between majority and minority classes, the second results from the fact that in different areas of the input space there might be different levels of imbalance. Depending on the complexity of the input space, different subclusters of minority and majority instances may be present. In order to achieve a balance between minority and majority instances, these subclusters should be treated separately. Assuming that the role of a classifier is to create rules in such a way that it is able to isolate the different relevant sub-concepts that represent both the majority and minority classes, the classifier will create multiple disjunct rules that describe these concepts. If the input space is simple and the classes' instances are grouped together in a unique cluster, the classifier will only need to create (general) rules that comprise large portions of instances belonging to the same class.

To the contrary, if the input space is complex and scatters through multiple small clusters, the classifier will need to learn a more complex set of (specific) rules, which can be seen in Figure 3.1. It is important to note that small clusters can happen both in the minority and majority class, although they will tend to be more frequent in the minority class due to its underrepresentation.

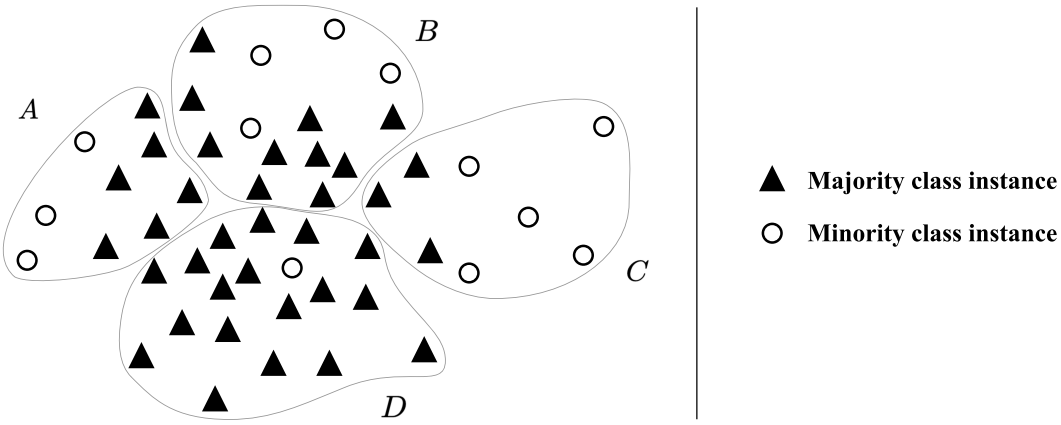


Figure 3.1.: Example of a complex input space. In this example, a classifier would need to separate the minority class' samples across 4 distinguishable clusters (A, B, C and D).

The efficacy of K-means SMOTE is tested using different types of classifiers. To do so, we employ both commonly used and/or state-of-the-art oversamplers as benchmarking methods: Random oversampling (ROS), SMOTE and Borderline-SMOTE (B-SMOTE) [24]. Also as a baseline score we include classification results without the use of any resampling method.

This paper is organized in 5 sections: section 3.2 provides an overview of the state-of-art, section 3.3 describes the proposed methodology, section 3.4 covers the results and discussion and section 3.5 presents the conclusions taken from this study.

This paper's main contributions are:

- Propose a cluster-based multiclass oversampling method appropriate for LULC classification and compare its performance with the remaining oversamplers in a multiclass context with seven benchmark LULC classification datasets. Allows us to check the oversamplers' performance across benchmark LULC datasets.
- Introducing a cluster-based oversampling algorithm within the remote sensing domain, as well as comparing its performance with the remaining oversamplers in a multiclass context.
- Make available to the remote sensing community the implementation of the algorithm in a Python library and the experiment's source code.

3.2. Imbalanced Learning Approaches

Imbalanced learning has been addressed in three different ways: over/undersampling, cost-sensitive training and changes/adaptations in the learning algorithms [75]. These approaches impact different phases of the learning process, while over/undersampling can be seen as a pre-processing step, cost-sensitive and changes in the algorithm imply a more customized and complex intervention in the algorithms. In this

section, we focus on previous work related with resampling methods, while providing a brief explanation of cost-sensitive and algorithmic level solutions.

All of the most common classifiers used for LULC classification tasks [1, 96] are sensitive to class imbalance [106]. Algorithm-based approaches typically focus on adaptations based on ensemble classification methods [107] or common non-ensemble based classifiers such as Support Vector Machines [108]. In [109], the reported results show that algorithm-based methods have comparable performance to resampling methods.

Cost-sensitive solutions refer to changes in the importance attributed to each instance through a cost matrix [110, 111, 112]. A relevant cost sensitive solution [110] uses the inverse class frequency (i.e., $1/|C_i|$, where C_i refers to the frequency of class i) to give higher weight to minority classes. Cui et al. [111] extended this method by adding a hyperparameter β to class weights as $(1-\beta)/(1-\beta^{|C_i|})$. When $\beta = 0$, no re-weighting is done. When $\beta \to 1$, weights are the inverse of the frequency class matrix. Another method [112] explores adaptations of Cross-entropy classification loss by adding different formulations of class rectification loss.

Resampling (over/undersampling) is the most common approach to imbalanced learning in machine learning in general and remote sensing in particular [98]. The generation of artificial instances (i.e., augmenting the dataset), based on rare instances, is done independently of any other step in the learning process. Once the procedure is applied, any standard machine learning algorithm can be used. Its simplicity makes resampling strategies particularly appealing for any user (especially the non-sophisticated user) interested in applying several classifiers, while maintaining a simple approach. It is also important to notice that over/undersampling methods can also be easily applied to multiclass problems, common in LULC classification tasks.

3.2.1. Non-informed resampling methods

There are two main non-informed resampling methods. Random Oversampling (ROS) generates artificial instances through random duplication of minority class instances. This method is used in remote sensing for its simplicity [113, 114], even though its mechanism makes the classifier prone to overfitting [115]. [114] found that using ROS returned worse results than keeping the original imbalance in their dataset.

A few of the recent remote sensing studies employed Random Undersampling (RUS) [116], which randomly removes instances belonging to majority classes. Although it's not as prone to overfitting as ROS, it incurs into information loss by eliminating instances from the majority class [98], which can be detrimental to the quality of the results.

Another disadvantage of non-informed resampling methods is their performance-wise inconsistency across classifiers. ROS' impact on the Indian Pines dataset was found inconsistent between Random Forest Classifiers (RFC) and Support Vector Machines (SVM) and lowered the predictive power of an artificial neural network (ANN) [100]. Similarly, RUS is found to generally lead to a lower overall accuracy due to the associated information loss [100].

3.2.2. Heuristic methods

The methods presented in this section appear as a means to overcome the insufficiencies found in non-informed resampling. They use either local or global information to generate new, relevant, non-duplicated instances to populate the minority classes and/or remove irrelevant instances from majority classes. In a comparative analysis between over- and undersamplers' performance for LULC classification [7] using the rotation forest ensemble classifier, authors found that oversampling methods consistently outperform undersampling methods. This result led us to exclude undersampling from our study.

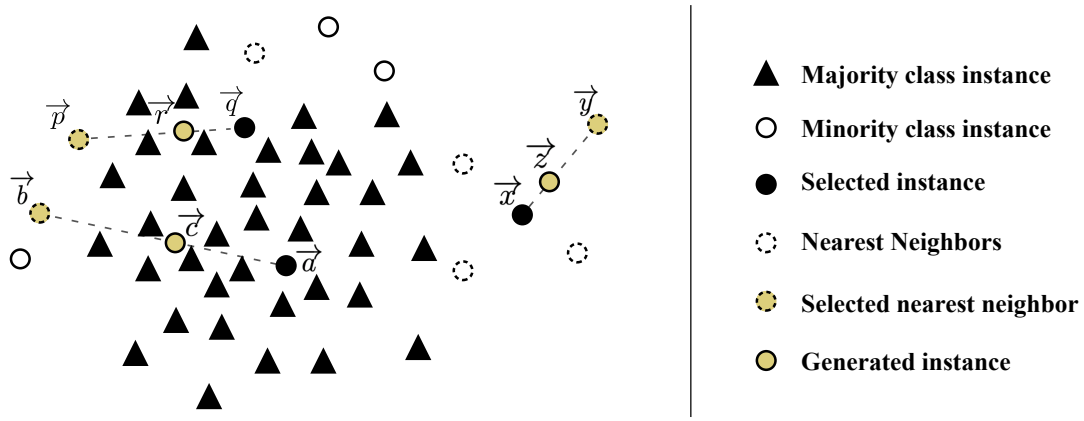


Figure 3.2.: Example of SMOTE's data generation process. SMOTE randomly selects instance $\vec{x}$ and randomly selects one of its k-nearest neighbors $\vec{y}$ to produce $\vec{z}$. Noisy instance $\vec{r}$ was generated by randomly selecting $\vec{q}$ and randomly selecting its nearest neighbor $\vec{p}$ from a different minority class cluster. Noisy instance $\vec{c}$ was generated by randomly selecting the noisy minority class instance $\vec{a}$ and one of its nearest neighbors $\vec{b}$.

SMOTE [22] was the first heuristic oversampling algorithm to be proposed and has been the most popular one since then, likely due to its fair degree of simplicity and quality of generated data. It takes a random minority class sample and introduces synthetic instances along the line segment that join a random k minority class nearest neighbor to the selected sample. Specifically, a single synthetic sample $\vec{z}$ is generated within the line segment of a randomly selected minority class instance $\vec{x}$ and one of its k nearest neighbors $\vec{y}$ such that $\vec{z} = \alpha\vec{x} + (1-\alpha)\vec{y}$, where α is a random real number between 0 and 1, as shown in Figure 3.2.

A number of studies implement SMOTE within the LULC classification context and reported improvements on the quality of the trained predictors [117, 118]. Another study proposes an adaptation of SMOTE on an algorithmic level for deep learning applications [119]. This method combines both typical computer vision data augmentation techniques, such as image rotation, scaling and flipping on the generated instances to populate minority classes. Another algorithmic implementation is the variational semi-supervised learning model [120]. It consists of a generative model that allows learning from both labeled and unlabeled instances while using SMOTE to balance the data.

Despite SMOTE's popularity, its limitations have motivated the development of more sophisticated oversampling algorithms [23, 24, 25, 121, 122, 123]. [23] identify four major weaknesses of the SMOTE algorithm, which can be summarized as:

1. Generation of noisy instances due to random selection of a minority instance to oversample. The random selection of a minority instance makes SMOTE oversampling prone to the amplification of existing noisy data. This has been addressed by variants such as B-SMOTE [24] and ADASYN [123].
2. Generation of noisy instances due to the selection of the k nearest neighbors. In the event an instance (or a small number thereof) is not noisy but is isolated from the remaining clusters, known as the "small disjuncts problem" [124], much like sample $\vec{b}$ from Figure 3.2, the selection of any nearest neighbor of the same class will have a high likelihood of producing a noisy sample.
3. Generation of nearly duplicated instances. Whenever the linear interpolation is done between two instances that are close to each other, the generated instance becomes very similar to its parents and

increases the risk of overfitting. G-SMOTE [23] attempts to address both the k nearest neighbor selection mechanism problem as well as the generation of nearly duplicated instances problem.

4. Generation of noisy instances due to the use of instances from two different minority class clusters. Although an increased k could potentially avoid the previous problem, it can also lead to the generation of artificial data between different minority clusters, as depicted Figure 3.2 with the generation of point $\overrightarrow{r}$ using minority class instances $\overrightarrow{p}$ and $\overrightarrow{q}$. Cluster-based oversampling methods attempt to address this problem.

This last issue, the generation of noisy instances due to the existence of several minority class clusters, is particularly relevant in remote sensing. It is frequent that instances belonging to the same minority class can have different spectral signatures, meaning that they will be clustered in different parts of the input space. For example, in the classification of a hyperspectral scene dominated by agricultural activities, patches relating to urban areas may constitute a minority class. These patches frequently refer to different types of land use, such as housing regions, small gardens, asphalt roads, etc., all these containing different spectral signatures. In this context, the use of SMOTE will lead to the generation of noisy instances of the minority class. This problem can be efficiently mitigated through the use of a cluster-based oversampling method. According to our literature review cluster-based oversampling approaches have never been applied in the context of remote sensing. On the other hand, while there are references of the application of cluster-based oversampling in the context of machine learning [25, 121, 122, 125], the multiclass case is rarely addressed, which is a fundamental requirement for the application of oversampling in the context of LULC.

Cluster-based oversampling approaches introduce an additional layer to SMOTE's selection mechanism, which is done through the inclusion of a clustering process. This ensures that both between-class data balance and within-class balance is preserved. The self-organizing map oversampling (SOMO) [122] algorithm transforms the dataset into a 2-dimensional input, where the areas with the highest density of minority samples are identified. SMOTE is then used to oversample each of the identified areas separately. ClUstered REsampling SMOTE (CURE-SMOTE) [121] applies a hierarchical clustering algorithm to discard isolated minority instances before applying SMOTE. Although it avoids noise generation problems, it ignores within-class data distribution. Another method [125] uses K-means to cluster the entire input space and applies SMOTE to clusters with the fewest instances, regardless of their class label. The label of the generated instance is copied from one of its parents. This method cannot ensure a balanced dataset since class imbalance is not specifically addressed, but rather dataset imbalance.

K-means SMOTE [25] avoids noisy data generation by modifying the data selection mechanism. It employs k-means clustering to identify safe areas using cluster-specific Imbalance Ratio (IR, defined by $\frac{count(C_{majority})}{count(C_{minority})}$) and determine the quantity of generated samples per cluster based on a density measure. These samples are finally generated using the SMOTE algorithm. The K-means SMOTE's data generation process is depicted in Figure 3.3. Note that the number of samples generated for each cluster varies according to the sparsity of each cluster (the sparser the cluster is, the more samples will be generated) and a cluster is rejected if the cluster's IR surpasses the threshold. Therefore, this method can be combined with any data generation mechanism, such as G-SMOTE. Also K-means SMOTE includes the SMOTE algorithm as a special case when the number of clusters is set to one. Consequently, K-means SMOTE returns results as good as or better than SMOTE.

Although no other study was found to implement cluster-based oversampling, another study [2] compared the performance of SMOTE, ROS, ADASYN, B-SMOTE and G-SMOTE in a highly imbalanced LULC classification dataset. The authors found that G-SMOTE consistently outperformed the remaining oversampling algorithms regardless of the classifier used.

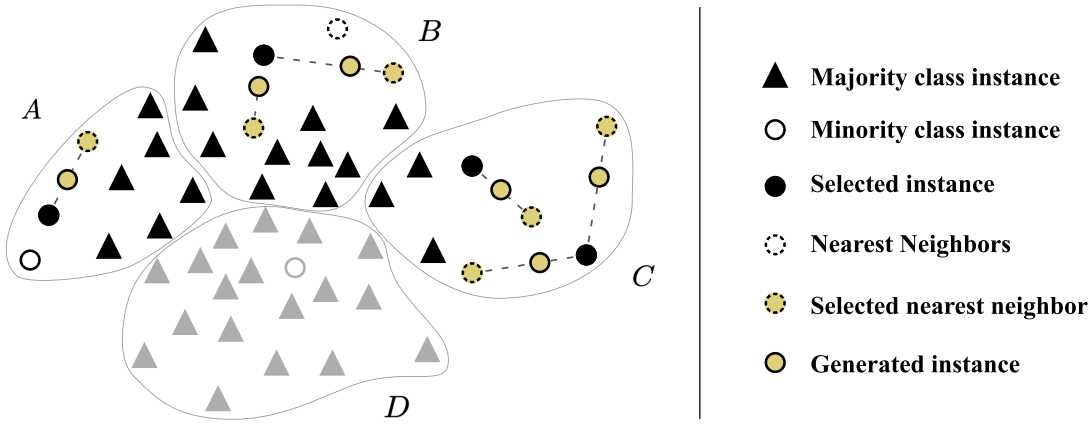


Figure 3.3.: Example of K-means SMOTE's data generation process. Clusters A, B and C are selected for oversampling, whereas cluster D was rejected due to its high imbalance ratio. The oversampling is done using the SMOTE algorithm and the k nearest neighbors selection only considers instances within the same cluster.

3.3. Methodology

The purpose of this work is to understand the performance of K-means SMOTE as opposed to other popular and/or state-of-the-art oversamplers for LULC classification. This is done using 7 datasets with predominantly land use information, along with 3 evaluation metrics and 3 classifiers to evaluate the performance of oversamplers. In this section we describe the datasets, evaluation metrics, oversamplers, classifiers and software used as well as the procedure developed.

3.3.1. Datasets

The datasets used were extracted from publicly available hyperspectral scenes. Information regarding each of these scenes is provided in this subsection. The data collection and preprocessing pipeline is shown in Figure 3.4 and is common to all hyperspectral scenes:

1. Data collection of publicly available hyperspectral scenes. The original hyperspectral scenes and ground truth data were collected from a single publicly available data repository available here.
2. Conversion of each hyperspectral scene to a structured dataset and removal of instances with no associated LULC class. This done to reshape the dataset from $(h, w, b + gt)$ into a conventional dataframe of shape $(h * w, b + gt)$, where gt, h, w and b represents the ground truth, height, width and number of bands in the scene, respectively. The pixels without ground truth information are discarded from further analysis.
3. Stratified random sampling to maintain similar class proportions on a sample of 10% of each dataset. This is done by computing the relative class frequencies in the original hyperspectral scene (minus the class representing no ground truth availability) and retrieving a sample that ensures the original relative class frequencies remain unchanged.
4. Removal of instances belonging to a class with frequency lower than 20 or higher than 1000. This is done to maintain the datasets to a practicable size due to computational constraints, while conserving the relative LULC class frequencies and data distribution.

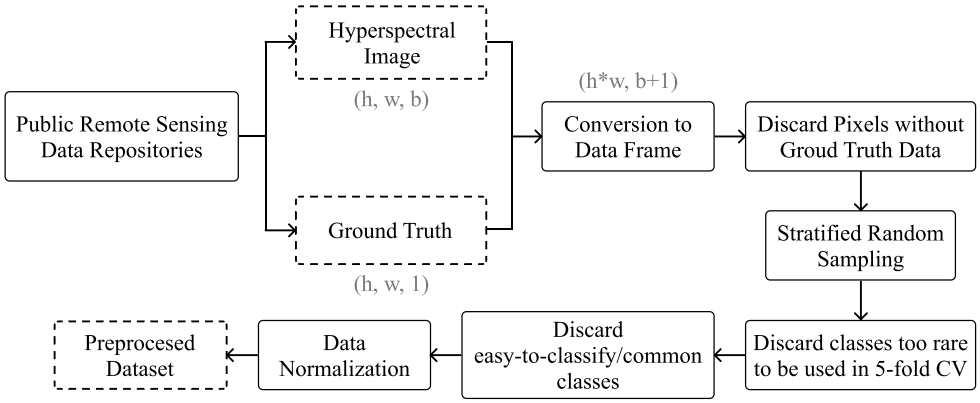


Figure 3.4.: Data collection and preprocessing pipeline.

Dataset	Features	Instances	Min. Instances	Maj. Instances	IR	Classes
Botswana	145	288	20	41	2.05	11
Pavia Centre	102	3898	278	879	3.16	7
Kennedy Space Center	176	497	23	80	3.48	11
Salinas A	224	535	37	166	4.49	6
Pavia University	103	2392	89	679	7.63	8
Salinas	224	4236	91	719	7.9	15
Indian Pines	220	984	21	236	11.24	11

Table 3.1.: Description of the datasets used for this experiment.

5. Data normalization using the MinMax scaler. This ensures all features (i.e., bands) are in the same scale. In this case, the data was rescaled between 0 and 1.

Table 3.1 provides a description of the final datasets used for this work, sorted according to its IR. Figure 3.5 shows the original hyperspectral scene out of which the dataset used in this experiment were extracted. In the representation of the ground truth of these scenes, the blue regions in the ground truth of each hyperspectral scene represent unlabeled regions (i.e., no ground truth is available). Particularly, in the Botswana and Kennedy Space Center scenes the truth was photointerpreted in more limited regions of the scene. However, the scenes are still represented as they are in order to maintain a standardized analysis over all datasets extracted for the experiment.

Botswana

The Botswana scene was acquired by the Hyperion sensor on the NASA EO-1 satellite over the Okavango Delta, Botswana in 2001-2004 at a 30m spatial resolution. Data preprocessing was performed by the UT Center for Space Research. The scene comprises a 1476×256 pixels with 145 bands and 14 classes regarding land cover types in seasonal and occasional swamps, as well as drier woodlands (see Figure 3.5a). The classes with rare instances are Short mopane and Hippo grass.

Pavia Center and University

Both Pavia Center and University scenes were acquired by the ROSIS sensor. These scenes are located in Pavia, northern Italy. Pavia Center is a 1096×1096 pixels image with 102 spectral bands, whereas Pavia

University is a 610×610 pixels image with 103 spectral bands. Both images have a geometrical resolution of 1.3m and their ground truths are composed of 9 classes each (see Figures 3.5b and 3.5c). After data preprocessing, the classes with rare instances are Asphalt and Bitumen (the class Shadows was removed for being too rare for cross validation after random sampling).

Kennedy Space Center

The Kennedy Space Center scene was acquired by the AVIRIS sensor over the Kennedy Space Center, Florida, on March 23, 1996. Out of the original 224 bands, water absorption and low SNR bands were removed and a total of 176 bands at a spatial resolution of 18m are used. The scene is a 512×614 pixel image and contains a total of 16 classes (see figure 3.5d). The classes with rare instances are hardwood swamp, slash pine and willow swamp (both hardwood swamp and slash pine were removed for being too rare for cross validation after random sampling).

Salinas and Salinas-A

These scenes were collected by the AVIRIS sensor over Salinas Valley, California and contain at-sensor radiance data. Salinas is a 512×217 pixels image with 224 bands and 16 classes regarding vegetables, bare soil and vineyard fields (see Figure 3.5e). Salinas-A, a subscene of Salinas, comprises 86×83 pixels and contains 6 classes regarding vegetables (see Figure 3.5f). These scenes have a geometrical resolution of 3.7m. Salinas-A's minority class has the label "Brocoli_green_weeds_1" and Salina's minority class has the label "Lettuce_romaine_6wk"

Indian Pines

The Indian Pines scene [126] was collected on June 12, 1992 and consists of AVIRIS hyperspectral image data covering the Indian Pine Test Site 3, located in North-western Indiana, USA. As a subset of a larger scene, it is composed of 145×145 pixels (see Figure 3.5g) and 220 spectral reflectance bands in the wavelength range 400 to 2500 nanometers at a spatial resolution of 20m. Approximately two thirds of this scene is composed by agriculture and the other third is composed of forest and other natural perennial vegetation. Additionally, the scene also contains low density buildup areas. The classes with rare instances are Alfalfa, Oats, Grass-pasture-mowed, Wheat and Stone-Steel-Towers (which removed for being too rare for cross validation after random sampling). After data preprocessing, the classes with rare instances are Corn, Buildings-Grass-Trees-Drives and Grass-Pasture.

3.3.2. Machine Learning Algorithms

To assess the quality of the K-means SMOTE algorithm, three other oversampling algorithms were used for benchmarking. ROS and SMOTE were chosen for their simplicity and popularity. B-SMOTE chosen as a popular variation of the SMOTE algorithm. We also include the classification results of no oversampling (NONE) as a baseline.

To assess the performance of each oversampler, we use the classifiers Logistic Regression (LR) [127], K-Nearest Neighbors (KNN) [128] and Random Forest (RF) [129]. This choice was based on the classifiers' popularity for LULC classification, learning type and training time [96, 100]. Since this is a multinomial classification task, for the LR classification we adopted a one-versus-all approach for each label. The predicted label is assigned according to the class predicted with highest probability.

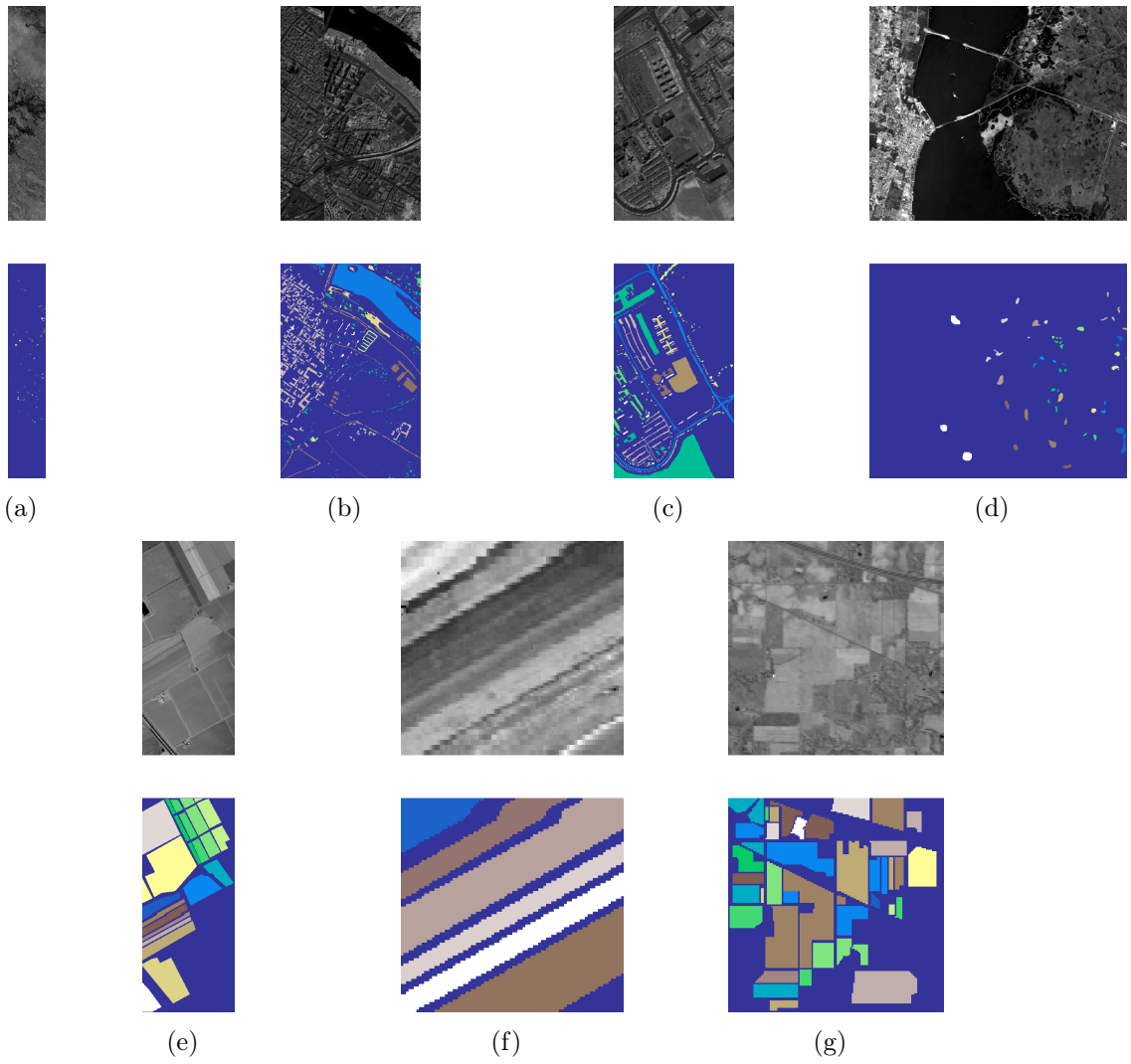


Figure 3.5.: Gray scale visualization of a band (top row) and ground truth (bottom row) of each scene used in this study. (a) Botswana, (b) Pavia Center, (c) Pavia University, (d) Kennedy Space Center, (e) Salinas, (f) Salinas A, (g) Indian Pines.

3.3.3. Evaluation Metrics

Most of the satellite-based LULC classification studies (nearly 80%) employ *Overall Accuracy* (OA) and the *Kappa Coefficient* [96]. Although, some authors argue that both evaluation metrics, even when used simultaneously, are insufficient to fully address the area estimation and uncertainty information needs [130, 131]. Other metrics like User's Accuracy (or *Precision*) and Producer's Accuracy (or *Recall*) are also common metrics to evaluate per-class prediction power. These metrics consist of ratios employing the True and False Positives (*TP* and *FP*, number of correctly/incorrectly classified instances of a given class) and True and False Negatives (*TN* and *FN*, number of correctly/incorrectly classified instances as not belonging to a given class). These metrics are formulated as $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$. While metrics like OA and *Kappa Coefficient* are significantly affected by imbalanced class distributions, *F-Score* is less sensitive to data imbalance and a more appropriate choice for performance evaluation [132].

The datasets used present significantly high IRs (see Table 3.1). Therefore, it is especially important to attribute equal importance to the predictive power of all classes, which does not happen with OA and *Kappa Coefficient*. In this study, we employ 3 evaluation metrics: 1) *G-mean*, since it is not affected by skewed class distributions, 2) *F-Score*, as it proved to be a more appropriate metric for this problem when compared to other commonly used metrics [132], and 3) *Overall Accuracy*, for discussion purposes.

- The *G-mean* consists of the geometric mean of $Specificity = \frac{TN}{TN+FP}$ and *Sensitivity* (also known as *Recall*). For multiclass problems, The *G-mean* is expressed as:

$$G\text{-}mean = \sqrt{\overline{Sensitivity} \times \overline{Specificity}}$$

- *F-score* is the harmonic mean of *Precision* and *Recall*. The *F-score* for the multi-class case can be calculated using their average per class values [133]:

$$F\text{-}score = 2\frac{\overline{Precision} \times \overline{Recall}}{\overline{Precision} + \overline{Recall}}$$

- *Overall Accuracy* is the number of correctly classified instances divided by the total amount of instances. Having c as the label of the various classes, *Accuracy* is given by the following formula:

$$Accuracy = \frac{\sum_c \mathrm{TP}_c}{\sum_c (\mathrm{TP}_c + \mathrm{FP}_c)}$$

In the case of *G-mean* and *F-score*, both metrics are computed for each label and their unweighted mean is calculated (*i.e.*, following a "macro" approach). In this study we assume that all labels have an equivalent importance for the classification task.

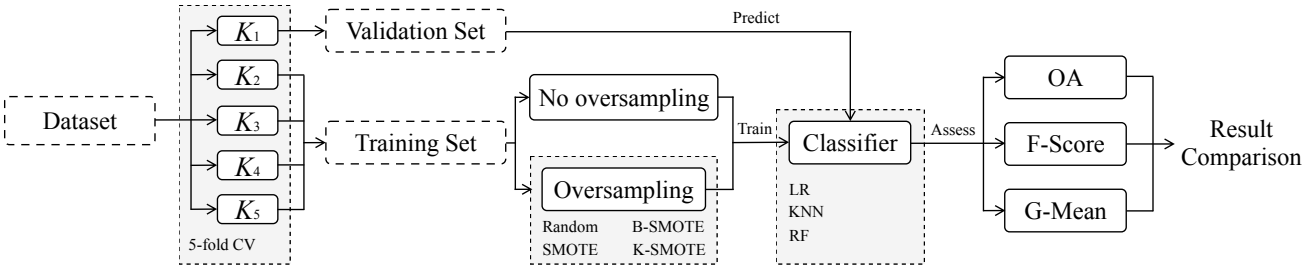


Figure 3.6.: Experimental procedure. The performance metrics are averaged over the 5 folds across each of the 3 different initializations of this procedure for a given combination of oversampler, classifier and hyperparameter definition.

Classifier	Hyperparameters	Values
LR	maximum iterations	10000
KNN	# neighbors	3, 5, 8
RF	maximum depth	None, 3, 6
	# estimators	50, 100, 200
Oversampler		
K-means SMOTE	# neighbors	3, 5
	# clusters (as % of number of instances)	1*, 0.1, 0.3, 0.5, 0.7, 0.9
	Exponent of mean distance	auto, 2, 5, 7
	IR threshold	auto, 0.5, 0.75, 1.0
SMOTE	# neighbors	3, 5
BORDERLINE SMOTE	# neighbors	3, 5

Table 3.2.: Hyper-parameters grid. * One cluster is generated in total, a corner case that mimics the behavior of SMOTE

3.3.4. Experimental Procedure

The procedure for the experiment started with the definition of a hyperparameter search grid, where a list of possible values for each relevant hyperparameter in both classifiers and oversamplers is stored. Based on this search grid, all possible combinations of oversamplers, classifiers and hyperparameters are formed. Finally, for each dataset, hyperparameter combination and initialization we use the evaluation strategy shown in Figure 3.6: k-fold cross-validation strategy where $k = 5$ to train each model defined and save the averaged scores of each split.

In the 5-fold cross validation strategy, a combination of oversampler, classifier and hyperparameters vector is fit 5 times per dataset. Before the training phase, the training set (containing $\frac{4}{5}$ of the dataset) is oversampled using one of the methods described (except for the baseline method NONE), creating an augmented dataset with the exact same number of instances for each class. The newly formed training dataset is used to train the classifier and the test set ($\frac{1}{5}$ of the dataset) is used to evaluate the performance of the classifier. The evaluation scores are then averaged over the 5 times the process is repeated. The range of hyperparameters used are shown in table 3.2. The definition of hyperparameters for the K-means SMOTE oversampler is defined according to the recommendations discussed in the original K-means SMOTE paper [25].

Classifier	Metric	NONE	ROS	SMOTE	B-SMOTE	K-SMOTE
LR	Accuracy	0.906 ± 0.039	0.904 ± 0.04	0.904 ± 0.04	0.901 ± 0.04	**0.909 ± 0.038**
LR	F-score	0.891 ± 0.041	0.893 ± 0.042	0.893 ± 0.042	0.890 ± 0.042	**0.898 ± 0.04**
LR	G-mean	0.936 ± 0.025	0.940 ± 0.025	0.940 ± 0.025	0.937 ± 0.025	**0.943 ± 0.024**
KNN	Accuracy	0.879 ± 0.043	0.865 ± 0.048	0.867 ± 0.05	0.862 ± 0.054	**0.881 ± 0.045**
KNN	F-score	0.859 ± 0.05	0.853 ± 0.049	0.861 ± 0.047	0.851 ± 0.053	**0.866 ± 0.048**
KNN	G-mean	0.919 ± 0.03	0.920 ± 0.029	0.926 ± 0.027	0.918 ± 0.03	**0.927 ± 0.027**
RF	Accuracy	0.898 ± 0.032	0.901 ± 0.031	0.900 ± 0.031	0.898 ± 0.032	**0.905 ± 0.031**
RF	F-score	0.879 ± 0.041	0.885 ± 0.037	0.887 ± 0.036	0.883 ± 0.037	**0.891 ± 0.036**
RF	G-mean	0.930 ± 0.024	0.935 ± 0.022	0.937 ± 0.021	0.935 ± 0.021	**0.939 ± 0.02**

Table 3.3.: Mean cross-validation scores of oversamplers.

3.3.5. Software Implementation

The experiment was implemented using the Python programming language, using the Scikit-Learn [86], Imbalanced-Learn [134], Geometric-SMOTE, Cluster-Over-Sampling and Research-Learn libraries. All functions, algorithms, experiments and results are provided at the GitHub repository of the project.

3.4. Results & Discussion

When evaluating the performance of an algorithm across multiple datasets, it is generally recommended to avoid direct score comparisons and use classification rankings instead [135]. This is done by assigning a ranking to oversamplers based on the different combinations of classifier, metric and dataset used. These rankings are also used for the statistical analyses presented in Section 3.4.2.

The rank values are assigned based on the mean validation scores resulting from the experiment described in Section 3.3. The averaged ranking results are computed over 3 different initialization seeds and a 5 fold cross validation scheme, returning a real number within the interval $[1, 5]$.

The hyperparameter optimization ensures that both oversamplers and classifiers are well adapted to each of the datasets used in the experiment. Specifically, the optimization of classifiers' hyperparameters is not particularly relevant since our focus is to study the relative performance scores across oversamplers. This will provide insights on the quality of the artificial data generated by each oversampler. The classifiers' hyperparameter tuning was done to avoid the over/underfitting of classifiers, since they are trained on the same data subsets along with artificial data generated with different methods.

3.4.1. Results

The mean ranking of oversamplers is presented in Figure 3.7. This ranking was computed by averaging the ranks of the mean cross-validation scores per dataset, oversampler and classifier. K-means SMOTE achieves the best mean ranking across datasets with low standard deviation.

The mean cross-validation scores are shown in Table 3.3. As discussed previously in this section, the disparity of performance levels across datasets makes the analysis of these scores less informative.

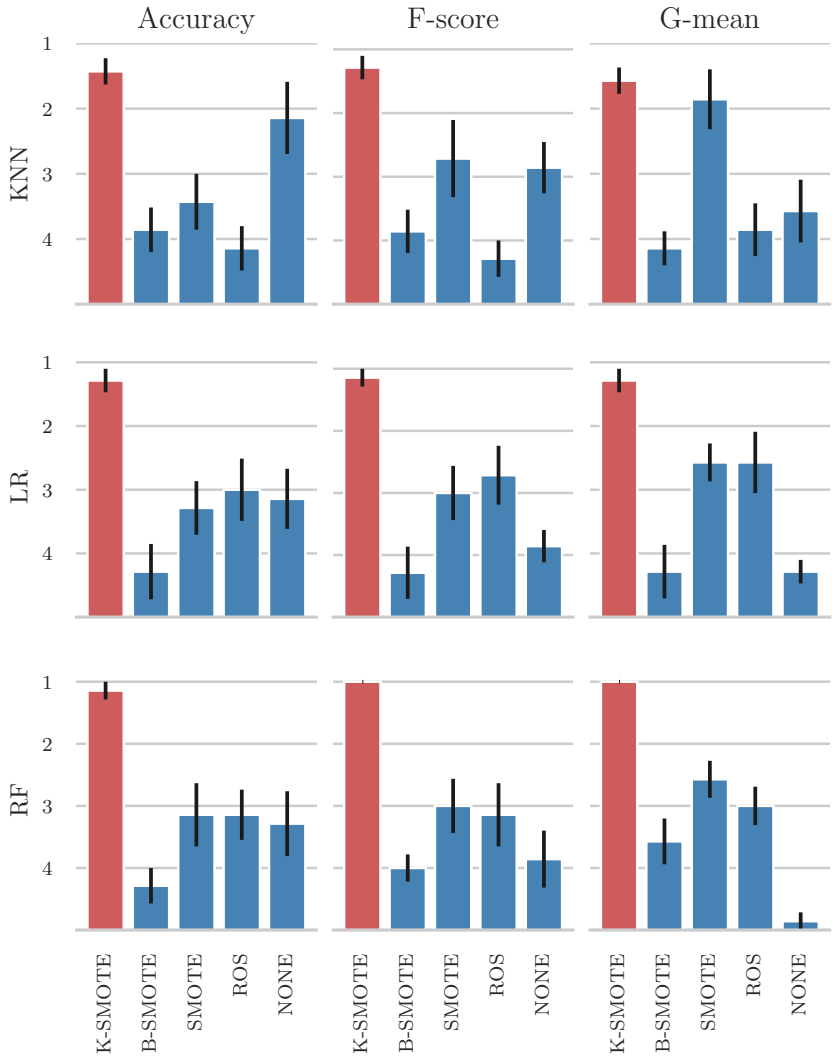


Figure 3.7.: Results for mean ranking of oversamplers across datasets.

The mean cross-validation scores for each dataset are presented in Table A.1 (see appendix). This table allows the direct comparison of the performance metrics being analysed.

3.4.2. Statistical Analysis

The experiment's multi-dataset context was used to perform a Friedman test [136]. Table 3.4 shows the results obtained in the Friedman test performed, where the null hypothesis is rejected in all cases. The rejection of the null hypothesis implies that the differences between the differences among the different oversamplers are not random, in other words, these differences are statistically significant.

Classifier	Metric	p-value	Significance
LR	Accuracy	9.8e-03	True
LR	F-score	2.3e-03	True
LR	G-mean	9.8e-04	True
KNN	Accuracy	4.3e-03	True
KNN	F-score	4.3e-03	True
KNN	G-mean	3.0e-03	True
RF	Accuracy	5.5e-03	True
RF	F-score	2.9e-03	True
RF	G-mean	1.8e-04	True

Table 3.4.: Results for Friedman test. Statistical significance is tested at a level of $\alpha = 0.05$. The null hypothesis is that there is no difference in the classification outcome across oversamplers.

A Wilcoxon signed-rank test [137] was also performed to understand whether K-means SMOTE's superiority was statistically significant across datasets and oversamplers, as suggested in [135]. This method is used as an alternative to the paired Student's t-test, since the distribution of the differences between the two samples cannot be assumed as normally distributed. The null hypothesis of the test is that K-means SMOTE's performance is similar to the compared oversampler (i.e., the oversamplers used follow a symmetric distribution around zero).

Dataset	NONE	ROS	SMOTE	B-SMOTE
Botswana	**3.1e-02**	**3.9e-03**	**3.9e-03**	**3.9e-03**
Pavia Centre	**3.1e-02**	**3.9e-03**	**1.2e-02**	**3.9e-03**
Kennedy Space Center	**3.1e-02**	**3.9e-03**	**2.7e-02**	**3.9e-03**
Salinas A	**3.1e-02**	**3.9e-03**	**1.2e-02**	**3.9e-03**
Pavia University	**3.1e-02**	**3.9e-03**	**3.9e-03**	**3.9e-03**
Salinas	**3.1e-02**	5.5e-02	**2.7e-02**	**3.9e-03**
Indian Pines	**3.1e-02**	**3.9e-03**	**7.8e-03**	**3.9e-03**

Table 3.5.: *p-values* of the Wilcoxon signed-rank test. Boldface values are statistically significant at a significance level of $\alpha = 0.05$.

3.4.3. Discussion

The mean rankings presented in Figure 3.7 show that on average, K-means SMOTE produced the best results for every classifier and performance metric used. This is due to the clustering phase and subsequent selection of data to be considered for oversampling. By successfully clustering and selecting the relevant areas in the data space to oversample, the generation of artificial instances is done only in the context of minority regions that represent well their spectral signature.

As previously discussed, the direct comparison of performance metrics averaged over various datasets is not recommended due to the varying levels of performance of classifiers across datasets [135]. Nonetheless, these results are shown in Table 3.3 to provide a fuller picture of the results obtained in the experiment. We found that on average K-means SMOTE provides increased performance, regardless of the classifier and performance metric used. More importantly, K-means SMOTE guaranteed a more consistent performance across datasets and with less variability, which can be attested in Figure 3.7 and Tables 3.3 and A.1.

As discussed in Subsection 3.3.3, Evaluation Metrics, our results are consistent with the findings in [130, 131]. Particularly, we consider the results obtained in our experiment using Overall Accuracy to be less informative than the results obtained with the remaining performance metrics, since this metric is affected by imbalanced class distributions. The majority class bias in this metric can be observed in our experiment in Figure 3.7 with the classifiers LR and KNN, where the control method (NONE) is only outperformed by K-means SMOTE. This effect is observed with more detail in Table 3.3, where the benchmark oversamplers are outperformed by the control method in 16 out of 63 tests (approximately 25%). Out of these, most refer to tests using overall accuracy among the four datasets with highest IR, showing the overall accuracy's class imbalance bias discussed in [130, 131]. The K-means SMOTE oversampler is only outperformed by the control method in 3 of tests (all of them using overall accuracy). This is an improvement over the benchmark oversamplers, showing that generally K-means SMOTE is the best choice even when overall accuracy is used as the main performance metric.

In the majority of the cases, K-means SMOTE was able to generate higher quality data due to the non-random selection of data spaces to oversample. This can be seen in the performance of the classifiers trained on top of this data generation step, making it a more informed data generation method in the context of LULC.

The performance of both oversamplers and classifiers is generally dependent on the dataset being used. Although both absolute and relative scores between the different oversamplers are dependent on the choice of metric and classifier, K-means SMOTE's relative performance is consistent across datasets and generally outperforms the remaining oversampling methods in 56 of the 63 tests (approximately 89%). The mean cross-validation results found in Table A.1 show that performance-wise, K-means SMOTE is always better than or as good as SMOTE, with the exception of 4 situations (representing 6% of the tests done), in which cases the percentage point difference is neglectable (≤ 0.1 percentage points).

The statistical tests showed that not only there is a statistically significant difference across the oversamplers used in this problem (found in the Friedman test presented in Table 3.4), but also that K-means SMOTE's superior performance is statistically significant at a level of 0.05 in 27 out of 28 tests in the Wilcoxon signed-rank test shown in Table 3.5 (approximately 96% of the tests performed). This shows that, in most cases, the usage of k-Means SMOTE improves the quality of LULC classification when compared to using SMOTE in its original format, which remains the most popular oversampler among the remote sensing community.

Although the usage of K-means SMOTE successfully captured the spectral signatures of the minority classes, it was done using K-means, a problem-agnostic clusterer. Consequently, the implementation of this method using a GIS-specific clusterer that considers the geographical traits of different regions (e.g., using the sampled pixels' geographical coordinates), may be a promising direction towards the development of more appropriate oversampling techniques in the remote sensing domain.

3.5. Conclusion

This research paper was motivated by the challenges faced when classifying rare classes for LULC mapping. Cluster-based oversampling is especially useful in this context because the spectral signature of a given class often varies, depending on its geographical distribution and the time period within which the image was acquired. This induces the representation of minority classes as small clusters in the input space. As a result, training a classifier capable of identifying LULC minority classes in the hyper/multi-spectral scene over different areas or periods becomes particularly challenging. The clustering procedure, performed before the data generation phase, allows for a more accurate generation of minority samples, as it identifies these minority clusters.

A number of existing methods to address the imbalanced learning problem were identified and their limitations discussed. Typically, algorithm-based approaches and cost-sensitive solutions are not only difficult to implement, but they are also context dependent. In this paper we focused on oversampling methods due to their widespread usage, easy implementation and flexibility. Specifically, this paper demonstrated the efficacy of a recent oversampler, K-Means SMOTE, applied in a multi-class context for Land Cover Classification tasks. This was done with sampled data from seven well known and naturally imbalanced benchmark datasets: Indian Pines, Pavia Center, Pavia University, Salinas, Salinas A, Botswana and Kennedy Space Center. For each combination of dataset, oversampler and classifier, the results of every classification task was averaged across a 5 fold stratification strategy with 3 different initialization seeds, resulting in a mean validation score of 15 classification tasks. The mean validation score of each combination was then used to perform the analyses presented in this report.

In 56 out of 63 classification tasks (approximately 89%), K-means SMOTE led to better results than ROS, SMOTE, B-SMOTE and no oversampling. More importantly, we found that K-Means SMOTE is always better or equal than the second best oversampling method. K-means SMOTE's performance was independent from both the classifier and performance metric under analysis. In general, K-means SMOTE shows a higher performance among the non tree-based classifiers employed (LR and KNN) when compared with the remaining oversamplers, where these oversamplers generally failed to improve the quality of classification. Although these findings are case dependent, they are consistent with the results presented in [25]. The proposed method also had the most consistent results across datasets, since it produced the lowest standard deviations across datasets in 7 out of 9 cases for both analyses, either based on ranking or mean cross-validation scores.

The proposed algorithm is a generalization of the original SMOTE algorithm. In fact, the SMOTE algorithm represents a corner case of K-means SMOTE i.e., when the number of clusters equals to 1. Its data selection phase differs from the one used in SMOTE and Borderline SMOTE, providing artificially augmented datasets with less noisy data than the commonly used methods. This allows the training of classifiers with better defined decision boundaries, especially in the most important regions of the data space (the ones populated by a higher percentage of minority class instances).

As stated previously, the usage of this oversampler is technically simple. It can be applied to any classification problem relying on an imbalanced dataset, alongside any classifier. K-means SMOTE is available as an open source implementation for the Python programming language (see Subsection 3.3.5). Consequently, it can be a useful tool for both remote sensing researchers and practitioners.

4. Increasing the Effectiveness of Active Learning: Introducing Artificial Data Generation in Active Learning for Land Use/Land Cover Classification

Published as Joao Fonseca, Georgios Douzas, Fernando Bacao, in Remote Sensing, 2021

In remote sensing, Active Learning (AL) has become an important technique to collect informative ground truth data "on-demand" for supervised classification tasks. In spite of its effectiveness, it is still significantly reliant on user interaction, which makes it both expensive and time consuming to implement. Most of the current literature focuses on the optimization of AL by modifying the selection criteria and the classifiers used. Although improvements in these areas will result in more effective data collection, the use of artificial data sources to reduce human-computer interaction remains unexplored. In this paper, we introduce a new component to the typical AL framework, the data generator, a source of artificial data to reduce the amount of user-labeled data required in AL. The implementation of the proposed AL framework is done using Geometric SMOTE as data generator. We compare the new AL framework to the original one using similar acquisition functions and classifiers over three AL-specific performance metrics in seven benchmark datasets. We show that this modification of the AL framework significantly reduces cost and time requirements for a successful AL implementation in all of the datasets used in the experiment.

Keywords: Active Learning; Artificial Data Generation; Land Use/Land Cover Classification; Oversampling; SMOTE

4.1. Introduction

The technological development of air and spaceborne sensors, as well as the increasing number of remote sensing missions have allowed the continuous collection of large amounts of high quality remotely sensed data. This data is often composed of multi and hyper spectral satellite imagery, essential for numerous applications, such as Land Use/Land Cover (LULC) change detection, ecosystem management [138], agricultural management [139], water resource management [140], forest management, and urban monitoring [1]. Despite LULC maps being essential for most of these applications, their production is still a challenging task [95, 96]. They can be updated using one of the following strategies:

1. Photo-interpretation. This approach consists of evaluating a patch's LULC class by a human operator based on orthophoto and satellite image interpretation [141]. This method guarantees a decent level of accuracy, as it is dependent on the interpreter's expertise and human error. Typically,

it is an expensive, time-consuming task that requires the expertise of a photo-interpreter. This task is also frequently applied to obtain ground-truth labels for training and/or validating Machine Learning (ML) algorithms for related tasks [142, 143].

2. Automated mapping. This approach is based on the usage of a ML method or a combination of methods in order to obtain an updated LULC map. The development of a reliable automated method is still a challenge among the ML and remote sensing community, since the effectiveness of existing methods varies across applications and geographical areas [96]. Typically, this method requires the existence of ground-truth data, which is frequently outdated or nonexistent for the required time frame [138]. On the other hand, employing a ML method provides readily available and relatively inexpensive LULC maps. The increasing quality of state-of-the-art classification methods have motivated the application and adaptation of these methods in this domain [100].

3. Hybrid approaches. These approaches employ photo-interpreted data to augment the training dataset and improve the quality of automated mapping [144]. It attempts to accelerate the photo-interpretation process by selecting a smaller sample of the study area to be interpreted. The goal is to minimize the inaccuracies found in the LULC map by supplying high-quality ground-truth data to the automated method. The final (photo-interpreted) dataset consists of only the most informative samples, *i.e.*, patches that are typically difficult to classify for a traditional automated mapping method [145].

The latter method is best know as AL. It is especially useful whenever there is a shortage or even absence of ground-truth data and/or the mapping region does not contain updated LULC maps [33]. In a context of limited sample-collection budget, the collection of the most informative samples capable of optimally increasing the classification accuracy of a LULC map is of particular interest [33]. AL attempts to minimize the human-computer interaction involved in photo-interpretation by selecting the data points to include in the annotation process. These data points are selected based on an uncertainty measure and represent the points close to the decision borders. Afterwards, they are passed on for photo-interpretation and added to the training dataset, while the points with the lowest uncertainty values are ignored for photo-interpretation and classification. This process is repeated until a convergence criterion is reached [146].

The relevant work developed within AL is described in detail in Section 4.2. This paper attempts to address some of the challenges found in AL, mainly inherited from automated and photo-interpreted mapping: mapping inaccuracies and time consuming human-computer interactions. These challenges have different sources:

1. Human error. The involvement of photo-interpreters in the data labeling step carries an additional risk to the creation of LULC patches. The minimum mapping unit being considered, as well as the quality of the orthophotos and satellite images being used, are some of the factors that may lead to the overlooking of small-area LULC patches and label-noisy training data [4].

2. High-dimensional datasets. Although the amount of bands (*i.e.*, features) present in multi and hyper spectral images contain useful information for automated classification, they also introduce an increased level of complexity and redundancy in the classification step [5]. These datasets are often prone to the Hughes phenomenon, also known as the curse of dimensionality.

3. Class separability. Producing an LULC map considering classes with similar spectral signatures makes them difficult to separate [6]. A lower pixel resolution of the satellite images may also imply mixed-class pixels, which may lead to both lower class separability as well as higher risk of human error.

4. Existence of rare land cover classes. The varying morphologies of different geographical regions naturally implies an uneven distribution of land cover classes [7]. This is particularly relevant in the context of AL since the data selection method is based on a given uncertainty measure over data

points whose class label is unknown. Consequently, AL's iterative process of data selection may disregard wrongly classified land cover areas belonging to a minority class.

Research developed in the field of AL typically focus on the reduction of human error by minimizing the human interaction with the process through the development of more efficient classifiers and selection criteria within the generally accepted AL framework. Concurrently, the problem of rare land cover classes is rarely addressed. This is a frequent problem in the ML community, known as the Imbalanced Learning problem. This problem exists whenever there is an uneven between-class distribution in the dataset [74]. Specifically, most classifiers are optimized and evaluted using accuracy-like metrics, which are designed to work primarily with balanced datasets. Consequently, these metrics tend to introduce a bias towards the majority class by attributing an importance to each class proportional to its relative frequency [100]. As an example, such a classifier could achieve an overall accuracy of 99% on a binary dataset where the minority class represents 1% of the overall dataset and still be useless. A number of methods have been developed to deal with this problem. They can be categorized into three different types of approaches [9, 75]. Cost-sensitive solutions perform changes to the cost matrix in the learning phase. Algorithmic level solutions modify specific classifiers to reinforce learning on minority classes. Resampling solutions modify the training data by removing majority samples and/or generating artificial minority samples. The latter is independent from the context and can be used alongside any classifier. Since we are interested in the introduction of artificial data generation in AL, we will analyze the state-of-the-art on resampling techniques (specifically oversampling) in Section 4.3.

In this paper, we propose a novel AL framework to address two limitations commonly found in the literature: minimize human-computer interaction and reduce the class imbalance bias. This is done with the introduction of an additional component in the iterative AL procedure (the generator) that is used to generate artificial data to both balance and augment the training dataset. The introduction of this component is expected to reduce the number of iterations required until the classifier reaches a satisfactory performance.

This paper is organized as follows: Section 4.1 explains the problem and its context, Sections 4.2 and 4.3 describe the state of the art in AL and Oversampling techniques, Section 4.4 explains the proposed method, Section 4.5 covers the datasets, evaluation metrics, ML classifiers and experimental procedure, Section 4.6 presents the experiment's results and discussion and Section 4.7 presents the conclusions drawn from our findings.

4.2. Active Learning Approaches

As the amount of unlabeled data increases, the interest and practical usefulness of AL follows that trend [36]. AL is used as the general definition of frameworks aiming to train a learning system in multiple steps, where a set of new data points are chosen and added to the training dataset each time [144]. Typically, an AL framework is composed of the following elements [33, 34, 144]:

1. Unlabeled dataset. Consists of the original data source (or a sample thereof). It is used in combination with the chooser and the selection criterion to expand the training dataset in regions where the classification uncertainty is higher. Therefore, the unlabeled dataset is used for both producing the initial training dataset by selecting a set of instances for the supervisor to annotate (discussed in point 3) and calculating the uncertainty map to augment the training dataset.
2. Supervisor. A human annotator (or team of human annotators) to which the uncertainty map is presented to. The supervisor is responsible for annotating unlabeled instances to be added to the augmented dataset. In remote sensing, the supervisor is typically a photo-interpreter, as is the case in [37]. Some of the research also refers to the supervisor as the *oracle* [31, 32, 144, 147].

3. Initial training dataset. It is a small, labeled sample of the original data source used to initiate the first AL iteration. The size of the initial training sample normally varies between no instances at all and 10% of the unlabeled dataset [148].
4. Current and expanded training dataset. It is the concatenation of the initial training dataset and the datasets labeled by the supervisor in past iterations (discussed in point 2).
5. Chooser (classifier). Produces the class probabilities for each unlabeled instance.
6. Selection criterion. It quantifies the chooser's uncertainty level for each instance belonging to the unlabeled dataset. It is typically based on the class probabilities assigned by the chooser. In some situations, the chooser and the selection criterion are grouped together under the concept *acquisition function* [144] or *query function* [33]. Some of the literature refers to the selection criterion by using the concept *sampling scheme* [145].

Figure 4.1 schematizes the steps involved in a complete AL iteration. For a better context within the remote sensing domain, the prediction output can be identified as the LULC map. This framework starts by collecting unlabeled data from the original data source. It is used to generate a random initial training sample and is labeled by the supervisor. In practical applications, the supervisor is frequently a group of photo-interpreters [36]. The chooser is trained on the resulting dataset and is used to predict the class probabilities on the unlabeled dataset. The class probabilities are fed into a selection criterion to estimate the prediction's uncertainty, out of which the instances with the highest uncertainty will be selected. This calculation is motivated by the absence of labels in the uncertainty dataset. Therefore, it is impossible to estimate the prediction's accuracy in the unlabeled dataset in a real case scenario. The iteration is completed when the selected points are tagged by the supervisor and added to the training dataset (*i.e.*, the augmented dataset).

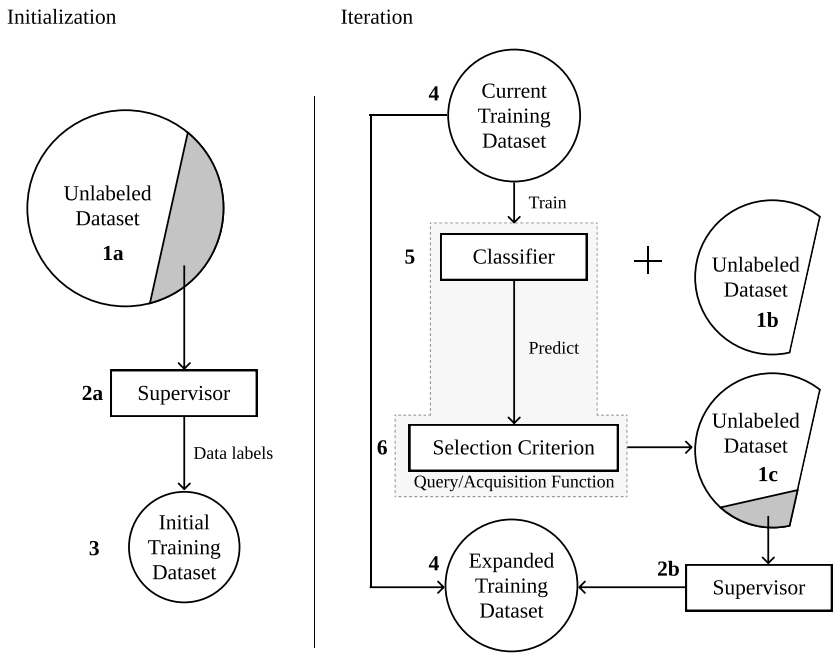


Figure 4.1.: Diagram depicting the typical AL framework.

A common challenge found in AL tasks is ensuring the consistency of AL over different initializations [36]. There are two factors involved in this phenomenon. On one hand, the implementation of the same method over different initializations may result in significantly different initial training samples, amounts to varying accuracy curves. On the other hand, the lack of a robust selection criterion and/or classifier may

also result in inconsistencies across AL experiments with different initializations. This phenomenon was observed and documented in a LULC classification context in [149].

The classification method plays a central role in the efficacy of AL. The classifier used should be able to generalise with a relatively small training dataset. Specifically, deep learning models are used in image classification due to its capability of producing high quality predictions. Although, to make such models generalizable the training set must be large enough, making its suitability for AL applications an open challenge [26, 150, 151]. Some studies in the Remote Sensing domain were developed to address this gap. In [26, 151], the authors propose a deep learning-based AL approach by training the same Convolutional Neural Network incrementally across iterations and smoothen the decision boundaries of the model using the Markov Random Field model and a Best-versus-Second Best labelling approach. This allows the introduction of additional data variability in the final training dataset. Another study [150] combined transfer learning, active classification and segmentation techniques for vehicle detection. By combining different techniques, they were able to produce a classification mechanism that performed well when the amount of training data is limited. However, the exploration of advanced deep learning classifiers in AL is still limited. In [61], the authors show that deep learning classifiers performs well on LULC classification, but are still not generalizable for different geographical regions or periods. Specifically, AL methods are still incapable of providing generalizable deep learning classifiers, which benefit from multiple advantages. The development of Convolutional Neural Networks with both 2 and 3-dimensional convolutions was explored in [152] and reported superior classification performance on benchmark datasets. However, a large amount of training data was used to produce the final classification map.

Selecting an efficient selection criterion is particularly important to find the instances closest to the decision border (*i.e.*, instances difficult to classify) [28]. Therefore, many AL related studies focus on the design of the query/acquisition function [33].

4.2.1. Non-informed selection criteria

Only one non-informed (*i.e.*, random) selection criterion was found in the literature. Random sampling selects unlabeled instances without considering any external information produced by the chooser. Since the method for selecting the unlabeled instances is random, this method disregards the usage of a chooser and is comparatively worse than any other selection criterion. However, random sampling is still a powerful baseline method [147].

4.2.2. Ensemble-based selection criteria

Ensemble disagreement is based on the class predictions of a set of classifiers. The disagreement between all the predictions for a given instance is a common measure for uncertainty, although computationally inefficient [144, 146]. It is calculated using the set of classifications over a single instance, given by the number of votes assigned to the most frequent class [28]. This method was implemented successfully for complex applications such as deep active learning [144].

Multiview [153] consists on the training of multiple independent classifiers using different views, which correspond to the selection of subsets of features or instances in the dataset. Therefore, it can be seen as a bootstrap aggregation (bagging) ensemble disagreement method. It is represented by the maximum disagreement score out of set of disagreements calculated for each view [28]. A lower value for this metric means a higher classification uncertainty. Multiview-based maximum disagreement has been successfully applied to hyper-spectral image classification in [154] and [56].

An adapted disagreement criterion for an ensemble of k-nearest neighbors has been proposed in [146]. This method employs a k-nearest neighbors classifier and computes an instance's classification uncertainty based on the neighbors' class frequency using the maximum disagreement metric over varying values for k.

As a result, this method is comparable to computing the dominant class' score over a weighted k-nearest neighbors classifier. This method was also used on a multimetric active learning framework [155].

Another relevant ensemble-based selection criterion is the binary random forest-based query model [33]. This method employs a one-versus-one ensemble method to demonstrate an efficient data selection method using the estimated probability of each binary random forest and determining the classification uncertainty based on the probabilities closest to 0.5 (*i.e.*, the least separable pair of classes are used to determine the uncertainty value). However, this study fails to compare the proposed method with other benchmark methods, such as random sampling.

4.2.3. Entropy-based criteria

A number of contributions have focused on entropy-based querying. The application of entropy is common among active deep learning applications [32], where the training of an ensemble of classifiers is often too expensive.

Entropy query-by-bagging (EQB), also defined as maximum entropy [145], is an ensemble approach of the entropy selection criterion, originally proposed in [156]. This strategy uses the set of predictions produced by the ensemble classifier to calculate those many entropy measurements. The estimated uncertainty measure for one instance is given by the maximum entropy within that set. EQB was observed to be an efficient selection criterion. Specifically, [28] applied EQB on hyper-spectral remote sensing imagery using Support Vector Machines (SVM) and Extreme Learning Machines (ELM) as choosers, achieving optimal results when combining EQB with ELM. Another study successfully implemented this method on an active deep learning application [145]. Another study improved over this method with a normalized EQB selection criterion [157].

4.2.4. Other relevant criteria

Margin Sampling is a SVM-specific criterion, based on the distance of a given point to the SVM's decision boundary [28]. This method is less popular than the remaining methods because it is limited to one type of chooser (SVMs). One extension of this method is the multiclass level uncertainty [28], calculated by subtracting the instance's distance to the decision boundaries of the two most probable classes [158].

The Mutual Information-based (MI) criterion selects the new training instances by maximizing the mutual information between the classifier and class labels in order to select instances from regions that are difficult to classify. Although this method is commonly used, it is frequently outperformed by the breaking ties selection criterion [51, 159].

The breaking ties (BT) selection criterion was originally introduced in [160]. It consists of the subtraction between the probabilities of the two most likely classes. Another related method is Modified Breaking Ties scheme (MBT), which aims at finding the instances containing the largest probabilities for the dominant class [51, 161]

Another type of selection criteria identified is the loss prediction method [31]. This method replaces the selection criterion with a predictor whose goal is to estimate the chooser's loss for a given prediction. This allows the new classifier to estimate the prediction loss on unlabeled instances and select the ones with the highest predicted loss.

Some of the literature fails to specify the strategy employed, although inferring it is generally intuitive. For example, [162] successfully used AL to address the imbalanced learning problem. They employed an ensemble of SVMs as the chooser, as well as an ensemble-based selection criterion. All of the research found related to this topic focused on the improvement of AL through modifications on the selection

criterion and classifiers used. None of these publications proposed significant variations to the original AL framework.

4.3. Artificial Data Generation Approaches

The generation of artificial data is a common approach to address imbalanced learning tasks [75], as well as improving the effectiveness of supervised learning tasks [17]. In recent years some sophisticated data generation approaches were developed. However, the scope of this work is to propose the integration of a generator within the AL framework. To do this, we will focus on heuristic data generation approaches, specifically, oversamplers.

Heuristic data resampling methods employ local and/or global information to generate new, relevant, non-duplicate instances. These methods are most commonly used to populate minority classes and balance the between-class distribution of a dataset. The Synthetic Minority Oversampling Technique (SMOTE) [22] is a popular heuristic oversampling algorithm, proposed in 2002. The simplicity and effectiveness of this method contributes to its prevailing popularity. It generates a new instance through a linear interpolation of a randomly selected minority-class instance and one of its randomly selected k-nearest neighbors. The implementation of SMOTE for LULC classification tasks has been found to improve the quality of the predictors used [117, 118]. Despite its popularity, its drawbacks motivated the development of other oversampling methods [23].

Geometric SMOTE (G-SMOTE) [23] introduces a modification of the SMOTE algorithm in the data generation mechanism to produce artificial instances with higher variability. Instead of generating artificial data as a linear combination of the parent instances, it is done within a deformed, truncated hyper-spheroid. G-SMOTE generates an artificial instance $\overrightarrow{z}$ within a hyper-spheroid, formed by selecting a minority instance $\overrightarrow{x}$ and one of its nearest neighbors $\overrightarrow{y}$, as shown in Figure 4.2. The truncation and deformation parameters define the shape of the spheroid's geometry. The method also modifies the selection strategy for the k-nearest neighbors, accepting the generation of artificial instances using instances from different classes, as shown in Figure 4.2d. The modification of both selection and generation mechanisms addresses the main drawbacks found in SMOTE, the generation of both noisy data (*i.e.,* generate minority class instances within majority class regions) and near-duplicate minority class instances [23]. G-SMOTE has shown superior performance when compared with other oversampling methods for LULC classification tasks, regardless of the classifier sed [2].

4.4. Proposed method

Within the literature identified, most of the work developed in the AL domain revolved around improving the quality of classification algorithms and/or selection criteria. Although these methods allow earlier convergence of the AL iterative process, the impact of these methods are only observed between iterations. Consequently, none of these contributions focused on the definition of decision borders within iterations. The method proposed in this paper modifies the AL framework by introducing an artificial data generation step within AL's iterative process. We define this component as the generator and is intended to be integrated into the AL framework as shown in Figure 4.3.

This modification, by using a new source of data to augment the training set, leverages the data annotation work conducted by the human operator. The artificial data that is generated between iterations reduces the amount of labeled data required to reach optimal performance and lower the amount of human labor

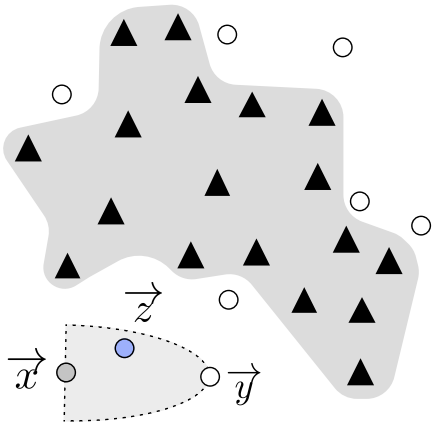


Figure 4.2.: Example of G-SMOTE's generation process. G-SMOTE randomly selects instance $\overrightarrow{x}$ and one of its nearest neighbors $\overrightarrow{y}$ to produce instance $\overrightarrow{z}$.

required to train a classifier to its optimal performance. This process lowers the annotation and overall training costs by translating some of the annotation cost into computational cost.

This method leverages the capability of artificial data to introduce more data variability into the augmented dataset and facilitate the chooser's training phase with a more consistent definition of the decision boundaries at each iteration. Therefore, any algorithm capable of producing artificial data, be it agnostic or specific to the domain, can be employed. The artificial data is only used to train the classifiers involved in the process and is discarded once the training phase is completed. The remaining steps in the AL framework remain unchanged. This method addresses the limitations found in the previous sections:

1. The convergence of classification performance should be anticipated with the clearer definition of the decision boundaries across iterations.
2. Annotation cost is expected to reduce as the need for labeled instances reduces along with the early convergence of the classification performance.
3. The class imbalance bias observed in typical classification tasks, as well as in AL is mitigated by balancing the class frequencies at each iteration.

Although the performance of this method is shown within a LULC classification context, the proposed framework is independent from the domain. The high dimensionality of remotely sensed imagery make its classification particularly challenging when the availability of labeled data is scarce and/or comes at a high cost, being subjected to the curse of dimensionality. Consequently, it is a relevant and appropriate domain to test this method.

4.5. Methodology

In this section we describe the datasets, evaluation metrics, oversampler, classifiers, software used and the procedure developed. We demonstrate the proposed method's efficiency over 7 datasets, sampled from publicly available, well-known remote sensing hyperspectral scenes frequently found in remote sensing literature. The datasets and sampling strategy are described in Subsection 4.5.1. On each of these datasets, we apply 3 different classifiers over the entire training set to estimate the optimal classification performance, the original AL framework as the baseline reference and the proposed method using G-SMOTE as a

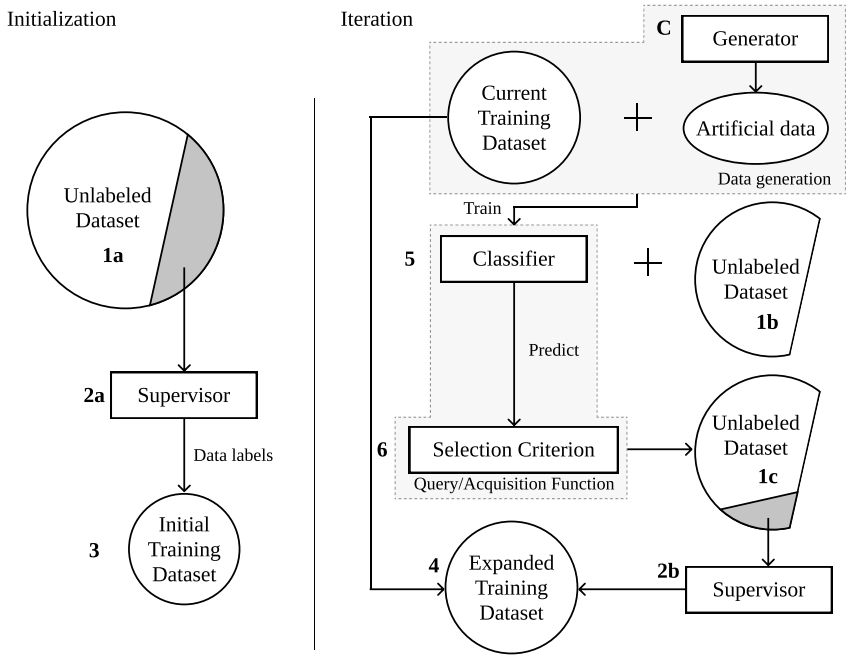


Figure 4.3.: Proposed AL framework. This paper's contribution comprises a change in the AL framework through the introduction of a data generation mechanism, represented as the generator (marked with C), which is used to add artificial instances to the training dataset.

generator, described in Subsection 4.5.2. The metrics used to estimate the performance of these algorithms are described in Subsection 4.5.3. Finally, the experimental procedure is described in Subsection 4.5.4.

Our methodology focuses on two objectives: (1) Comparison of optimal classification performance among active learners and traditional supervised learning and (2) Comparison of classification convergence efficiency among AL frameworks.

4.5.1. Datasets

The datasets used were extracted from publicly available repositories containing hyperspectral images and ground truth data. Additionally, all datasets were collected using the same sampling procedure. The description of the hyperspectral scenes used in this study is provided in Table 4.1. These scenes were chosen because of their popularity in the research community and their high baseline classification scores. Consequently, demonstrating an outperforming method in this context is particularly challenging and valuable.

The Indian Pines scene [126] is composed of agriculture fields in approximately two thirds of its coverage, low density buildup areas and natural perennial vegetation in the remainder of its area (see Figure 4.4a). The Pavia Centre and University scenes are hyperspectral, high-resolution images containing ground truth data composed of urban-related coverage (see Figures 4.4b and 4.4c). The Salinas and Salinas A scenes contain at-sensor radiance data. As subset of Salinas, the Salinas A scene contains contains the vegetables fields present in Salinas and the latter is also composed of bare soils and vineyard fields (see Figures 4.4d and 4.4e). The Botswana scene contains ground truth data composed of seasonal swamps, occasional swamps, and drier woodlands located in the distal portion of the Delta (see Figure 4.4f). The Kennedy

Dataset	Sensor	Location	Dimension	Bands	Res. (m)	Classes
Botswana	Hyperion	Okavango Delta	1476 x 256	145	30	14
Salinas A	AVIRIS	California, USA	86 x 83	224	3.7	6
Kennedy Space Center	AVIRIS	Florida, USA	512 x 614	176	18	16
Indian Pines	AVIRIS	NW Indiana, USA	145 x 145	220	20	16
Salinas	AVIRIS	California, USA	512 x 217	224	3.7	16
Pavia University	ROSIS	Pavia, Italy	610 x 610	103	1.3	9
Pavia Centre	ROSIS	Pavia, Italy	1096 x 1096	102	1.3	9

Table 4.1.: Description of the hyperspectral scenes used in this experiment. The column "Res. (m)" refers to the resolution of the sensors (in meters) that captured each of the scenes.

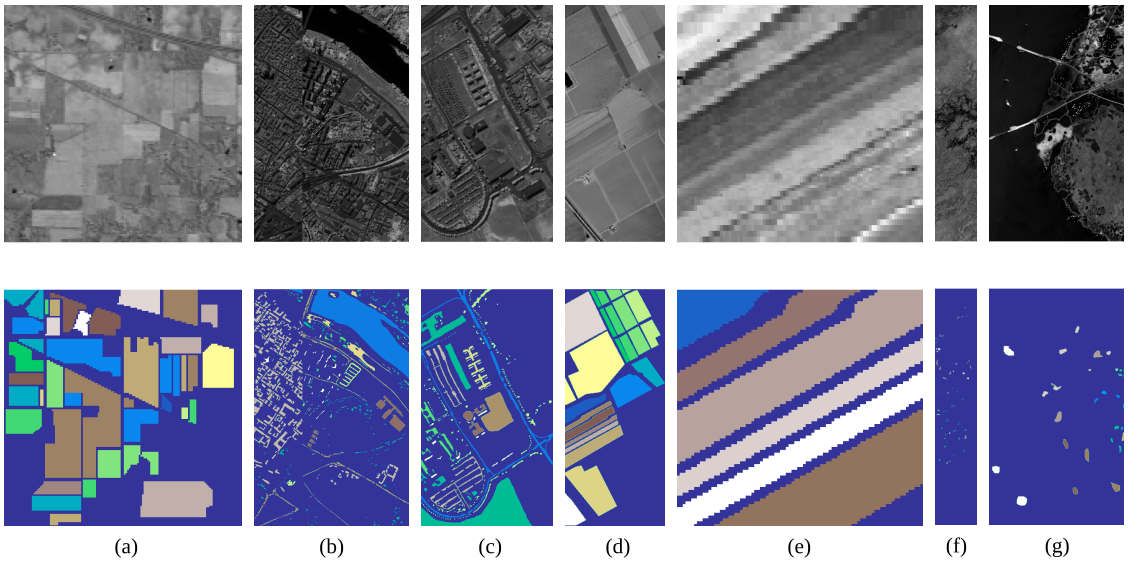


Figure 4.4.: Gray scale visualization of a band (top row) and ground truth (bottom row) of each scene used in this study. (a) Indian Pines, (b) Pavia Centre, (c) Pavia University, (d) Salinas, (e) Salinas A, (f) Botswana, (g) Kennedy Space Center

Space Center scene contains a ground truth composed of both vegetation and urban-related coverage (see Figure 4.4g).

The sampling strategy is similar to all datasets. The pixels without a ground truth label are first discarded. All the classes with cardinality lower than 150 are also discarded. This is done to maintain feasible Imbalance Ratios (IR) across datasets (where $IR = \frac{count(C_{maj})}{count(C_{min})}$). Finally, a stratified sample of 1500 instances are selected for the experiment. The resulting datasets are described in Table 4.2. The motivation for this strategy is three fold: (1) reduce the datasets to a manageable size and allow the experimental procedure to be completed within a feasible time frame, (2) ensure the relative class frequencies in the scenes are preserved and (3) ensure equivalent analyses across datasets and AL frameworks. In this context, a fixed number of instances per dataset is especially important to standardize the AL-related performance metrics.

4.5.2. Machine Learning Algorithms

Dataset	Features	Instances	Min. Instances	Maj. Instances	IR	Classes
Botswana	145	1500	89	154	1.73	12
Salinas A	224	1500	109	428	3.93	6
Kennedy Space Center	176	1500	47	272	5.79	12
Indian Pines	220	1500	31	366	11.81	12
Salinas	224	1500	25	312	12.48	16
Pavia University	103	1500	33	654	19.82	9
Pavia Centre	102	1500	27	668	24.74	9

Table 4.2.: Description of the datasets collected from each corresponding scene. The sampling strategy is similar to all scenes.

We use two different types of ML algorithms. A data generation algorithm, used to form the generator, and classification algorithms, used to calculate the classification uncertainties in the unlabeled dataset and predict the class labels in the validation and test sets.

Although any method capable of generating artificial data can be used as a generator, the one used in this experiment is an oversampler, originally developed to deal with imbalanced learning problems. Specifically, we chose G-SMOTE, a state-of-the-art oversampler.

Three classification algorithms are used. We use different types of classifiers to test the framework's performance under varying situations: neighbors-based, linear and ensemble models. The neighbors-based classifier chosen was K-nearest neighbors (KNN) [128], a logistic regression (LR) [127] is used as the linear model and a random forest classifier (RFC) [163] was used as the ensemble model.

The acquisition function is completed by testing three different selection criteria. Random selection is used as a baseline selection criterion, whereas entropy and breaking ties are used due to their popularity and independence of the classifier used.

4.5.3. Evaluation Metrics

Since the datasets used in this experiment have an imbalanced distribution of class frequencies, metrics such as the *Overall Accuracy* (OA) and *Kappa coefficient* are insufficient to accurately depict classification performance [130, 131]. Instead, metrics such as Producer's Accuracy (or *Recall*) and User's Accuracy (or *Precision*) can be used. Since they consist of ratios based on True/False Positives (TP and FP) and Negatives (TN and FN), they provide per class information regarding the classifier's classification performance. However, in this experiment, the meaning and number of classes available in each dataset varies, making these metrics difficult to synthesize.

The performance metric *Geometric mean* (G-mean) and *F-score* are less sensitive to the data imbalance bias [132, 164]. Therefore, we employ both of these scorers. G-mean consists of the geometric mean of $Specificity = \frac{TN}{TN+FP}$ and $Sensitivity = \frac{TP}{TP+FN}$ (also known as *Recall*) [164]. Both metrics are calculated in a multiclass context considering a one-versus-all approach. For multiclass problems, the *G-mean* scorer is calculated as its average per class values:

$$G\text{-}mean = \sqrt{\overline{Sensitivity}_i \times \overline{Specificity}_i}$$

The F-score performance metric is the harmonic mean of *Precision* and *Recall*. The two metrics are also calculated considering a one-versus-all approach. The *F-score* for the multi-class case can be calculated using its average per class values [133]:

$$F\text{-}score = 2\frac{\overline{Precision} \times \overline{Recall}}{\overline{Precision} + \overline{Recall}}$$

The comparison of classification convergence across AL frameworks and selection criteria is done using 2 AL-specific performance metrics. Particularly, we follow the recommendations found in [36]. Each AL configuration is evaluated using the *Area Under the Learning Curve* (AULC) performance metric. It is the sum of the classification performance values of all iterations. To facilitate the analysis of the results, we fix the range of this metric between $[0, 1]$ by dividing it with the total amount of iterations (*i.e.*, the maximum performance area).

The *Data Utilization Rate* (DUR) [165] metric consists of the ratio between the number of instances required to reach a given G-mean score threshold by an AL strategy and an equivalent baseline strategy. For easier interpretability, we simplify this metric by using the percentage of training data used by an AL strategy to reach the performance threshold, instead of presenting these values as a ratio of the baseline strategy. The DUR metric is measured at 9 different performance levels, between 0.6 and 0.95 G-mean scores at a 0.05 step.

4.5.4. Experimental Procedure

A common practice in methodological evaluations is the implementation of an offline experiment [166]. It consists of using an existing set of labeled data as a proxy for the population of unlabeled instances. Because the dataset is already fully labeled, the supervisor's typical annotation process involved in each iteration is done at zero cost. Each AL and classifier configuration is tested using a stratified 5-fold cross validation testing scheme. For each round, the larger partition is split in a stratified fashion to form a training and validation set (containing 20% of the original partition). The validation set is used to evaluate the convergence efficiency of active learners; the chooser's classification performance metrics and amount of data points used at each iteration are used to compute the AULC and DUR. Additionally, within the AL iterative process, the classifier with optimal performance on the validation set is evaluated using the test set. In order to further reduce possible initialization biases, this procedure is repeated 3 times with different initialization seeds and the results of all runs are averaged (*i.e.*, each configuration is trained and evaluated 15 times). Finally, the maximum performance lines are calculated using the same approach. In those cases, the validation set is not used. The experimental procedure is depicted in Figure 4.5.

To make the AL-specific metrics comparable among active learners, the configurations of the different frameworks must be similar. For each dataset, the number of instances is constant to facilitate the analysis of the same metrics.

In most practical AL applications it is assumed that the number of instances in the initial training sample is too small to perform hyperparameter tuning. Consequently, in order to ensure realistic results, our experimental procedure does not include hyperparameter optimization. The predefined hyperparameters are shown in Table 4.3. They were set up based on general recommendations and default settings for the classifiers and generators used.

The AL iterative process is set up with a randomly selected initial training sample with 15 initial samples. At each iteration, 15 additional samples are added to the training set. This process is stopped after

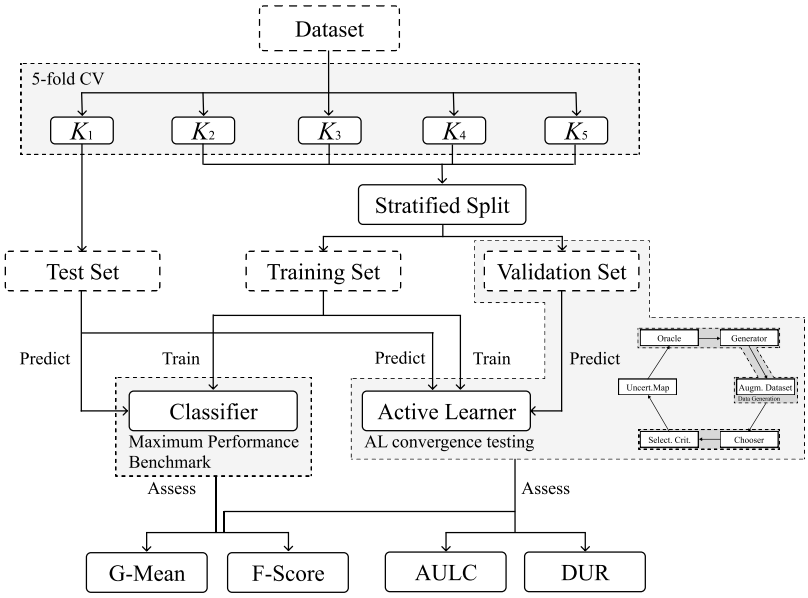


Figure 4.5.: Experimental procedure. The datasets extracted from hyperspectral scenes are split in 5 folds. 1 of those (*e.g.*, K_1) is used to test the optimal performance of AL algorithms and the classification without AL. The training set is used to iterate AL algorithms and train classifiers. The validation set is used to test the convergence of AL algorithms. The results are averaged over the 5 folds across each of the 3 different initializations of this procedure.

49 iterations, once 50% of the entire dataset (*i.e.*, 78% of the training set) is added to the augmented dataset.

4.5.5. Software Implementation

The experiment was implemented using the Python programming language, along with the Python libraries Scikit-Learn [86], Imbalanced-Learn [134], Geometric-SMOTE, Cluster-Over-Sampling and Research-Learn libraries. All functions, algorithms, experiments and results are provided in the GitHub repository of the project.

4.6. Results & Discussion

The evaluation of the different AL frameworks in a multiple dataset context should not rely uniquely on the mean of the performance metrics across datasets. [135] recommends the use of mean ranking scores, since the performance levels of the different frameworks varies according to the data it is being used on. Consequently, evaluating these performance metrics solely based on their mean values might lead to inaccurate analyses. Accordingly, the results of this experiment are analysed using both the mean ranking and absolute scores for each model. The rank values are assigned based on the mean scores resulting from three different initializations of 5-fold cross validation for each classifier and active learner. The goal of this analysis is to understand whether the proposed framework (AL with the integration of an artificial data generator) is capable of using less data from the original dataset while simultaneously achieving better classification results than the standard AL framework, *i.e.*, guarantee a faster classification convergence.

Classifier	Hyperparameters	Values
LR	maximum iterations	10000
	solver	sag
	penalty	None
KNN	# neighbors	5
	weights	uniform
	metric	euclidean
RF	maximum tree depth	None
	# estimators	100
	criterion	gini
Generator		
G-SMOTE	# neighbors	5
	deformation factor	0.5
	truncation factor	0.5

Table 4.3.: Hyper-parameter definition for the classifiers and generator used in the experiment.

Classifier	Evaluation Metric	Standard	Proposed
KNN	F-score	2.00 ± 0.0	**1.00 ± 0.0**
KNN	G-mean	2.00 ± 0.0	**1.00 ± 0.0**
LR	F-score	1.71 ± 0.45	**1.29 ± 0.45**
LR	G-mean	2.00 ± 0.0	**1.00 ± 0.0**
RF	F-score	1.86 ± 0.35	**1.14 ± 0.35**
RF	G-mean	2.00 ± 0.0	**1.00 ± 0.0**

Table 4.4.: Mean rankings of the AULC metric over the different datasets (7), folds (5) and runs (3) used in the experiment. This means that the use of G-SMOTE almost always improves the results of the original framework.

4.6.1. Results

Table 4.4 shows the average rankings and standard deviations across datasets of the AULC scores for each active learner.

The mean AULC absolute scores are provided in Table 4.5. These values are computed as the mean of the sum of the scores of a specific performance metric over all iterations (for an AL configuration). In other words, these values correspond to the average AULC over 7 $datasets \times 5$ $folds \times 3$ $initializations$.

The average DURs are shown in Table 4.6. They were calculated for various G-mean scores thresholds, varying at a step of 5% between 60% and 95%. Each row shows the percentage of training data required by the different AL configurations to reach that specific G-mean score.

G-mean Score	Classifier	Standard	Proposed
0.60	KNN	4.0%	**2.1%**
0.60	LR	2.2%	**2.1%**
0.60	RF	2.2%	**2.1%**
0.65	KNN	5.6%	**2.8%**

G-mean Score	Classifier	Standard	Proposed
0.65	LR	3.0%	**2.7%**
0.65	RF	3.1%	**2.6%**
0.70	KNN	7.9%	**4.1%**
0.70	LR	4.2%	**4.1%**
0.70	RF	4.5%	**3.6%**
0.75	KNN	13.5%	**7.1%**
0.75	LR	7.2%	**6.6%**
0.75	RF	6.6%	**5.4%**
0.80	KNN	24.4%	**16.9%**
0.80	LR	13.1%	**11.7%**
0.80	RF	11.6%	**9.2%**
0.85	KNN	29.8%	**23.6%**
0.85	LR	19.8%	**18.8%**
0.85	RF	23.1%	**17.3%**
0.90	KNN	41.0%	**36.1%**
0.90	LR	28.1%	**24.8%**
0.90	RF	37.1%	**30.3%**
0.95	KNN	71.3%	**69.1%**
0.95	LR	45.8%	**40.2%**
0.95	RF	64.6%	**62.2%**

Table 4.6.: Mean data utilization of AL algorithms, as a percentage of the training set.

The DUR of the proposed method relative to the baseline method is shown in Figure 4.6. A DUR below 1 means that the proposed framework requires less data to reach the same performance threshold (as a percentage, relative to the amount of data required by the baseline framework). For instance, in the upper left graphic we can see that the proposed framework achieves 90% classification using F-score while using 91% of the amount of data used by the traditional AL framework, in other words 9% less data.

The averaged optimal classification scores are shown in Table 4.7. The maximum performance (MP) classification scores are shown as a benchmark and represent the performance of the corresponding classifier using the entire training set.

4.6.2. Statistical Analysis

The methods used to test the experiment's results must be appropriate for a multi-dataset context. Therefore the statistical analysis is performed using the Wilcoxon signed-rank test [137] as a post-hoc analysis. The variable used for this test is the data utilization rate based on the G-mean performance metric, considering the various performance thresholds from Table 4.6.

The Wilcoxon signed-rank test results are shown in Table 4.8. We test as null hypothesis that the performance of the proposed framework is the same as the original AL framework. The null hypothesis was rejected in all datasets.

Classifier	Evaluation Metric	Standard	Proposed
KNN	F-score	0.762 ± 0.131	**0.794 ± 0.123**
KNN	G-mean	0.864 ± 0.079	**0.886 ± 0.073**
LR	F-score	0.839 ± 0.119	**0.843 ± 0.116**
LR	G-mean	0.907 ± 0.074	**0.911 ± 0.071**
RF	F-score	0.810 ± 0.109	**0.819 ± 0.1**
RF	G-mean	0.890 ± 0.068	**0.901 ± 0.059**

Table 4.5.: Average AULC of each AL configuration tested. Each AULC score is calculated using the G-mean scores of each iteration in the validation set. By the end of the iterative process, each AL configuration used a total of 750 instances of the 960 instances that compose the training set.

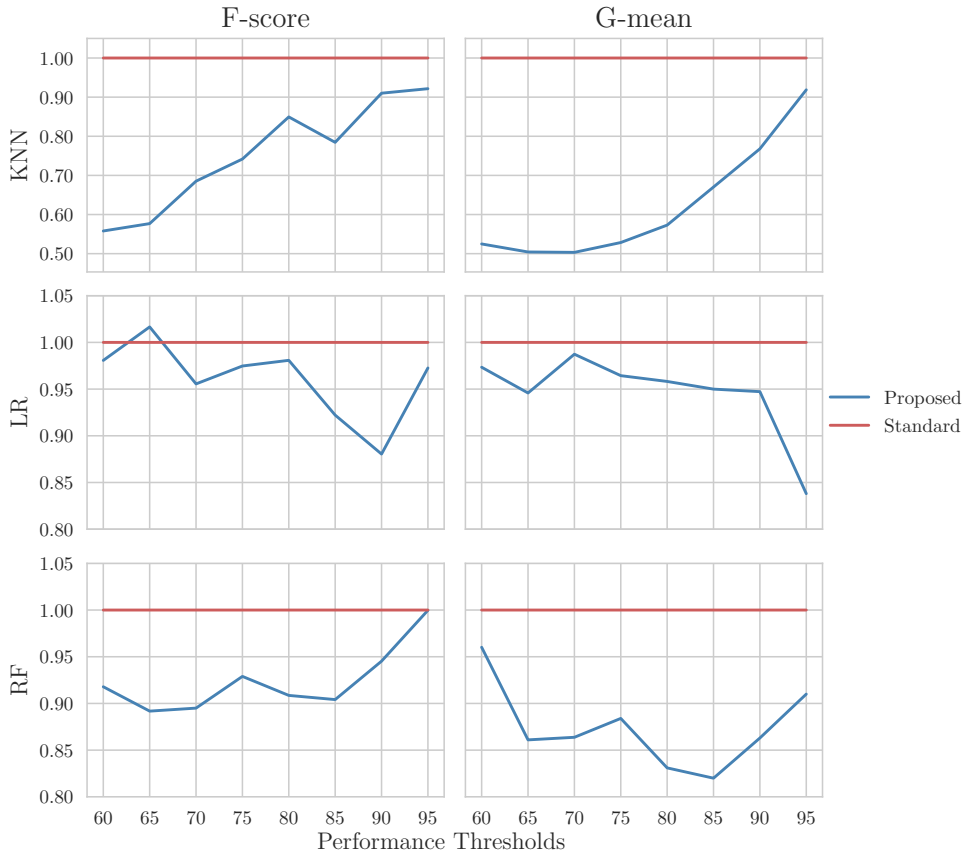


Figure 4.6.: Mean data utilization rates. The y-axis shows the percentage of data (relative to the baseline AL framework) required to reach the different performance thresholds.

Classifier	Evaluation Metric	MP	Standard	Proposed
KNN	F-score	0.838 ± 0.106	0.835 ± 0.115	**0.843 ± 0.105**
KNN	G-mean	0.907 ± 0.063	0.904 ± 0.069	**0.912 ± 0.061**
LR	F-score	**0.890 ± 0.084**	0.883 ± 0.096	0.887 ± 0.097
LR	G-mean	0.935 ± 0.052	0.931 ± 0.059	**0.938 ± 0.055**
RF	F-score	0.859 ± 0.083	0.866 ± 0.081	**0.869 ± 0.08**
RF	G-mean	0.918 ± 0.051	0.921 ± 0.051	**0.930 ± 0.043**

Table 4.7.: Optimal classification scores. The Maximum Performance (MP) classification scores are calculated using classifiers trained using the entire training set.

Dataset	p-value	Significance
Botswana	3.8e-03	True
Indian Pines	2.3e-04	True
Kennedy Space Center	1.3e-04	True
Pavia Centre	4.3e-03	True
Pavia University	4.6e-05	True
Salinas	4.6e-05	True
Salinas A	3.0e-03	True

Table 4.8.: Adjusted p-values using the Wilcoxon signed-rank method. Bold values are statistically significant at a level of $\alpha = 0.05$. The null hypothesis is that the performance of the proposed framework is similar to that of the original framework.

4.6.3. Discussion

This paper expands the AL framework by adding an artificial data generator into its iterative process. This modification is done to accelerate the classification convergence of the standard AL procedure, which is reflected in the reduction of the amount of data necessary to reach better classification results.

The convergence efficiency of the proposed method is always higher than the baseline AL framework, with the exception of one comparison, as shown in Table 4.4 and Figure 4.6. This means the proposed AL framework using data generation was able to outperform the baseline AL in nearly all scenarios.

The mean AULC scores in Table 4.5 show a significant improvement in the performance of AL when a generator is used. The mean performance of the proposed framework is always better than the baseline framework. This improvement is explained by:

1. Earlier convergence of AL, *i.e.*, requiring less data to achieve comparable performance levels. This effect is shown in Table 4.6, where we found that the proposed framework always uses less data for similar performance levels, regardless of the classifier used.
2. Higher optimal classification performance, *i.e.*, reaching higher performance levels overall. This effect is shown in Table 4.7, where we found that using a generator in AL led to a better classification performance and was capable of outperforming the MP threshold.

Our results show statistical significance in every dataset. The proposed framework had a superior performance with statistical significance on each dataset at a level of $\alpha = 0.05$. This indicates that regardless of the context under which an AL algorithm is used, the proposed framework reduces the amount of data necessary in the AL's iterative process.

This paper introduces the concept of applying data a generation algorithm in the AL framework. This was done with the implementation of a recent state of the art generalization of a popular data generation algorithm. Although, since this algorithm is based on heuristics, future work should focus on improving these results through the design of new data generation mechanisms, at the cost of additional computational power. In addition, we also noticed significant standard errors in our experimental results (see Subsection 4.6.1). This indicates that AL procedures seem to be particularly sensitive to the initialization method, which is still a limitation of AL, regardless of the framework and configurations used. This is consistent with the findings in [36], which future work should attempt to address. Although using a generator marginally reduced this standard error, it is not sufficient to address this specific limitation.

4.7. Conclusion

The aim of this experiment was to test the effectiveness of a new AL framework that introduces artificial data generation in its iterative process. The experiment was designed to test the proposed method under particularly challenging conditions, where the maximum performance line is naturally high in most datasets. The element that constitute the Generator component was set up in a plug-and-play scheme, without significant tuning of the G-SMOTE oversampler. Using a generator in AL improved the original AL framework in all scenarios. These results could be further improved through the modification and more intense tuning of the data generation strategy. In our experiment, artificial data was generated only to match each non-majority class frequency with the majority class frequency, strictly balancing the class distribution. Generating a larger amount of data for all classes can further improve these results.

The high performance scores for the baseline AL framework made the achievement of significant improvements over the traditional AL framework under these conditions particularly meaningful. The advantage of the proposed AL framework is shown in Table 4.6. In most of the presented scenarios there is a substantial reduction of data necessary to reach a given performance threshold.

The results from this experiment show that using a data generator in the AL framework will improve the convergence of the method. This framework successfully anticipate the predictor's optimal performance, as shown in Tables 4.4, 4.5 and 4.6. Therefore, in a real application, the annotation cost would have been reduced since less iterations and labeled instances are necessary to reach near optimal classification performance.

5. Improving Active Learning Performance Through the Use of Data Augmentation

Published as Joao Fonseca, Fernando Bacao, in International Journal of Intelligent Systems, 2023

Active Learning (AL) is a well-known technique to optimize data usage in training, through the interactive selection of unlabeled observations, out of a large pool of unlabeled data, to be labeled by a supervisor. Its focus is to find the unlabeled observations that, once labeled, will maximize the informativeness of the training dataset, therefore reducing data related costs. The literature describes several methods to improve the effectiveness of this process. Nonetheless, there is a paucity of research developed around the application of artificial data sources in AL, especially outside image classification or NLP. This paper proposes a new AL framework, which relies on the effective use of artificial data. It may be used with any classifier, generation mechanism and data type, and can be integrated with multiple other state-of-the-art AL contributions. This combination is expected to increase the ML classifier's performance and reduce both the supervisor's involvement and the amount of required labeled data, at the expense of a marginal increase in computational time. The proposed method introduces a hyperparameter optimization component to improve the generation of artificial instances during the AL process, as well as an uncertainty-based data generation mechanism. We compare the proposed method to the standard framework and an oversampling-based active learning method for more informed data generation in an AL context. The models' performance was tested using four different classifiers, two AL-specific performance metrics and three classification performance metrics over 15 different datasets. We demonstrate that the proposed framework, using data augmentation, significantly improves the performance of AL, both in terms of classification performance and data selection efficiency.[1]

Keywords: Active Learning; Data Augmentation; Oversampling

5.1. Introduction

The importance of training robust ML models with minimal data requirements is substantially increasing [34, 44, 167]. Although the growing amount of valuable data sources and formats being developed and explored is affecting various domains [168], this data is often unlabeled. Only a tiny amount of the data being produced and stored can be helpful in supervised learning tasks. In addition, it is often difficult and expensive to label data for specific Machine Learning (ML) projects, especially when data-intensive ML techniques are involved (*e.g.,* Deep Learning classifiers) [167]. In this scenario, labeling the full dataset becomes impractical, time-consuming and expensive. Two different ML techniques attempt to address this problem: Semi-Supervised Learning (SSL) and Active Learning (AL). Even though they address the

[1] All the code and preprocessed data developed for this study is available at https://github.com/joaopfonseca/publications/.

same problem, the two follow different approaches. SSL focuses on observations with the most certain predictions, whereas AL focuses on observations with the least certain predictions [8].

SSL attempts to use a small, predefined set of labeled and unlabeled data to produce a classifier with superior performance. This method uses the unlabeled observations to help define the classifier's decision boundaries [169]. Simultaneously, the amount of labeled data required to reach a given performance threshold is also reduced. It is a particular case of ML because it falls between the supervised and unsupervised learning perspectives. AL, instead of optimizing the informativeness of an existing training set, expands the dataset to include the most informative and/or representative observations [170]. It is an iterative process where a supervised model is trained and simultaneously identifies the most informative unlabeled observations to increase the performance of that classifier. The combination of SSL with AL has been explored in the past, achieving state-of-the-art results [171].

Several studies have pointed out the limitations of AL within an Imbalanced Learning context [172, 173]. With imbalanced data, AL approaches frequently have low performance, high computational time, or data annotation costs. Studies addressing this issue tend to adopt classifier-level modifications, such as the Weighted Extreme Learning Machine [172, 174, 175]. However, classifier or query function-level modifications (See Section 5.2.1) have limited applicability since a universally good AL strategy has not yet been found [170]. Other methods address imbalanced learning by weighing the observations as the function of the observation's class imbalance ratio [92]. Alternatively, other techniques reduce the imbalanced learning bias by combining Informative and Representative-based query approaches (see Section 5.2.1) [176]. Another approach to deal with imbalanced data and data scarcity, in general, is generating synthetic data [133]. This approach has the advantage of being classifier-agnostic, it potentially reduces the imbalanced learning bias, and also works as a regularization method in data-scarce environments, such as AL implementations [177]. However, most recent studies improve the AL performance by modifying the design/choice of the classifier and query functions used.

Recently, synthetic data generation techniques gathered attention among ML researchers for its effectiveness over a wide range of applications: regularization, oversampling, semi-supervised learning, self-supervised learning, etc. Data augmentation generates synthetic observations to complement naturally occurring observations. It aims to reinforce the definition of a ML classifier's decision boundary during the learning phase and improve the generalization of the algorithm. These techniques have the advantage of being a data level technique (despite the existence of augmentation methods applied internally in the ML classifier). Therefore, they can be implemented in a way that will not affect the choice of classifier and does not exclude the usage of other regularization approaches. In an AL context, the generation of synthetic data becomes particularly appealing, especially with randomized and statistical-based approaches; it ensures better model performance with reduced involvement of a human agent, at the expense of a marginal increase in computational power. In addition, synthetic data is expected to reduce the amount labeled data required for a good AL implementation.

Figure 5.1 illustrates the difference across AL iterations between the standard AL approach and the proposed method. Synthetic data allowed for a quicker expansion of the labeled input area at an early stage of the process, with better defined decision boundaries and near-covergence of the ML classifier's performance at the third iteration. Data augmentation influences the choice of unlabeled observations for labeling into regions where synthetic data is not being able to represent the unlabeled data pool.

5.1.1. Motivation and contributions

The usage of data augmentation in AL is not new. The literature found on the topic (see Section 5.2.3) focuses on either image classification or Natural Language Processing and uses Deep Learning-based data augmentation to improve the performance of neural network architectures in AL. These methods, although showing promising results, represent a limited perspective of the potential of data augmentation in a real-world setting:

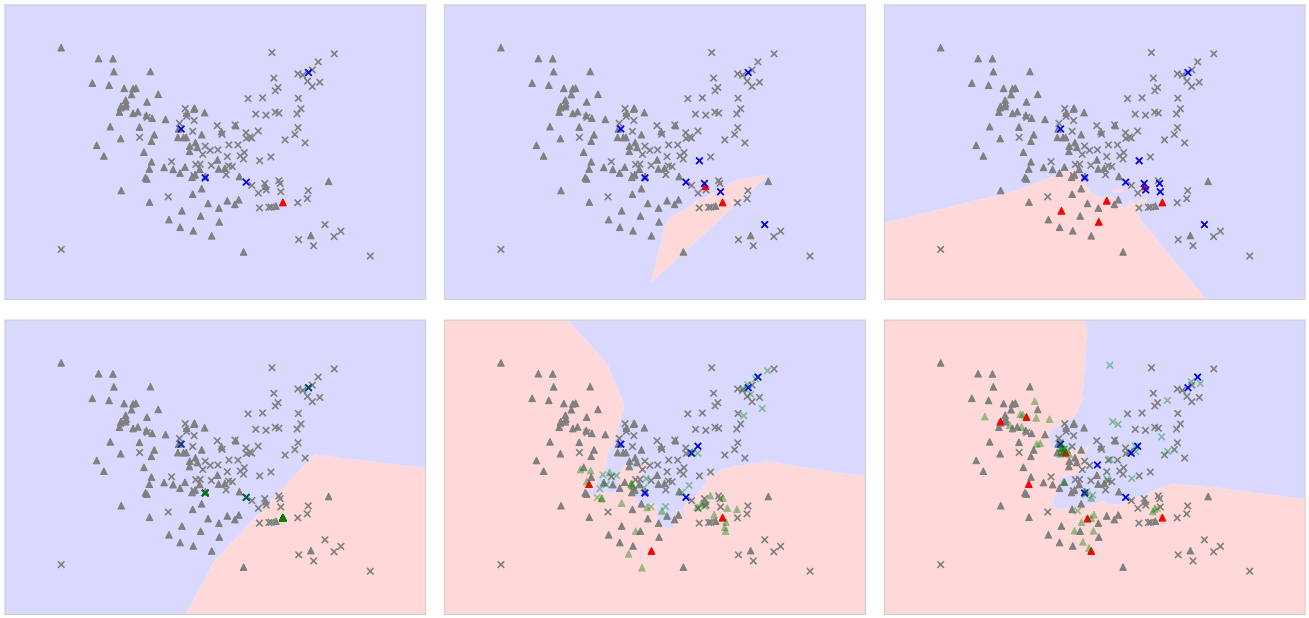


Figure 5.1.: Illustration of the different acquisition processes in AL using a K-Nearest Neighbors classifier and Shannon's entropy as the uncertainty estimation function, with five observations being collected and labeled per iteration. The top row shows the behavior of a standard AL implementation, while the bottom row shows the behavior of the proposed method. Column (a), (b) and (c) show the decision boundaries at iterations 1 (after the collection of five random initial training observations), 2 (with 10 labeled observations) and 3 (with 15 labeled observations), respectively. The initial labeled dataset for both approaches is the same. The two classes are distinguished with $\triangle$ and $\times$, and are colored as red and blue (respectively) if they are labeled. The transparent green observations are synthetic observations (bottow row only).

1. Using Deep Learning in an iterative setting requires access to significant computational power.
2. These models tend to use sophisticated data augmentation methods, whose implementation may not be accessible to non-sophisticated users.
3. They require a significant amount of processing time per iteration and are inappropriate for settings with limited time budgets.
4. The studies found on the topic are specific to the domain, classifier, and data augmentation method. In addition, all of the related methods found (except one) focus on either image or natural language processing classification problems.

Consequently, the direct effect of data augmentation is unclear: these studies implement different neural network-based techniques for different classification problems, whose performance may be attributed to various elements within the AL framework.

In this study, we explore the effect of data augmentation in AL in a context-agnostic setting, along with two different data augmentation policies: oversampling (where the amount of data generated for each class equals the amount of data belonging to the majority class) and non-constant data augmentation policies (where the amount of data generated exceeds the amount of data belonging to the majority class in varying quantities) between iterations. We start by conceptualizing the AL framework and each of its elements, as well as the modifications involved to implement data augmentation in the AL iterative process. We argue

that simple, non-domain specific data augmentation heuristics are sufficient to improve the performance of AL implementations, without the need to resort to deep learning-based data augmentation algorithms. These contributions can be summarized as follows:

1. We propose a flexible AL framework with pipelined data augmentation for tabular data that may be adapted for any domain or data type. This implementation is directed towards use cases with limited computational power and/or processing time.
2. We use a geometric-based data augmentation method for non-network based classifiers and adapt it to leverage information from the AL process. To the best of our knowledge, most existing methods use domain/classifier-specific augmentations or the Mixup approach, which is incompatible with most classifiers that are not neural network-based.
3. We provide empirical evidence that the integration of varying data augmentation policies between iterations in the AL framework not only further reduces the amount of labeled data required, but is also a viable training strategy for fully supervised learning settings.

When compared to the standard AL framework, the proposed framework contains two additional components: the Generator and the Hyperparameter Optimizer. We implement a modified version of the Geometric Synthetic Minority Oversampling Technique (G-SMOTE) [23] as a data augmentation method with an optimized generation policy (explained in Section 1.1). We also propose a hyperparameter optimization module, which is used to find the best data augmentation policy at each iteration. We test the effectiveness of the proposed method in 15 datasets of different domains. We implement three AL frameworks (standard, oversampling and varying data augmentation) using four different classifiers, three different performance metrics and calculate two AL-specific performance metrics.

The remainder of this manuscript is structured as follows: Section 5.2 introduces relevant topics discussed in the paper and describes the related work. Section 5.3 elucidates the proposed method. Section 5.4 details the methodology of the study's experiment. Section 5.5 presents the results obtained from the experiment, as well as a discussion of these results. Section 5.6 presents the conclusions drawn from this study.

5.2. Background

In this section we describe the AL problem, data augmentation techniques, and review the literature that combines AL with data augmentation. Table 5.1 describes the notations used throughout the rest of this study.

Table 5.1.: Description of all notations and symbols used throughout the manuscript.

Symbol	Meaning
f_c	ML classifier.
x_i	Observation at index i.
$f_{acq}(x_i; f_c)$	Acquisition function.
$f_{aug}(x_i; \tau)$	Augmentation function.
τ	Augmentation policy.
$\mathcal{D}$	Data pool. Contains both labeled and unlabeled data.
$\mathcal{D}^t_{lab}$	Labeled data set at iteration t.
$\mathcal{D}^t_{pool}$	Unlabeled data pool at iteration t.
$\mathcal{D}^t_{new}$	Set of observations from $\mathcal{D}^t_{pool}$to be labeled and added to $\mathcal{D}^{t+1}_{lab}$.

Continued on next page

Table 5.1.: Description of all notations and symbols used throughout the manuscript.

Symbol	Meaning
T	Iteration budget.
n	Annotation budget per iteration.

5.2.1. Active Learning

This paper focuses on pool-based AL methods as defined in [178]. The goal of AL models is to maximize the performance of a classifier, f_c, while annotating as least observations, x_i, as possible. They use a data pool, $\mathcal{D}$, where $\mathcal{D} = \mathcal{D}_{lab} \cup \mathcal{D}_{pool}$ and $|\mathcal{D}_{pool}| \gg |\mathcal{D}_{lab}|$. $\mathcal{D}_{pool}$ and $\mathcal{D}_{lab}$ refer to the sets of unlabeled and labeled data, respectively. Having a budget of T iterations (where $t \in \{1, 2, \ldots, T\}$) and n annotations per iteration, at iteration t, f_c is trained using $\mathcal{D}^t_{lab}$ to produce, for each $x_i \in \mathcal{D}^t_{pool}$, an uncertainty score using an acquisition function $f_{acq}(x_i; f_c)$. These uncertainty scores are used to annotate the n observations with highest uncertainty from $\mathcal{D}^t_{pool}$ to form $\mathcal{D}^t_{new}$. The iteration ends with the update of $\mathcal{D}^{t+1}_{lab} = \mathcal{D}^t_{lab} \cup \mathcal{D}^t_{new}$ and $\mathcal{D}^{t+1}_{pool} = \mathcal{D}^t_{pool} \setminus \mathcal{D}^t_{new}$ [33, 34]. This process is shown in Figure 5.2. Before the start of the iterative process, assuming $\mathcal{D}^{t=0}_{lab} = \emptyset$, the data used to populate $\mathcal{D}^{t=1}_{lab}$ is typically collected randomly from $\mathcal{D} = \mathcal{D}^{t=0}_{pool}$ and is labeled by a supervisor [31, 32, 76].

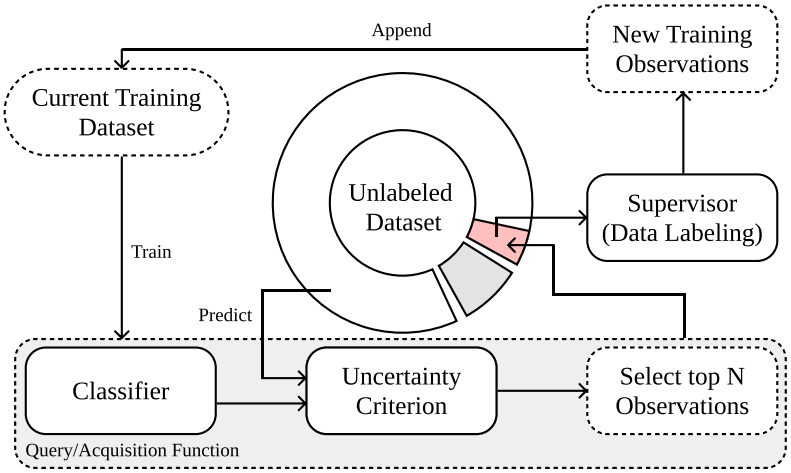


Figure 5.2.: Diagram depicting a typical AL iteration. In the first iteration, the training set collected during the initialization process becomes the "Current Training Dataset".

Research focused on AL has typically been focused on the specification of f_{acq} [179] and domain-specific applications, such as malware detection [180] or Land Use/Land Cover classification [181]. Acquisition functions can be divided into two different categories [40, 182]:

1. Informative-based. These strategies use the classifier's output to assess the importance of each observation towards the performance of the classifier [41].
2. Representative-based. These strategies estimate the optimal set of observations that will optimize the classifier's performance [40].

Although there are significant contributions toward the development of more robust query functions and classifiers in AL, modifications to AL's basic structure are rarely explored. In [31] the authors introduce a loss prediction module in the AL framework to replace the uncertainty criterion. This model implements

a second classifier to predict the expected loss of the unlabeled observations (using the actual losses collected during the training of the original classifier) and return the unlabeled observations with the highest expected loss. However, this contribution is specific to deep neural networks and was only tested for image classification.

AL techniques may also be used to complement other well-known learning challenges. For example, security bug report prediction tasks are typically developed in imbalanced learning environment, where it is necessary to manually label large amounts of data, which may result in mislabeled data [183]. Another related example is machinery fault diagnostics, where the quality and quantity of the data collected is often recognized as a bottleneck [184]. In this case, ML-based techniques frequently leverage unlabeled data to improve the classification performance [185] and rely on manual data acquisition [186]. In these examples, the application of an AL technique could reduce the amount of labeled data required, reduce the strain in the supervisor's labeling process and reduce the amount of label noise.

An under explored challenge in the AL literature is the effective handling of different data structures. One method to address this problem are autoencoder architectures [187] or, in the case of text data, semantic representation networks [188]. However, understanding how to integrate these two types of methods is a subject of future research. Within other research streams, such as deep reinforcement learning, some research also focus on optimizing observation efficiency during the learning process [189].

5.2.2. Data Augmentation

The standard AL model can be complemented with a data augmentation function, $f_{aug}(x_i; \tau)$, where τ defines the augmentation policy. In this context, τ refers to the transformation applied and its hyperparameters and f_{aug} produces a modified observation, $\tilde{x} \in \mathcal{D}_{aug}$ where $\mathcal{D}_{aug}$ is the set of modified observations. This involves the usage of a new set of data, $\mathcal{D}^t_{train} = \mathcal{D}^t_{lab} \cup \mathcal{D}^t_{aug}$, to train the classifier.

Data Augmentation methods expand the training dataset by introducing new and informative observations [15]. The production of artificial data may be done via the introduction of perturbations on the input [30], feature [17], or output space [15]. Data Augmentation methods may be divided into two categories [18]:

1. Heuristic approaches attempt to generate new and relevant observations by applying a predefined procedure, usually incorporating some degree of randomness [21]. Since these methods typically occur in the input space, they require fewer data and computational power when compared to Neural Network methods.

2. Neural Network approaches, on the other hand, map the original input space into a lower-dimensional representation, known as the feature space [17]. The generation of artificial data occurs in the feature space and is reconstructed into the input space. Although these methods allow the generation of less noisy data in high-dimensional contexts and more plausible artificial data, they are significantly more computationally intensive.

While some techniques may depend on the domain, others are domain-agnostic. For example, Random Erasing [16], Translation, Cropping and Flipping are examples of image data-specific augmentation methods. Other methods, such as autoencoders, may be considered domain agnostic.

5.2.3. Data Augmentation in Active Learning

The only AL model found that uses data augmentation outside of the computer vision or NLP domains implements a pipelined approach, described in [76]. In this study, the AL model proposed is applied for tabular data using an oversampling data augmentation policy (*i.e.*, the artificial data was only generated to balance the target class frequencies). However, this AL model was applied in a Land Use/Land Cover classification context with specific characteristics that are not necessarily found in other supervised learning problems. Specifically, these types of datasets are high dimensional and have limited data variability within each class (*i.e.,* cohesive spectral signatures within classes) due to their geographical proximity. Furthermore, this method does not allow augmentation policy optimization (i.e., every hyperparameter has to be hard-coded *a priori*).

The Bayesian Generative Active Deep Learning (BGDAL) [190] is another example of a pipelined combination of f_{acq} and f_{aug}, applied to image classification. BGDAL uses a Variational AutoEncoder (VAE) architecture to generate artificial observations. However, the proposed model is computationally expensive, requires a large data pool to train the VAE, and is not only dependent on the quality of the augmentations performed, but also on the performance of the discriminator and classifiers used.

The method proposed in [177], Look-Ahead Data Acquisition for Deep Active Learning, implements data augmentation to train a deep-learning classifier. However, adapting existing AL applications to use this approach is often impractical and implies the usage of image data since the augmentations used are image data specific and occur on the unlabeled observations, before the unlabeled data selection.

The Variational Adversarial Active Learning (VAAL) model [191] is a deep AL approach to image classification that uses as inputs the embeddings produced by a VAE into a secondary classifier, working as f_{acq}, to predict if $x_i \in \mathcal{D}$ belongs to $\mathcal{D}_{pool}$. The n true positives with the highest uncertainty are labeled by the supervisor and $\mathcal{D}_{pool}$ and $\mathcal{D}_{lab}$ are updated as described in Section 5.2.1. The Task-aware VAAL model [192] extends the VAAL model by introducing a ranker, which consists of the Learning Loss module introduced in [31]. These models use data augmentation techniques to train the different neural network-based components of the proposed models. However, the AL components used are specific image classification, computationally expensive and the analysis of the effect of data augmentation in these AL models is not discussed.

In [193], the proposed AL method was explicitly designed for image data classification, where a deep learning model was implemented as a classifier, but its architecture is not described, the augmentation policies used are unknown and the results reported correspond to single runs of the discussed model. The remaining AL models found implement data augmentation for NLP applications, in [194, 195]. However, these methods were designed for specific applications within that domain and are not necessarily transferable to other domains or tasks.

5.3. Proposed Method

Based on the literature found on AL, most of the contributions and novel implementations of AL algorithms have focused on the improvement of the choice/architecture of the classifier or the improvement of the uncertainty criterion. In addition, the resulting classification performance of AL-trained classifiers is frequently inconsistent and marginally improve the classification performance when compared to classifiers trained over the entire training set. In addition, there is also significant variability in the data selection efficiency during different runs of the AL iterative process [76].

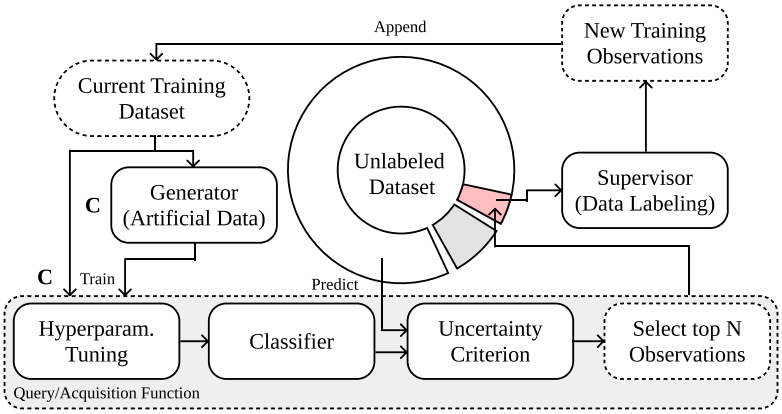


Figure 5.3.: Diagram depicting the proposed AL iteration. The proposed modifications are comprised within the red polygon and marked with a boldface "C."

This paper provides a context-agnostic AL framework for the integration of Data Augmentation within AL, with the following contributions:

1. Improvement of the AL framework by introducing a parameter tuning stage only using the labeled dataset available at the current iteration (*i.e.,* no labeled hold-out set is needed).
2. Generalization of the generator module proposed in [76] from oversampling techniques to any other data augmentation mechanism and/or policy.
3. Implementation of data augmentation outside the Deep AL realm, which was not previously found in the literature.
4. Analysis of the impact of Data Augmentation and Oversampling in AL over 15 different datasets of different domains, while comparing them with the standard AL framework.

The proposed AL framework is depicted in Figure 5.3. The generator element becomes an additional source of data and is expected to introduce additional data variability into the training dataset. This aspect should allow the classifier to generalize better and perform more consistently over unseen observations. However, in this scenario, the amount of data to generate per class at each iteration is unknown. Consequently, the hyperparameter tuning step was introduced to estimate the optimal data augmentation policy at each iteration. In our implementation, this step uses the current training dataset to perform an exhaustive search over specified generator parameters, tested over a 5-fold cross-validation method. The best augmentation policy found is used to train the iteration's classifier in the following step. This procedure is described in Algorithm 1.

We implemented a simple modification in the selection mechanism of the G-SMOTE algorithm to show the effectiveness of data augmentation in an AL implementation. We use the uncertainties produced by f_{acq} to compute the probabilities of observations to be selected for augmentation as an additional parameter. This modification is described in Algorithm 2

This modification facilitates the usage of G-SMOTE beyond its original oversampling purposes. However, in this paper, the data augmentation strategies are also used to ensure that class frequencies are balanced. Furthermore, the amount of artificial data produced for each class is defined by the *augmentation factor*, α_{af}, which represents a percentage of the majority class C_{maj} (*e.g.,* an augmentation factor of 1.2 will ensure there are $count(C_{maj}) \times 1.2$ observations in every class). In this paper's experiment, the data generation mechanism is similar to the one in [76]. This factor allows the direct comparison of the two frameworks and establishes a causality of the performance variations to the data generation mechanism

Algorithm 1: Proposed AL Framework (Single iteration)

Given: $t \geq 1$, performance metric f_{pm}
Input: $\mathcal{D}_{pool}$, $\mathcal{D}_{lab}$, f_c, f_{aug}, f_{acq}, τ_{grid}, k, n
Output: $\mathcal{D}_{pool}$, $\mathcal{D}_{lab}$

Function *ParameterTuning*(f_c, f_{aug}, τ_{grid}, $\mathcal{D}_{lab}$, k)**:**
    $p \leftarrow 0$
    $\tau \leftarrow \emptyset$
    $\{\mathcal{D}^1_{lab}, \dots \mathcal{D}^k_{lab}\} \leftarrow \mathcal{D}_{lab}$ // $\mathcal{D}^n_{lab} \cap \mathcal{D}^m_{lab} = \emptyset, \forall (n, m) \in 1, \dots, k$
    forall $\tau' \in \tau_{grid}$ **do**
        $p' \leftarrow \emptyset$
        forall $\mathcal{D}^i_{lab} \in \{\mathcal{D}^1_{lab}, \dots \mathcal{D}^k_{lab}\}$ **do**
            $\mathcal{D}'_{test} \leftarrow \mathcal{D}^i_{lab}$
            $\mathcal{D}'_{train} \leftarrow \mathcal{D}_{lab} \setminus \mathcal{D}^i_{lab}$
            $\mathcal{D}'_{train} \leftarrow f_{aug}(\mathcal{D}'_{train}; \tau')$
            train f_c using $\mathcal{D}'_{train}$
            $p' \leftarrow p' \cup \{f_{pm}(f_c(\mathcal{D}_{test}))\}$
        $p' \leftarrow \frac{\sum_{x_i \in p'} x_i}{k}$
        if $p' > p$ **then**
            $p \leftarrow p'$
            $\tau \leftarrow \tau'$
    return τ
begin
    $\tau \leftarrow ParameterTuning(f_c, f_{aug}, \tau_{grid}, \mathcal{D}_{lab}, k)$
    $\mathcal{D}_{train} \leftarrow f_{aug}(\mathcal{D}_{lab}; \tau)$
    train f_c using $\mathcal{D}_{train}$
    $\mathcal{D}_{new} = \arg\max_{\mathcal{D}'_{pool} \subset \mathcal{D}_{pool}, |\mathcal{D}'_{pool}| = n} \sum_{x \in \mathcal{D}'_{pool}} f_{acq}(x; f_c)$
    annotate $\mathcal{D}_{new}$
    $\mathcal{D}_{pool} \leftarrow \mathcal{D}_{pool} \setminus \mathcal{D}_{new}$
    $\mathcal{D}_{lab} \leftarrow \mathcal{D}_{lab} \cup \mathcal{D}_{new}$

(*i.e.,* augmentation vs normal oversampling) and hyperparameter tuning steps. However, in this case, the hyperparameter tuning is solely going to be used for augmentation policy optimization.

In the proposed framework, we (1) generalize the generator module to accept any data augmentation method or policy and (2) introduce a hyperparameter tuning module to estimate the optimal data augmentation policy. This framework was designed to be task-agnostic. Specifically, any data augmentation method (domain-specific or not) may be applied, as well as any other parameter search method. It is also expected to be compatible with other AL modifications, including those that do not affect solely the classifier or uncertainty criterion, such as the one proposed in [31].

5.4. Methodology

This section describes the different elements included in the experimental procedure. The datasets used were acquired in open data repositories. Their sources and preprocessing steps are defined in Subsection 5.4.1. The classifiers used in the experiment are defined in Subsection 5.4.2. The metrics

Algorithm 2: G-SMOTE Modified for Data Augmentation in AL

Given: $t \geq 1$, $\mathcal{D}^t_{lab} \neq \emptyset$, $\mathcal{D}_{lab} = \mathcal{D}^{min}_{lab} \cup \mathcal{D}^{maj}_{lab}$, $GSMOTE$
Input: $\mathcal{D}^t_{pool}$, $\mathcal{D}^t_{lab}$, f_c^{t-1}, f_{acq}, τ
Output: $\mathcal{D}^t_{train}$

Function *DataSelection*($\mathcal{D}^t_{lab}$, f_{acq}, f_c^{t-1})**:**
  $U \leftarrow \emptyset$
  $P \leftarrow \emptyset$
  $p_s \sim \mathcal{U}(0, 1)$
  forall $x_i \in \mathcal{D}^t_{lab}$ **do**
    $u_{x_i} \leftarrow f_{acq}(x_i; f_c^{t-1})$
    $U \leftarrow U \cup \{u_{x_i}\}$
  forall $u_{x_i} \in U$ **do**
    $p_{x_i} \leftarrow \frac{u_{x_i}}{\sum U} + \sum P$
    $P \leftarrow P \cup \{p_{x_i}\}$
  $i \leftarrow argmax(P < p_s)$
  return i-th element in $\mathcal{D}^t_{lab}$
begin
  $\mathcal{D}^{min}_{aug} \leftarrow \emptyset$
  $\mathcal{D}^{maj}_{aug} \leftarrow \emptyset$
  $\alpha_{af}, \alpha_{trunc}, \alpha_{def} \leftarrow \tau$
  $N \leftarrow count(C_{maj}) \times \alpha_{af}$
  forall $\mathcal{D}'_{aug} \in \{\mathcal{D}^{min}_{aug}, \mathcal{D}^{maj}_{aug}\}$, $\mathcal{D}'_{lab} \in \{\mathcal{D}^{min}_{lab}, \mathcal{D}^{maj}_{lab}\}$ **do**
    while $|\mathcal{D}'_{aug}| < N$ **do**
      $x_{center} \leftarrow DataSelection(\mathcal{D}'_{lab}, f_{acq}, f_c^{t-1})$
      $x_{gen} \leftarrow GSMOTE(x_{center}, \mathcal{D}^t_{lab}, \alpha_{trunc}, \alpha_{def})$
      $\mathcal{D}'_{aug} \leftarrow \mathcal{D}'_{aug} \cup \{x_{gen}\}$
  $\mathcal{D}_{aug} \leftarrow \mathcal{D}^{min}_{aug} \cup \mathcal{D}^{maj}_{aug}$
  $\mathcal{D}^t_{train} \leftarrow \mathcal{D}^t_{lab} \cup \mathcal{D}_{aug}$

chosen to measure AL performance and overall classification performance are defined in Subsection 5.4.3. The experimental procedure is described in Subsection 5.4.4.

The methodology developed serves a two-fold purpose: (1) Compare classification performance once all the AL procedures are completed (*i.e.,* optimal performance of a classifier trained via iterative data selection) and (2) Compare the amount of data required to reach specific performance thresholds (*i.e.,* the number of AL iterations required to reach similar classification performances).

5.4.1. Datasets

The datasets used to test the proposed method are publicly available in open data repositories. Specifically, they were retrieved from the OpenML and the UCI Machine Learning Repository websites. They were chosen considering diverse application domains, imbalance ratios, dimensionality and number of target classes, all of them focused on classification tasks. The goal is to demonstrate the performance of the different AL frameworks in various scenarios and domains. The data preprocessing approach was similar

Table 5.2.: Description of the datasets collected after data preprocessing. The sampling strategy is similar across datasets. Legend: (IR) Imbalance Ratio

Dataset	Features	Instances	Minority instances	Majority instances	IR	Classes
Image Segmentation	14	1155	165	165	1.0	7
Mfeat Zernike	47	1994	198	200	1.01	10
Texture	40	1824	165	166	1.01	11
Waveform	40	1666	551	564	1.02	3
Pendigits	16	1832	176	191	1.09	10
Vehicle	18	846	199	218	1.1	4
Mice Protein	69	1073	105	150	1.43	8
Gas Drift	128	1987	234	430	1.84	6
Japanese Vowels	12	1992	156	323	2.07	9
Usps	256	1859	142	310	2.18	10
Gesture Segmentation	32	1974	200	590	2.95	5
Volkert	147	1943	45	427	9.49	10
Steel Plates	24	1941	55	673	12.24	7
Baseball	15	1320	57	1196	20.98	3
Wine Quality	11	1599	10	681	68.1	6

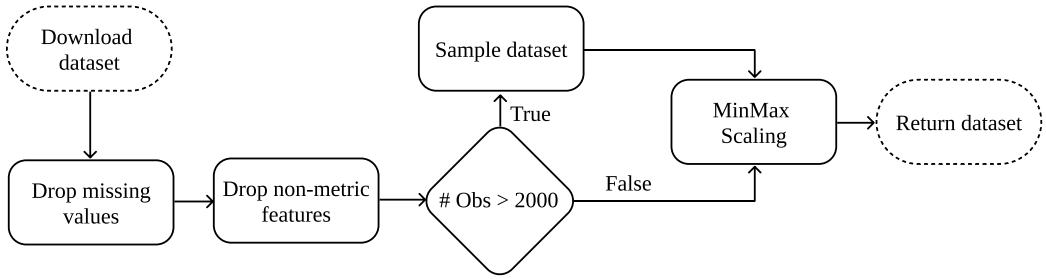


Figure 5.4.: Data preprocessing pipeline.

across all datasets. Table 5.2 describes the key properties of the 15 preprocessed datasets where the experimental procedure was applied.

The data preprocessing pipeline is depicted as a flowchart in Figure 5.4. The missing values are removed from each dataset by removing the corresponding observations. This step ensures that the input data in the experiment is kept as close to its original form as possible. The non-metric features (*i.e.,* binary, categorical, and ordinal variables) were removed since the application of G-SMOTE is limited to continuous and discrete features. The datasets containing over 2000 observations were downsampled in order to maintain the datasets to a manageable size. The data sampling procedure preserves the relative class frequency of the dataset, in order to maintain the Imbalance Ratio (IR) originally found in each dataset (where $IR = \frac{count(C_{maj})}{count(C_{\min})}$). The remaining features of each dataset are scaled to the range of $[-1, 1]$ to ensure a common range across features.

The preprocessed datasets were stored into an SQLite database file and is available along with the experiment's source code in the project's GitHub repository (see final remarks regarding data and software availability).

5.4.2. Machine Learning Algorithms

We used a total of four classification algorithms and a heuristic data augmentation mechanism. The choice of classifiers was based on the popularity and family of the classifiers (tree-based, nearest neighbors-based, ensemble-based and linear models). Our proposed method was tested using a Decision Tree (DT) [196], a K-nearest neighbors classifier (KNN) [128], a Random Forest Classifier (RF) [163] and a Logistic Regression (LR) [127]. Since the target variables are multi-class, the LR classifier was implemented using the one-versus-all approach. The predicted class is assigned to the label with the highest likelihood.

The oversampler G-SMOTE was used as a data augmentation method. The typical data generation policy of oversampling methods is to generate artificial observations on non-majority classes such that the number of majority class observations matches those of each non-majority class. We modified this data generation policy to generate observations for all classes, as a percentage of the number of observations in the majority class. In addition, the original G-SMOTE algorithm was modified to accept data selection probabilities based on classification uncertainty. These modifications are discussed in Section 5.3.

Every AL procedure was tested with different selection criteria: Random Selection, Entropy, and Breaking Ties. The baseline used is the standard AL procedure. As a benchmark, we add the AL procedure using G-SMOTE as a standard oversampling method, as proposed in [76]. Our proposed method was implemented using G-SMOTE as a data augmentation method to generate artificial observations for all classes, while still balancing the class distribution, as described in Section 5.3.

5.4.3. Evaluation Metrics

Considering the imbalanced nature of the datasets used in the experiment, commonly used performance metrics such as Overall Accuracy (OA), although being intuitive to interpret, are insufficient to quantify a model's classification performance [132]. The Cohen's Kappa performance metric, similar to OA, is also biased towards high-frequency classes since its definition is closely related to the OA metric, making its behavior consistent with OA [197]. However, these metrics remain popular choices for the evaluation of classification performance. Other performance metrics like $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$ (also known as Sensitivity) or $Specificity = \frac{TN}{TN+FP}$ are calculated as a function of True/False Positives (TP and FP) and True/False Negatives (TN and FN) and can be used on a per-class basis instead. In a multiple dataset scenario with varying amounts of target classes and meanings, comparing the performance of different models using these metrics becomes impractical.

Based on the recommendations found in [132, 164], we used two metrics found to be less sensitive to the class imbalance bias, along with OA as a reference for easier interpretability:

- The Geometric-mean scorer (G-mean) consists of the geometric mean of Specificity and Recall [164]. Both metrics are calculated in a multi-class context considering a one-versus-all approach. For multi-class problems, the G-mean scorer is calculated as its average per class values:

$$G\text{-}mean = \sqrt{\overline{Recall} \times \overline{Specificity}}$$

- The F-score metric consists of the harmonic mean of Precision and Recall. The two metrics are also calculated considering a one-versus-all approach. The F-score for the multi-class case can be calculated using its average per class values [132]:

$$F\text{-}score = 2 \times \frac{\overline{Precision} \times \overline{Recall}}{\overline{Precision} + \overline{Recall}}$$

- The OA consists of the number of TP divided by the total amount of observations. Considering c as the label for the different classes present in a target class, OA is given by the following formula:

$$OA = \frac{\sum_c \mathrm{TP}_c}{\sum_c (\mathrm{TP}_c + \mathrm{FP}_c)}$$

The comparison of the performance of AL frameworks is based on its data selection and augmentation efficacy. Specifically, an efficient data selection/generation policy allows the production of classifiers with high performance on unseen data while using as least non-artificial training data as possible. We follow the recommendations found in [36]. To measure the performance of the different AL setups, the performance of an AL setup will be compared using two AL-specific performance metrics:

- Area Under the Learning Curve (AULC). It is the sum of the classification performance over a validation/test set of the classifiers trained of all AL iterations. The resulting AULC scores are fixed within the range $[0, 1]$ by dividing the AULC scores by the total amount of iterations (*i.e.*, the maximum performance area) to facilitate the interpretability of this metric.
- Data Utilization Rate (DUR) [165]. Measures the percentage of training data required to reach a given performance threshold, as a ratio of the percentage of training data required by the baseline framework. This metric is also presented as a percentage of the total amount of training data, without making it relative to the baseline framework. The DUR metric is measured at 45 different performance thresholds, ranging between $[0.10, 1.00]$ at a 0.02 step.

5.4.4. Experimental Procedure

The evaluation of different active learners in a live setting is generally expensive, time-consuming, and prone to human error. Instead, a common practice is to compare them in an offline environment using labeled datasets [166]. Since the dataset is already labeled, the annotation process is done at zero cost in this scenario. Figure 5.5 depicts the experiment designed for one dataset over a single run.

A single run starts with the splitting of a preprocessed dataset into five different partitions, stratified according to the class frequencies of the target variable using the K-fold Cross Validation method. During this run, an active learner or classifier is trained five times using a different partition as the Test set each time. For each training process, a validation set containing 25% of the subset is created and is used to measure the data selection efficiency (*i.e.,* AULC and DUR using the classification performance metrics, specific to AL). Therefore, for a single training procedure, 20% of the original dataset is used as the validation set, 20% is used as the Test set and 60% is used as the training set. The AL simulations and the classifiers' training occur within the training set. However, the classifiers used to find the maximum performance classification scores are trained over the full training set. The AL simulations are run over a maximum of 50 iterations (including the initialization step), adding 1.6% of the training set each time (*i.e.,* all AL simulations use less than 80% of the training set). Once the training phase is completed, the Test set classification scores are calculated using the trained classifiers. For the case of AL, the classifier with the optimal validation set score is used to estimate the AL's optimal classification performance over unseen data.

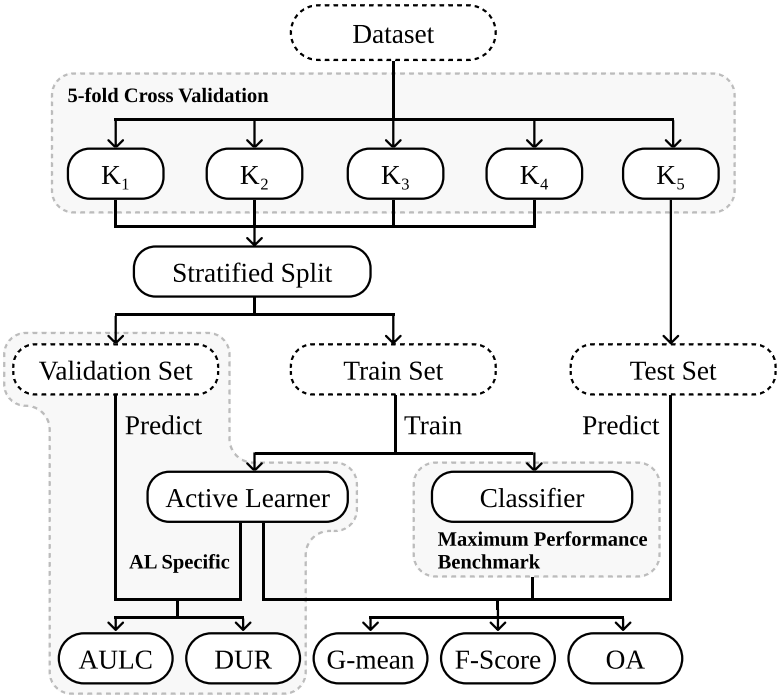


Figure 5.5.: Experimental procedure flowchart. The preprocessed datasets are split into five folds. One of the folds is used to test the best-found classifiers using AL and the classifiers trained using the entire training dataset (containing the remaining folds). The training set is used to run both the AL simulations as well as train the normal classifiers. The validation set is used to measure AL-specific performance metrics over each iteration. We use different subsets for overall classification performance and AL-specific performance to avoid data leakage.

The process shown in Figure 5.5 is repeated over three runs using different random seeds over the 15 different datasets collected. The final scores of each AL configuration and classifier correspond to the average of the three runs and 5-fold Cross-Validation estimations (*i.e.,* the mean score of 15 fits, across 15 datasets).

The hyperparameters defined for the AL frameworks, Classifiers, and Generators are shown in Table 5.3. In the Generators table, we distinguish the G-SMOTE algorithm working as a normal oversampling method from G-SMOTE-AUGM, which generates additional artificial data on top of the usual oversampling mechanism. Since the G-SMOTE-AUGM method is intended to be used with varying parameter values (via within-iteration parameter tuning), the parameters were defined as a list of various possible values. The remaining parameters were selected based on knowledge gathered in previous literature and typical default values for each of the algorithms. This choice was motivated by the impossibility of parameter tuning in a real-world setting when applying the benchmark AL methods. Although the proposed method addresses this limitation, we show that exclusively tuning the parameters on the augmentation policy is already sufficient to achieve superior, statistically significant performance.

5.4.5. Software Implementation

The experiment was implemented using the Python programming language, along with the Python libraries Scikit-Learn [86], Imbalanced-Learn [134], Geometric-SMOTE [23], Research-Learn and ML-Research libraries. All functions, algorithms, experiments, and results are provided in the project's GitHub

Table 5.3.: Hyperparameter definition for the active learners, classifiers, and generators used in the experiment.

Active Learners	Hyperparameters	Inputs
Standard	# initial obs.	1.6%
	# additional obs. per iteration	1.6%
	max. iterations + initialization	50
	evaluation metrics	G-mean, F-score, OA
	selection strategy	Random, Entropy, Breaking Ties
	within-iteration param. tuning	None
	generator	None
	classifier	DT, LR, KNN, RF
Oversampling	generator	G-SMOTE
Proposed	generator	G-SMOTE-AUGM
	within-iteration param. tuning	Grid Search K-fold CV
Classifier		
DT	min. samples split	2
	criterion	gini
LR	maximum iterations	100
	multi-class	One-vs-All
	solver	liblinear
	penalty	L2 (Ridge)
KNN	# neighbors	5
	weights	uniform
	metric	euclidean
RF	min. samples split	2
	# estimators	100
	criterion	gini
Generator		
G-SMOTE	# neighbors	4
	deformation factor	0.5
	truncation factor	0.5
G-SMOTE-AUGM	# neighbors	3, 4, 5
	deformation factor	0.5
	truncation factor	0.5
	augmentation factor	$[1.1, 2.0]$ at 0.1 step

repository. The original datasets used in this study are publicly available in open data repositories. They were retrieved from OpenML and the UCI Machine Learning Repository.

5.5. Results & Discussion

In a multiple dataset experiment, the analysis of results should not rely upon the average performance scores across datasets uniquely. The domain of application and fluctuations of performance scores between datasets make the analysis of these averaged results less accurate. Instead, it is generally recommended to use the mean ranking scores to extend the analysis [135]. Since mean performance scores are still intuitive to interpret; we will present and discuss both results. The rank values are assigned based on the mean scores of three different 5-fold Cross-Validation runs (15 performance estimations per dataset) for each combination of dataset, AL configuration, classifier, and performance metric.

5.5.1. Results

The average rankings of the AL methods' AULC estimations are shown in Table 5.4. The proposed method almost always improves AL performance and ensures higher data selection efficiency.

Table 5.4.: Mean rankings of the AULC metric over the different datasets (15), folds (5), and runs (3) used in the experiment. The proposed method constantly improves the results of the original framework and, on average, almost always improves the results of the oversampling framework.

Classifier	Evaluation Metric	Standard	Oversampling	Proposed
DT	Accuracy	2.13 ± 0.96	2.40 ± 0.49	**1.47 ± 0.62**
DT	F-score	2.47 ± 0.81	2.20 ± 0.40	**1.33 ± 0.70**
DT	G-mean	2.73 ± 0.57	1.93 ± 0.44	**1.33 ± 0.70**
KNN	Accuracy	2.07 ± 0.93	2.07 ± 0.68	**1.87 ± 0.81**
KNN	F-score	2.47 ± 0.81	1.87 ± 0.50	**1.67 ± 0.87**
KNN	G-mean	2.87 ± 0.34	**1.47 ± 0.50**	1.67 ± 0.70
LR	Accuracy	2.13 ± 0.88	2.20 ± 0.65	**1.67 ± 0.79**
LR	F-score	2.80 ± 0.40	1.87 ± 0.50	**1.33 ± 0.70**
LR	G-mean	2.80 ± 0.40	1.80 ± 0.54	**1.40 ± 0.71**
RF	Accuracy	2.27 ± 0.85	**1.87 ± 0.50**	**1.87 ± 0.96**
RF	F-score	2.73 ± 0.57	1.80 ± 0.54	**1.47 ± 0.72**
RF	G-mean	2.87 ± 0.34	**1.53 ± 0.50**	1.60 ± 0.71

Table 5.5 shows the average AULC scores, grouped by the classifier, Evaluation Metric and AL framework. The performance of the proposed method is almost always superior when considering the F-score and G-mean. On some occasions, the average AULC score is significantly improved when compared with the oversampling AL method.

The average DUR scores were calculated for various G-mean thresholds, varying between 0.1 and 1.0 at a 0.02 step (45 different thresholds in total). Table 5.6 shows the results obtained for these scores starting from a G-mean score of 0.6 and was filtered to show the thresholds ending with 0 or 6 only. In most cases, the proposed method reduces the amount of data annotation required to reach each G-mean score threshold.

Table 5.5.: Average AULC of each AL configuration tested. Each AULC score is calculated using the performance scores of each iteration in the validation set. By the end of the iterative process, each AL configuration used a maximum of 80% instances of the 60% instances that compose the training sets (*i.e.,* 48% of the entire preprocessed dataset).

Classifier	Evaluation Metric	Standard	Oversampling	Proposed
DT	Accuracy	0.663 ± 0.149	0.658 ± 0.153	**0.664 ± 0.155**
DT	F-score	0.610 ± 0.176	0.612 ± 0.179	**0.618 ± 0.181**
DT	G-mean	0.744 ± 0.129	0.751 ± 0.127	**0.755 ± 0.129**
KNN	Accuracy	**0.741 ± 0.160**	0.730 ± 0.178	0.734 ± 0.179
KNN	F-score	0.678 ± 0.208	0.684 ± 0.211	**0.687 ± 0.213**
KNN	G-mean	0.786 ± 0.152	**0.804 ± 0.139**	**0.804 ± 0.141**
LR	Accuracy	**0.736 ± 0.152**	0.723 ± 0.185	0.731 ± 0.184
LR	F-score	0.644 ± 0.228	0.673 ± 0.220	**0.682 ± 0.221**
LR	G-mean	0.767 ± 0.162	0.811 ± 0.134	**0.814 ± 0.136**
RF	Accuracy	**0.789 ± 0.148**	0.786 ± 0.153	0.785 ± 0.156
RF	F-score	0.724 ± 0.214	**0.735 ± 0.204**	**0.735 ± 0.205**
RF	G-mean	0.818 ± 0.150	**0.834 ± 0.135**	0.833 ± 0.135

The DUR scores relative to the Standard AL method are shown in Figure 5.6. A DUR below 1 means that the Proposed/Oversampling method requires less data than the Standard AL method to reach the same performance threshold. For example, running an AL simulation using the KNN classifier requires 80.7% of the amount of data required by the Standard AL method using the same classifier to reach an F-Score of 0.62 (*i.e.,* requires 19.3% less data).

The comparison of mean optimal classification scores of AL methods with Classifiers (using the entire training set, without AL) is shown in Table 5.7. Aside from the case of overall accuracy, the proposed AL method produces classifiers that almost consistently outperform classifiers using the whole training set (*i.e.,* the ones labeled as MP).

5.5.2. Statistical Analysis

When checking for statistical significance in a multiple dataset context it is critical to account for the multiple comparison problem. Consequently, our statistical analysis focuses on the recommendations found in [135]. Overall, we perform three statistical tests. The Friedman test [136] is used to understand whether there is a statistically significant difference in performance between the three AL frameworks. As *post hoc* analysis, the Wilcoxon signed-rank test [137] was utilized to check for statistical significance between the performance of the proposed AL method and the oversampling AL method across datasets. As a second *post hoc* analysis, the Holm-Bonferroni [198] method was employed to check for statistical significance between the methods using data generators and the Standard AL framework across classifiers and evaluation metrics.

Table 5.8 displays the *p-values* obtained with the Friedman test. The difference in performance across AL frameworks is statistically significant at a level of $\alpha = 0.05$ regardless of the classifier or evaluation metric being considered.

Table 5.9 contains the *p-values* obtained with the Wilcoxon signed-rank test. The proposed method was able to outperform both the standard AL framework, as well as the AL framework using a typical oversampling policy with statistical significance in 14 and 12 out of 15 datasets, respectively.

Table 5.6.: AL algorithms' mean data utilization as a percentage of the training set.

G-mean Score	Classifier	Standard	Oversampling	Proposed
0.60	DT	19.8%	**18.9%**	19.3%
0.60	KNN	18.4%	**11.8%**	12.8%
0.60	LR	23.0%	**9.7%**	**9.7%**
0.60	RF	14.1%	**7.7%**	7.8%
0.66	DT	23.1%	23.3%	**22.9%**
0.66	KNN	23.9%	**21.7%**	21.9%
0.66	LR	25.6%	**20.5%**	**20.5%**
0.66	RF	22.0%	17.6%	**17.5%**
0.70	DT	25.5%	25.0%	**24.8%**
0.70	KNN	26.8%	24.1%	**23.9%**
0.70	LR	29.9%	23.6%	**23.4%**
0.70	RF	23.8%	**22.1%**	22.3%
0.76	DT	33.4%	30.5%	**30.1%**
0.76	KNN	34.0%	27.7%	**27.3%**
0.76	LR	38.0%	27.6%	**26.2%**
0.76	RF	28.2%	**24.5%**	24.7%
0.80	DT	48.2%	43.8%	**41.2%**
0.80	KNN	38.8%	**34.4%**	34.6%
0.80	LR	43.7%	32.6%	**31.3%**
0.80	RF	32.4%	**27.2%**	27.7%
0.86	DT	69.6%	66.5%	**64.8%**
0.86	KNN	53.9%	**52.0%**	52.5%
0.86	LR	48.7%	45.3%	**45.0%**
0.86	RF	43.9%	**40.0%**	**40.0%**
0.90	DT	81.2%	79.4%	**76.6%**
0.90	KNN	60.9%	61.1%	**60.4%**
0.90	LR	62.1%	62.9%	**59.9%**
0.90	RF	57.1%	**55.7%**	56.2%
0.96	DT	100.0%	**99.7%**	100.0%
0.96	KNN	82.4%	79.7%	**77.1%**
0.96	LR	86.5%	84.0%	**81.8%**
0.96	RF	70.8%	71.1%	**70.3%**

The *p-values* shown in Table 5.10 refer to the results of the Holm-Bonferroni test. The proposed method's superior performance was statistically significant in 9 out of 12 cases.

5.5.3. Discussion

In this paper, we study the application of data augmentation methods through the modification of the standard AL framework. This is done to further reduce the amount of labeled data required to produce a reliable classifier, at the expense of artificial data generation. Overall, the proposed method achieves better and more consistent performance when compared to the remaining benchmark approaches. It was implemented to focus on the optimization of the data augmentation policy, as well as the introduction of a more informed AL-based data augmentation approach. The proposed method could be further extended and achieve an even higher performance by optimizing parameters of the ML classification using the

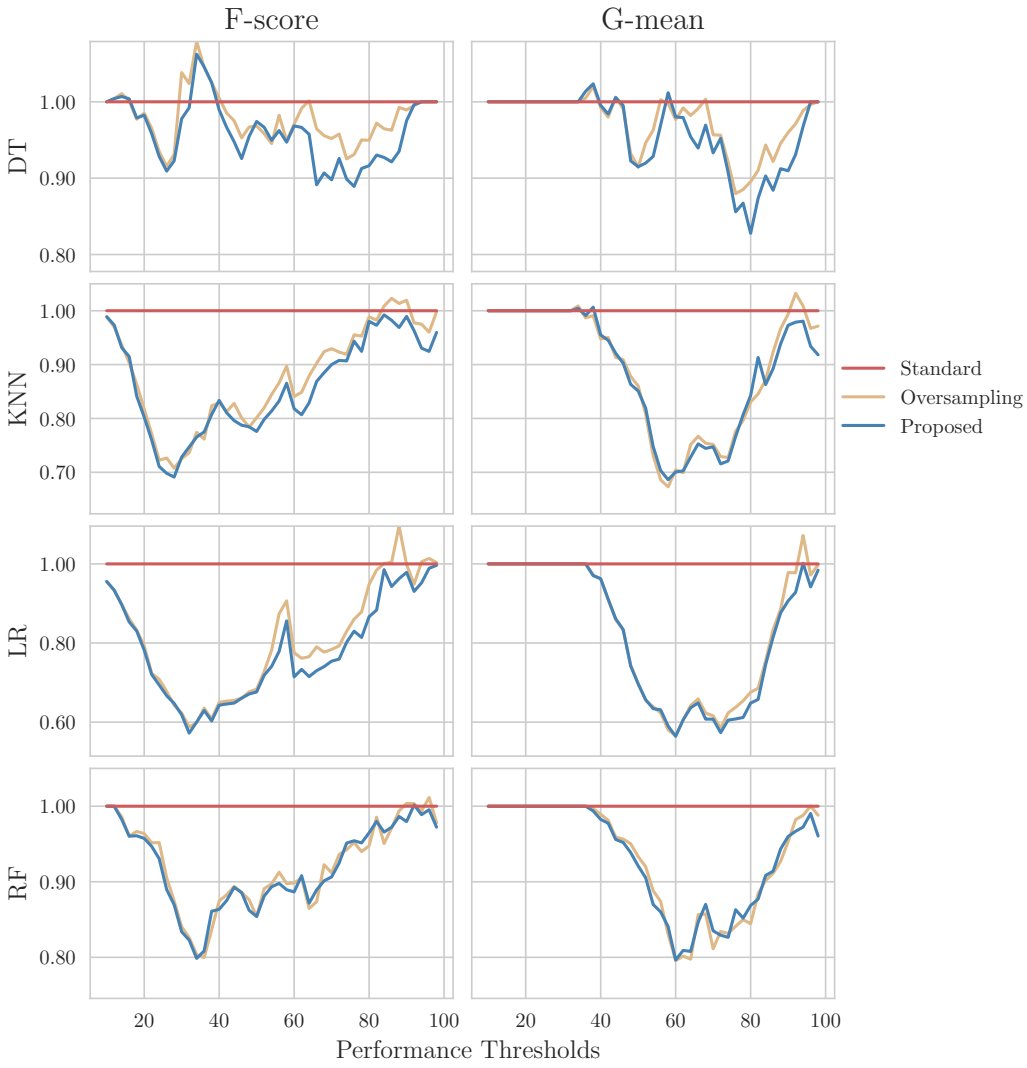


Figure 5.6.: Mean data utilization rates. The y-axis shows the percentage of data (relative to the baseline AL framework) required to reach the different performance thresholds.

hyperparameter optimizer. In addition, this framework could be further generalized by searching, within AL iterations, for the optimal ML classifier as well; at different stages of the data collection procedure some ML classifiers might be more useful than others. Although the proposed framework significantly improves the flexibility of AL implementations, we found that even a superficial parameter search is sufficient to ensure a superior performance when compared to related approaches.

In Table 5.4, we found that the proposed method was able to outperform the Standard AL framework in all scenarios. Except for the overall accuracy metric, the mean rankings are consistent with the mean AULC scores found in Table 5.5, while showing performance improvements between the proposed method and both the standard and oversampling methods. The Friedman test in Table 5.8 showed that the difference in the performance of these AL frameworks are statistically significant, regardless of the classifier or performance metric being used.

The proposed method evidenced more consistent data utilization requirements in most of the assessed G-mean score thresholds when compared to the remaining AL methods, as seen in Table 5.6. For example, to reach a G-mean score of 0.9 using the KNN and LR classifiers, the average amount of data required with the Oversampling AL approach increased when compared to the standard approach. However, the

Table 5.7.: Optimal classification scores. The Maximum Performance (MP) classification scores are calculated using classifiers trained using the entire training set.

Classifier	Evaluation Metric	MP	Standard	Oversampling	Proposed
DT	Accuracy	**0.732 ± 0.155**	0.726 ± 0.157	0.721 ± 0.167	0.727 ± 0.168
DT	F-score	0.682 ± 0.194	0.679 ± 0.193	0.679 ± 0.197	**0.684 ± 0.200**
DT	G-mean	0.792 ± 0.138	0.791 ± 0.136	0.797 ± 0.134	**0.800 ± 0.137**
KNN	Accuracy	**0.801 ± 0.164**	0.799 ± 0.168	0.784 ± 0.183	0.789 ± 0.183
KNN	F-score	0.742 ± 0.224	0.744 ± 0.223	0.741 ± 0.223	**0.746 ± 0.224**
KNN	G-mean	0.827 ± 0.160	0.829 ± 0.158	0.839 ± 0.146	**0.840 ± 0.147**
LR	Accuracy	0.778 ± 0.157	**0.791 ± 0.158**	0.764 ± 0.184	0.773 ± 0.185
LR	F-score	0.693 ± 0.243	0.717 ± 0.241	0.718 ± 0.222	**0.727 ± 0.226**
LR	G-mean	0.796 ± 0.171	0.814 ± 0.165	0.839 ± 0.130	**0.842 ± 0.137**
RF	Accuracy	0.827 ± 0.145	**0.832 ± 0.148**	0.827 ± 0.154	0.829 ± 0.153
RF	F-score	0.767 ± 0.215	0.775 ± 0.216	0.781 ± 0.204	**0.784 ± 0.204**
RF	G-mean	0.844 ± 0.148	0.849 ± 0.149	0.863 ± 0.131	**0.865 ± 0.131**

Table 5.8.: Friedman test results. Statistical significance is tested at a level of $\alpha = 0.05$. The null hypothesis is that there is no difference in the classification outcome across oversamplers.

Classifier	Evaluation Metric	p-value	Significance
DT	Accuracy	1.1e-15	True
DT	F-score	2.4e-31	True
DT	G-mean	2.3e-23	True
KNN	Accuracy	5.9e-20	True
KNN	F-score	8.8e-69	True
KNN	G-mean	8.8e-52	True
LR	Accuracy	1.1e-30	True
LR	F-score	4.0e-98	True
LR	G-mean	2.3e-83	True
RF	Accuracy	2.8e-26	True
RF	F-score	1.8e-88	True
RF	G-mean	1.8e-61	True

proposed method was able to decrease the amount of data required in both situations. The robustness of the proposed method is clearer in Figure 5.6. In most cases, this method was able to outperform the Oversampling method. At the same time, the proposed method also addresses inconsistencies in situations where the Oversampling method was unable to outperform the standard method.

The statistical analyses found in Tables 5.9 and 5.10 revealed that the proposed method's superiority was statistically significant in all datasets except three (Baseball, Usps, and Volkert) and established statistical significance when compared to the standard AL method for all combinations of classifier and performance metric, except for three cases regarding the use of the overall accuracy metric. These results show that the proposed method increased the reliability of the new AL framework and improved the quality of the final classifier while using fewer data.

Even though it was not the core purpose of this study, we found that the proposed AL method consistently outperformed the maximum performance threshold. Specifically, in Table 5.7, the performance of the classifiers originating from the proposed method was able to outperform classifiers trained using the full training dataset in 9 out of 12 scenarios. This outcome suggests that the selection of a meaningful training

Table 5.9.: Adjusted p-values using the Wilcoxon signed-rank method. Bold values are statistically significant at a level of $\alpha = 0.05$. The null hypothesis is that the performance of the proposed framework is similar to that of the oversampling or standard framework.

Dataset	Oversampling	Standard
Baseball	5.0e-01	3.4e-01
Gas Drift	**3.7e-26**	**4.6e-57**
Gesture Segmentation	**1.3e-02**	**8.7e-04**
Image Segmentation	**9.6e-18**	**2.1e-44**
Japanese Vowels	**2.4e-09**	**1.6e-32**
Mfeat Zernike	**1.2e-12**	**9.5e-40**
Mice Protein	**6.5e-32**	**1.5e-61**
Pendigits	**5.0e-18**	**2.3e-45**
Steel Plates	**3.4e-04**	**1.3e-08**
Texture	**1.5e-22**	**6.7e-57**
Usps	3.8e-01	**2.1e-29**
Vehicle	**7.4e-11**	**7.9e-13**
Volkert	2.5e-01	**1.3e-02**
Waveform	**8.9e-08**	**2.6e-02**
Wine Quality	**3.8e-05**	**6.1e-03**

Table 5.10.: Adjusted p-values using the Holm-Bonferroni method. Bold values are statistically significant at a level of $\alpha = 0.05$. The null hypothesis is that the Oversampling or Proposed method does not perform better than the control method (Standard AL framework).

Classifier	Evaluation Metric	Oversampling	Proposed
DT	Accuracy	7.7e-01	**1.1e-04**
DT	F-score	6.3e-02	**2.0e-06**
DT	G-mean	**1.0e-08**	**2.9e-12**
KNN	Accuracy	**1.0e-02**	8.5e-01
KNN	F-score	**7.1e-07**	**8.3e-13**
KNN	G-mean	**1.9e-11**	**1.0e-12**
LR	Accuracy	**3.2e-02**	8.3e-01
LR	F-score	**1.5e-09**	**5.8e-17**
LR	G-mean	**1.9e-13**	**5.6e-16**
RF	Accuracy	4.3e-01	4.3e-01
RF	F-score	**1.4e-11**	**1.1e-12**
RF	G-mean	**1.5e-10**	**1.2e-10**

subset training dataset paired with data augmentation not only matches the classification performance of ML algorithms, as it also improves them. Even in a setting with fully labeled training data, the proposed method may be used as a preprocessing technique to further optimize classification performance.

This study discussed the effect of data augmentation within the AL framework, along with the exploration of optimal augmentation methods within AL iterations. However, the conceptual nature of this study implies some limitations. Specifically, the large number of experiments required to test the method's efficacy, along with the limited computational power available, led to a limited exploration of the grid search's potential. Future work should focus on understanding how the usage of a more comprehensive parameter tuning approach improves the quality of the AL method. In addition, the proposed method was not able to outperform the standard AL method at 100% of scenarios. The exploration of other, more complex data augmentation techniques might further improve its performance by producing more meaningful training observations. Specifically, in this study, we assume that all datasets used follow a manifold, allowing the usage of G-SMOTE as a data augmentation approach. However, this method cannot be used in more complex, non-euclidean spaces. In this scenario, the usage of G-SMOTE is not valid and might lead to the production of noisy data. Deep Learning-based data augmentation techniques are able to address this limitation and improve the overall quality of the artificial data being generated. We also encountered significant standard errors throughout our experimental results (see Subsection 5.5.1), consistent with the findings in [36, 76]. This facet suggests that the usage of more robust generators did not decrease the standard error of AL performance. Instead, AL's performance variability is likely dependent on the quality of its initialization.

5.6. Conclusion and Future Directions

The ability to train ML classifiers is usually limited to the availability of labeled data. However, manually labeling data is often expensive, which makes the usage of AL particularly appealing for selecting the most informative observations and reducing the amount of required labeled data. On the other hand, the introduction of data variability in the training dataset can also be conducted via data augmentation. However, most, if not all, AL configurations that use some form of data augmentation are domain and/or task-specific. These methods typically apply deep learning approaches to both classification and data augmentation. Consequently, they may not apply to other classification tasks or when the available computational power is insufficient.

In this paper, we proposed a domain-agnostic AL framework that implements Data Augmentation and hyperparameter tuning. We found that a heuristic Data Augmentation algorithm is sufficient to improve the data selection efficiency in AL. Specifically, the data augmentation method used almost always increased AL performance, regardless of the target goal (*i.e.,* optimizing classification or data selection efficiency). The usage of data augmentation reduced the number of iterations required to train a classifier with a performance as good as (or better than) classifiers trained with the entire training dataset (*i.e.,* without using AL). In addition, the proposed method reduced the size of the training dataset, which is expanded with artificial data.

With this revised AL configuration, data selection in AL iterations aims towards observations that optimize the quality of the artificial data produced. The substitution of less informative labeled data with artificial data is especially useful in this context since it reduces some of the user interaction necessary to reach a sufficiently informative dataset. In order to further improve the proposed method, future work should (1) focus on the development of methods with varying data augmentation policies depending on the different input space regions, (2) develop augmentation-sensitive query functions capable of avoiding the unnecessary selection of similar observations from the unlabeled dataset, (3) understand the gap between randomized data augmentation techniques and neural network/feature space data augmentation

techniques in an AL context better, (4) explore more efficient ways to leverage the information collected in AL queries for better augmentation strategies and (5) expand the current framework to integrate alternative learning strategies using unlabeled data, such as self and semi supervised learning techniques.

Finally, the proposed method may be applied to any classification problem where labeled data is not readily available and an easily accessible unlabeled data pool. For more complex data structures, the application of this framework will require the learning of a manifold space as an additional preprocessing step. After that, this AL framework may be used as is.

6. Geometric SMOTE for Imbalanced Datasets with Nominal and Continuous Features

Submitted as Joao Fonseca, Fernando Bacao, to a Q1 Journal, 2023

Imbalanced learning can be addressed in 3 different ways: resampling, algorithmic modifications and cost-sensitive solutions. Resampling, and specifically oversampling, are more general approaches when opposed to algorithmic and cost-sensitive methods. Since the proposal of the Synthetic Minority Oversampling TEchnique (SMOTE), various SMOTE variants and neural network-based oversampling methods have been developed. However, the options to oversample datasets with nominal and continuous features are limited. We propose Geometric SMOTE for Nominal and Continuous features (G-SMOTENC), based on a combination of G-SMOTE and SMOTENC. Our method modifies SMOTENC's encoding and generation mechanism for nominal features while using G-SMOTE's data selection mechanism to determine the center observation and k-nearest neighbors and generation mechanism for continuous features. G-SMOTENC's performance is compared against SMOTENC's along with two other baseline methods, a State-of-the-art oversampling method and no oversampling. The experiment was performed over 20 datasets with varying imbalance ratios, number of metric and non-metric features and target classes. We found a significant improvement in classification performance when using G-SMOTENC as the oversampling method. An open-source implementation of G-SMOTENC is made available in the Python programming language.

Keywords: Imbalanced Learning; Oversampling; SMOTE; Data Generation; Nominal Data

6.1. Introduction

Various Machine Learning (ML) tasks deal with highly imbalanced datasets, such as fraud transactions detection, fault detection and medical diagnosis [199]. In these situations, predicting false positives is often a more acceptable error, since the class of interest is usually the minority class [200]. However, using standard ML classifiers on imbalanced datasets induces a bias in favor of the classes with the highest frequency, while limiting the predictive power on lower frequency classes [201, 202]. This effect is known in the ML community as the Imbalanced Learning problem.

Imbalanced learning involves a dataset with two or more target classes with varying class frequencies. The minority class is defined as the class with the least amount of observations and the majority class is the one with the highest amount of observations [203]. There are three main approaches to address imbalanced learning [204]:

1. Cost-sensitive solutions attribute a higher misclassification cost to the minority class observations to minimize higher cost errors;
2. Algorithmic level solutions modify ML classifiers to improve the learning of the minority class;
3. Resampling solutions generate synthetic minority class observations and/or remove majority class observations to balance the training dataset;

Since it is an external approach to imbalanced learning, the latter method becomes particularly useful. It dismisses the required domain knowledge to build a cost matrix and the technical complexity or knowledge to apply an imbalanced learning-specific classifier. Resampling can be done via undersampling, oversampling, or hybrid approaches [205]. In this paper, we will focus on oversampling approaches.

The presence of nominal features in imbalanced learning tasks limits the options available to deal with class imbalance. Even though it is possible to use encoding methods such as one-hot or ordinal encoding to convert nominal features into numerical, applying a distance metric on mixed-type datasets is questionable since the nominal feature values are unordered [206]. In this case, one possible approach is to use models that can handle different scales (*e.g.*, Decision Tree). However, this assumption may be limiting since there are few ML algorithms where this condition is verified. Another possible approach is transforming the variables to meet scale assumptions [206]. This method was explored in the algorithm Synthetic Minority Oversampling Technique for Nominal and Continuous features (SMOTENC) [22] (explained in Section 6.2).

In the presence of datasets with mixed data types, using most of the well-known resampling algorithms becomes unfeasible. This happens because these methods consider exclusively continuous data; they were not adapted to also use nominal features. Specifically, since the proposal of SMOTE, various other SMOTE-variants have been developed to address some of its limitations. Although, there was not a significant development in research to oversample datasets with both nominal and continuous features.

In this paper, we propose Geometric SMOTE for Nominal and Continuous features (G-SMOTENC). It generates the continuous feature values of a synthetic observation within a truncated hyper-spheroid with its nominal feature values using the most common value of its nearest neighbors. In addition, G-SMOTENC uses G-SMOTE's data selection strategy and SMOTENC's approach to find the center observation's nearest neighbors. G-SMOTENC is a generalization of both SMOTENC and G-SMOTE [23]. With the correct hyperparameters, our G-SMOTENC implementation can mimic the behavior of SMOTE, SMOTENC, or G-SMOTE. It is available in the open-source Python library "ML-Research" and is fully compatible with the Scikit-Learn ecosystem. These contributions can be summarized as follows:

1. We propose G-SMOTENC, an oversampling algorithm for datasets with nominal and continuous features;
2. We test the proposed oversampler using 20 datasets and compare its performance to SMOTENC, Random Oversampling, Random Undersampling and a State-of-the-art oversampler;
3. We provide an implementation of G-SMOTENC in the Python programming language;

The rest of this paper is structured as follows: Section 6.2 describes the related work and its limitations, Section 6.3 describes the proposed method (G-SMOTENC), Section 6.4 lays out the methodology used to test G-SMOTENC, Section 6.5 shows and discusses the results obtained in the experiment and Section 6.6 presents the conclusions drawn from this study.

6.2. Related Work

A classification problem contains n classes, having C_{maj} as the set of majority class observations (*i.e.*, observations belonging to the most common target class) and C_{min} as the set of minority class observations (*i.e.*, observations belonging to the least common target class). Typically, an oversampling algorithm will generate synthetic data in order to ensure $|C'_{min}| = |C_{maj}| = |C_i|, i \in \{1, \ldots, n\}$.

Since the proposal of SMOTE, other methods modified or extended SMOTE to improve the quality of the data generated. The process of generating synthetic data using SMOTE-based algorithms can be divided into two distinct phases [207]:

1. Data selection. A synthetic observation, x^{gen}, is generated based on two existing observations. A SMOTE-based algorithm employs a given heuristic to select a non-majority class observation as the center observation, x^c, and one of its nearest neighbors, x^{nn}, selected randomly. For the case of SMOTE, x^c is randomly selected from each non-majority class.
2. Data generation. Once x^c and x^{nn} have been selected, x^{gen} is generated based on a transformation between the two selected observations. In the case of SMOTE, this transformation is a linear interpolation between the two observations: $x^{gen} = \alpha x^c + (1 - \alpha)x^{nn}, \alpha \sim \mathcal{U}(0, 1)$.

Modifications to the SMOTE algorithm can be distinguished according to the phase where they were applied. This distinction is especially relevant for the case of oversampling on datasets with mixed data types since it raises the challenge of calculating meaningful distances and k-nearest neighbors among observations. For example, State-of-the-art oversampling methods, such as Borderline-SMOTE [24], ADASYN [123], K-means SMOTE [25] and LR-SMOTE [208] modify the data selection mechanism and show promising results in imbalanced learning [30]. However, these algorithms select x^c using procedures that include calculating each observation's k-nearest neighbors or clustering methods, which are not prepared to handle nominal data.

Modifications to SMOTE's generation mechanism are uncommon. A few oversampling methods, such as Safe-level SMOTE [209] and Geometric-SMOTE [23] proposed this type of modification and have shown promising results [2]. However, these methods are also unable to handle datasets with nominal data. Other methods attempt to replace the SMOTE data generation mechanism altogether using different Generative Adversarial Networks (GAN) architectures [210, 211, 212]. Network-based architectures, however, are computationally expensive to train and sensitive to the training initialization. It is also difficult to ensure a balanced training of the two networks involved and tuning their hyperparameters is often challenging or unfeasible [213].

As discussed in Section 6.1, research on resampling methods with mixed data types is scarce. The original paper proposing SMOTE also proposed SMOTE for Nominal and Continuous (SMOTENC), an adaptation of SMOTE to handle datasets with nominal and continuous features [22]. To determine the k-nearest neighbors of x^c, the Euclidean distance is modified to include the median of the standard deviations of the continuous features for every nominal feature with different values. Once x^c and x^{nn} are defined, the continuous feature values in x^{gen} are generated using the SMOTE generation mechanism. The nominal features are given the most common values occurring in the k-nearest neighbors.

Recently, a new SMOTE-based oversampling method for datasets with mixed data types, SMOTE-ENC [214], was proposed. This method modifies the encoding mechanism for nominal features used in the SMOTENC algorithm to account for nominal features' change of association with minority classes. The Multivariate Normal Distribution-based Oversampling for Numerical and Categorical features (MNDO-NC) [215] uses the original MNDO method [216] along with the SMOTENC encoding mechanism to find the values of the categorical features for the synthetic observation. However, the results reported in the

paper showed that MNDO-NC was consistently outperformed by SMOTENC, which led us to discard this approach from further consideration.

Alternatively to SMOTE-based methods, it is possible to use non-informed over and undersampling methods for datasets with nominal and continuous features, specifically Random Oversampling (ROS) and Random Undersampling (RUS). These methods consist of randomly duplicating minority class observations (in the case of ROS), which can lead to overfitting [217, 218], or randomly removing majority class observations (in the case of RUS), which may lead to underfitting [219].

6.3. Proposed Method

We propose G-SMOTENC to oversample imbalanced datasets with both nominal and continuous features. Our method builds on top of G-SMOTE's selection and generation mechanisms coupled with a modified version of SMOTENC. It attributes less importance to the nominal features (relative to the continuous features) when computing distances among observations compared to SMOTENC. However, this method can be extended with further modifications to the nominal data encoding and selection mechanisms in future work.

Similar to G-SMOTE being an extension of SMOTE, G-SMOTENC is also an extension of SMOTENC since any method or ML pipeline using the SMOTENC generation mechanism can replace it with G-SMOTENC without any further modifications. The proposed method is described in pseudo-code in Algorithm 3. The functions $SelectionMechanism$ and $GenerationMechanism$ are described in Algorithms 4 and 5, respectively.

Algorithm 3: G-SMOTENC.

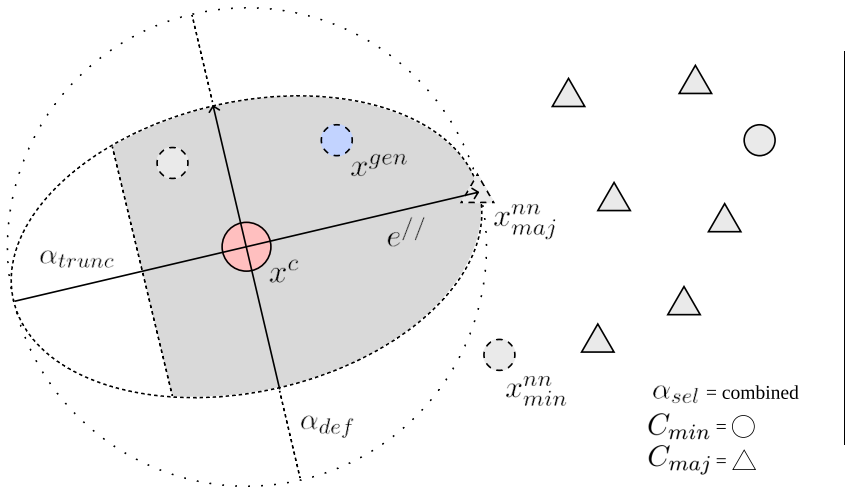
Given: Dataset with binary target classes C_{min} and C_{maj}
Input: $C_{maj}, C_{min}, \alpha_{sel}, \alpha_{trunc}, \alpha_{def}$
Output: C^{gen}

begin
  $N \leftarrow |C_{maj}| - |C_{min}|$
  $C^{gen} \leftarrow \emptyset$
  while $|C^{gen}| < N$ **do**
    $x^c, x^{nn}, X^{nn} \leftarrow SelectionMechanism(C_{maj}, C_{min}, \alpha_{sel})$
    $x^{gen} \leftarrow GenerationMechanism(x^c, x^{nn}, X^{nn}, \alpha_{trunc}, \alpha_{def})$
    $C^{gen} \leftarrow C^{gen} \cup \{x^{gen}\}$

G-SMOTENC's implementation involves additional considerations regarding the management of the nominal features. During the selection mechanism (identified as the function $SelectionMechanism$), the nominal features are encoded using the one-hot encoding technique, while the non-zero constant assumes the value of the median of the standard deviations of the continuous features in C_{min}, divided by two. This encoding mechanism varies from the one in SMOTENC in order to attribute less weight to the nominal features relative to the continuous features.

The selection strategy, α_{sel}, as well as C_{min} and C_{maj} are used to determine a central observation, x^c, its nearest neighbors, X^{nn}, and one of its nearest neighbors, $x^{nn} \in X^{nn}$. X^{nn} is calculated using the euclidean distance and both the continuous and encoded nominal features. The outcome of this step is dependent on the choice of α_{sel}:

1. If $\alpha_{sel} = minority$, X^{nn} will consist of x^c's k-nearest neighbors within C_{min};



X^{nn}		f_1^{nom}	f_2^{nom}	f_3^{nom}
x^{nn}_{maj}	△	A	C	B
x^{nn}_{min}	○	B	B	B
n/a	○	B	C	A
x^{gen}_{nom}	○	B	C	B

Figure 6.1.: A visual depiction of G-SMOTENC. In this example, α_{trunc} is approximately 0.5 and α_{def} is approximately 0.4.

2. If $\alpha_{sel} = majority$, X^{nn} will consist of x^c's nearest neighbor within C_{maj};
3. If $\alpha_{sel} = combined$, X^{nn} will consist of the union between x^c's k-nearest neighbors within C_{min} and x^c's nearest neighbor within C_{maj}, *i.e.*, $C_{min,k} \cup C_{maj,1}$. In this case, x^{nn} is selected using the majority class observation within X^{nn} as well as another randomly selected nearest neighbor, such that $x^{nn} = argmin(||x^{nn}_{min} - x^c||, ||x^{nn}_{maj} - x^c||)$;

Unlike in the original G-SMOTE generation mechanism, X^{nn} is used in the to determine the nominal feature values of x^{gen} based on the mode of these features within X^{nn}. G-SMOTENC's generation mechanism (identified as $GenerationMechanism$) uses two hyperparameters to generate the continuous features in x^{gen}: the truncation factor, α_{trunc}, and the deformation factor, α_{def}. They are generated by forming a hyper-sphere with center x^c and is modified according to the parameters:

1. α_{trunc} truncates the hyper-sphere to induce the generation of the artificial instance within a subset of the hypersphere. It varies between 1 and -1, where 1 would split the generation area in half and use the area between x^c and x^{nn}, -1 achieves the same effect and uses the other semi-hyper-sphere, and 0 applies no truncation.
2. α_{def} deforms the hyper-sphere as shown in Figure 6.1. It varies between 0 and 1, where 0 applies no deformation and 1 fully deforms the hyper-sphere into a line segment, corresponding to $e^{//}$.

Figure 6.1 depicts the effect of those hyperparameters in the data selection and generation phases. For an in-depth explanation of these hyperparameters, the reader is referred to [23].

6.3.1. Selection Mechanism

The data selection mechanism is preceded by the numerical encoding of the nominal features. It combines the selection mechanisms of SMOTENC and G-SMOTE, as shown in Algorithm 4. The selection mechanism inherits the minority, majority, and combined mechanisms proposed in G-SMOTE. The nominal features in the minority and majority class observations, C_{maj} and C_{min} are first encoded using a one-hot encoding approach and replacing the constant 1 with the median of the standard deviations of the continuous features in C_{min} divided by 2. The nearest-neighbors (X^{nn}) of x^c are determined based on α_{sel}, which are passed on to the generation mechanism to determine the nominal features' values of x^{gen} in the

generation mechanism. Simultaneously, x^{nn} is randomly selected from X^{nn} and will be used to generate x^{gen}'s continuous features' values.

Algorithm 4: G-SMOTENC's selection mechanism.

Input: $C_{maj}, C_{min}, \alpha_{sel}$
Output: x^c, x^{nn}, X^{nn}

Function *CatEncoder(C_{maj}, C_{min})*:
  $S \leftarrow$ Standard deviations of the continuous features in C_{min}
  $\sigma_{med} \leftarrow median(S)$
  forall $i \in \{maj, min\}$ **do**
    forall $f \in C_i^T$ **do**
      if f *is nominal* **then**
        $f' \leftarrow OneHotEncode(f) \times \sigma_{med}/2$
        $C_i' \leftarrow (C_i^T \setminus f)^T$
        $C_i' \leftarrow (C_i'^T \cup f')^T$
  return C'_{maj}, C'_{min}
Function *Surface(α_{sel}, x^c, C_{maj}, C_{min})*:
  if $\alpha_{sel} = minority$ **then**
    $x^{nn} \in C_{min,k}$ `// One of the k-nearest neighbors of` x^c `from` C_{min}
    $X^{nn} \leftarrow C_{min,k}$
  if $\alpha_{sel} = majority$ **then**
    $x^{nn} \in C_{maj,1}$ `// Nearest neighbor of` x^c `from` C_{maj}
    $X^{nn} \leftarrow C_{maj,1}$
  if $\alpha_{sel} = combined$ **then**
    $x^{nn}_{min} \in C_{min,k}$
    $x^{nn}_{maj} \in C_{maj,1}$
    $x^{nn} \leftarrow argmin(||x^{nn}_{min} - x^c||, ||x^{nn}_{maj} - x^c||)$
    $X^{nn} \leftarrow C_{min,k} \cup C_{maj,1}$
  return x^{nn}, X^{nn} `//` X^{nn} `is the set of` k`-nearest neighbors`
begin
  $C'_{maj}, C'_{min} \leftarrow CatEncoder(C_{maj}, C_{min})$
  $x^c \in C'_{min}$ `// Randomly select` x^c `from` C'_{min}
  $x^{nn}, X^{nn} \leftarrow Surface(\alpha_{sel}, x^c, C'_{maj}, C'_{min})$
  Reverse encoding of nominal features in x^c, x^{nn} and X^{nn}

6.3.2. Generation Mechanism

G-SMOTENC's generation mechanism is shown in Algorithm 5. It divides the generation of x^{gen} into two parts: (1) generation of continuous feature values and (2) generation of nominal feature values. First, the nominal features from x^c and x^{nn} are discarded. Afterward, the continuous features are generated using G-SMOTE's generation mechanism; within a hyper-spheroid defined with α_{trunc} and α_{def}, which allows the non-linear generation of synthetic observations between x^c and x^{nn}. Finally, the nominal feature values are generated by the mode of each feature within the observations in X^{nn}.

Algorithm 5: G-SMOTENC's generation mechanism.

Input: $x^c, x^{nn}, X^{nn}, \alpha_{trunc}, \alpha_{def}$

Output: x^{gen}

Function *Hyperball()*:
- $v_i \sim \mathcal{N}(0,1)$
- $r \sim \mathcal{U}(0,1)$
- $x^{gen} \leftarrow r^{1/p} \frac{(v_1,\ldots,v_p)}{||(v_1,\ldots,v_p)||}$
- **return** x^{gen}

Function *Vectors*(x^c, x^{nn}, x^{gen}):
- $e^{//} \leftarrow \frac{x^{nn}-x^c}{||x^{nn}-x^c||}$
- $x^{//} \leftarrow (x^{gen} \cdot e^{//})e^{//}$
- $x^{\perp} \leftarrow x^{gen} - x^{//}$
- **return** $x^{//}, x^{\perp}$

Function *Truncate*($x^c, x^{nn}, x^{gen}, x^{//}, \alpha_{trunc}$):
- **if** $|\alpha_{trunc} - x^{//}| > 1$ **then**
  - $x^{gen} \leftarrow x^{gen} - 2x^{//}$
- **return** x^{gen}

Function *Deform*($x^{gen}, x^{\perp}, \alpha_{def}$):
- **return** $x^{gen} - \alpha_{def} x^{\perp}$

Function *Translate*(x^c, x^{gen}, R):
- **return** $x^c + Rx^{gen}$

Function *GenNominal*(X^{nn}):
- $x^{gen}_{nom} = \emptyset$
- **forall** $f \in (X^{nn})^T$ **do**
  - **if** *f is nominal* **then**
    - $x^{gen}_{nom} \cup \{mode(f)\}$ `// Ties are decided with random selection`
- **return** x^{gen}_{nom}

begin
- Discard nominal features from x^c and x^{nn}
- $x^{gen} \leftarrow Hyperball()$
- $x^{//}, x^{\perp} \leftarrow Vectors(x^c, x^{nn}, x^{gen})$
- $x^{gen} \leftarrow Truncate(x^c, x^{nn}, x^{gen}, x^{//}, \alpha_{trunc})$
- $x^{gen} \leftarrow Deform(x^{gen}, x^{\perp}, \alpha_{def})$
- $x^{gen} \leftarrow Translate(x^c, x^{gen}, ||x^{nn}_{cont} - x^c||)$
- $x^{gen}_{nom} \leftarrow GenNominal(X^{nn})$
- $x^{gen} \leftarrow x^{gen} \cup x^{gen}_{nom}$

6.4. Methodology

This section describes how the evaluation of G-SMOTENC was performed. We describe the datasets used in the experiment, their source and preprocessing steps executed in Section 6.4.1. The resampling and classification methods used to analyze G-SMOTE's performance are listed in Section 6.4.2. The performance metrics used are defined in Section 6.4.3. Finally, the experimental procedure is described in Section 6.4.4.

6.4.1. Experimental Data

The datasets used in this experiment were extracted from the UC Irvine Machine Learning Repository. All of the datasets are publicly available and cover a range of different domains. The criteria to select the datasets ensured that all datasets are imbalanced and contained non-metric features (*i.e.*, ordinal, nominal or binary). These datasets are used to show how the performance of different classifiers varies across over/undersamplers.

All datasets were initially preprocessed manually with minimal manipulations. We removed features and/or observations with missing values and identified the non-metric features. The second stage of preprocessing was done systematically. It starts with the generation of artificially imbalanced datasets with different Imbalance Ratios ($IR = \frac{|C_{maj}|}{|C_{min}|}$). For each original dataset, we create its more imbalanced versions at intervals of 10, while ensuring that $|C_{min}| \geq 15$. The sampling strategy was determined for class $n \in \{1, \ldots, n, \ldots, m\}$ as a linear interpolation using $|C_{maj}|$ and $|C'_{min}| = \frac{|C_{maj}|}{IR_{new}}$, as shown in equation 6.1.

$$|C_i|^{imb} = \min(\frac{|C'_{min}| - |C_{maj}|}{n-1}.|C_i| + |C_{max}|, |C_i|) \tag{6.1}$$

The new, artificially imbalanced dataset, is formed by sampling observations without replacement from each C_i such that $C'_i \subseteq C_i, |C'_i| = |C_i|^{imb}$. The artificially imbalanced datasets are marked with its imbalance ratio as a suffix in Table 6.1.

The datasets (both original and artificially imbalanced versions) are then filtered to ensure all datasets have a minimum of 500 observations. The remaining datasets with a number of observations larger than 5000 are randomly sampled to match this number of observations. Afterward, we remove target classes with a frequency lower than 15 observations for each remaining dataset. Finally, the continuous and discrete features are scaled to the range $[0, 1]$ to ensure a common range between all features. The description of the resulting datasets is shown in Table 6.1.

Table 6.1.: Description of the datasets collected after data preprocessing. The sampling strategy is similar across datasets. Legend: (IR) Imbalance Ratio

Dataset	Metric	Non-Metric	Obs.	Min. Obs.	Maj. Obs.	IR	Classes
Abalone	1	7	4139	15	689	45.93	18
Adult	8	6	5000	1268	3732	2.94	2
Adult (10)	8	6	5000	451	4549	10.09	2
Annealing	4	6	790	34	608	17.88	4

Continued on next page

Table 6.1.: Description of the datasets collected after data preprocessing. The sampling strategy is similar across datasets. Legend: (IR) Imbalance Ratio

Dataset	Metric	Non-Metric	Obs.	Min. Obs.	Maj. Obs.	IR	Classes
Census	24	7	5000	337	4663	13.84	2
Contraceptive	4	5	1473	333	629	1.89	3
Contraceptive (10)	4	5	1036	62	629	10.15	3
Contraceptive (20)	4	5	990	31	629	20.29	3
Contraceptive (31)	4	5	973	20	629	31.45	3
Contraceptive (41)	4	5	966	15	629	41.93	3
Covertype	2	10	5000	20	2449	122.45	7
Credit Approval	9	6	653	296	357	1.21	2
German Credit	13	7	1000	300	700	2.33	2
German Credit (10)	13	7	770	70	700	10.00	2
German Credit (20)	13	7	735	35	700	20.00	2
German Credit (30)	13	7	723	23	700	30.43	2
German Credit (41)	13	7	717	17	700	41.18	2
Heart Disease	5	5	740	22	357	16.23	5
Heart Disease (21)	5	5	735	17	357	21.00	5

6.4.2. Machine Learning Algorithms

The choice of classifiers used in the experimental procedure was based on their type (tree-based, nearest neighbors-based, linear model and ensemble-based), popularity and consistency in performance. We used Decision Tree (DT), a K-Nearest Neighbors (KNN) classifier, a Logistic Regression (LR) and a Random Forest (RF).

Given the lack of existing oversamplers that address imbalanced learning problems with mixed data types, the amount of benchmark methods used is also limited. We used three appropriate, well-known methods and one state-of-the-art oversampling method: SMOTENC, RUS, ROS and SMOTE-ENC. Table 6.2 shows the hyperparameters used for the parameter search described in Section 6.4.4.

6.4.3. Performance Metrics

The choice of the performance metric plays a critical role in assessing the effect on classification tasks. The typical performance metrics, *e.g.*, Overall Accuracy (OA), are intuitive to interpret but are often inappropriate to measure a classifier's performance in an imbalanced learning context [220]. For example, to estimate an event that occurs in 1% of the dataset, a constant classifier would obtain an OA of 0.99 and still be unusable. However, this metric is still reported in some of our results to maintain interpretability.

Recent surveys consider Geometric-mean (G-mean), F1-score (F-score), $Sensitivity = \frac{TP}{FN+TP}$ and $Specificity = \frac{TN}{TN+FP}$ appropriate and common performance metrics in imbalanced learning contexts [132, 221, 222]. G-mean and F-score are defined equations 6.2 and 6.3, respectively.

$$G\text{-}mean = \sqrt{\overline{Sensitivity} \times \overline{Specificity}} \tag{6.2}$$

Table 6.2.: Hyperparameter definition for the classifiers and resamplers used in the experiment.

Classifier	Hyperparameter	Values
DT	min. samples split	2
	criterion	gini
	max depth	3, 6
LR	maximum iterations	10000
	multi-class	One-vs-All
	solver	saga
	penalty	None, L1, L2
KNN	# neighbors	3, 5
	weights	uniform
	metric	euclidean
RF	min. samples split	2
	# estimators	50, 100
	Max depth	3, 6
	criterion	gini
Resampler		
SMOTENC	# neighbors	3, 5
SMOTE-ENC	# neighbors	3, 5
G-SMOTENC	# neighbors	3, 5
	deformation factor	0.0, 0.25, 0.5, 0.75, 1.0
	truncation factor	-1.0, -0.5, 0.0, 0.5, 1.0
	selection strategy	"combined", "minority", "majority"
RUS	replacement	False
ROS	(no applicable parameters)	

$$F\text{-}score = 2 \times \frac{\overline{Precision} \times \overline{Recall}}{\overline{Precision} + \overline{Recall}} \tag{6.3}$$

They are calculated as a function of the number of False/True Positives (FP and TP) and False/True Negatives (FN and TN), with $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$. This led us to use, along with OA, both F-score and G-mean as the main performance metrics for this study.

6.4.4. Experimental Procedure

The experimental procedure was applied similarly to all combinations of resamplers, classifiers and hyperparameter combinations across all datasets. The evaluation of the models' performance was tested using a 5-fold Cross-Validation (CV) approach. The mean performance in the test set is calculated over the five folds and three different runs of the experimental procedure for each combination of resampling/classifier hyperparameters. For each dataset, we select the results of the hyperparameters that optimize the performance of a resampler/classifier. Figure 6.2 shows a diagram of the experimental procedure described.

A CV run consists of a stratified partitioning (*i.e.*, each partition contains the same relative frequencies of target labels) of the dataset into five parts. A given resampler/classifier combination with a specific set of hyperparameters is fit and tested five times, using one of the partitions as a test set and the remaining

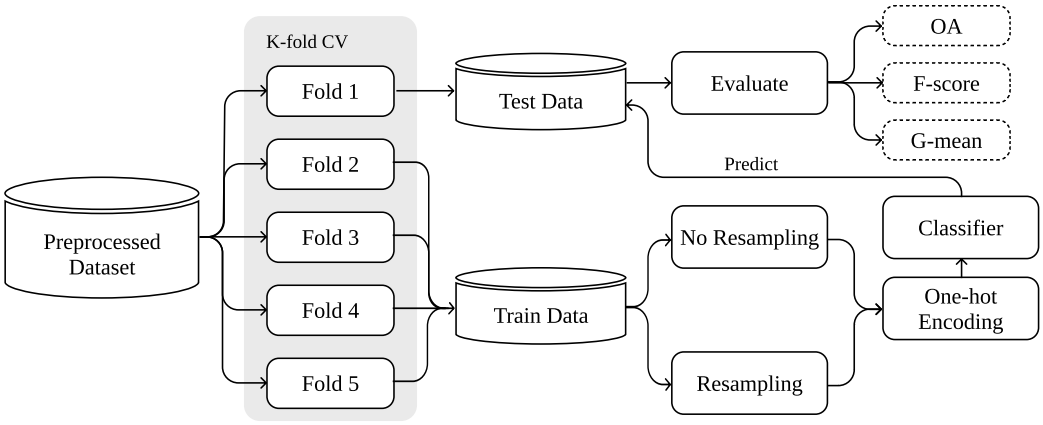


Figure 6.2.: Experimental procedure used in this study.

ones as the training set. In the ML pipeline defined for each run, the nominal features are one-hot encoded after oversampling and before passing the data to the classifier. The estimated performance consists of the average classification performance across the five tests and three runs (*i.e.*, a total of 15 tests).

6.4.5. Software Implementation

The algorithmic implementation of G-SMOTENC was written using the Python programming language and is available in the open-source package ML-Research [76], along with other utilities used to produce the experiment and outputs used in Section 6.5. In addition, the packages Scikit-Learn [86], Imbalanced-Learn [134] and Research-Learn were also used in the experimental procedure to get the implementations of the classifiers, benchmark over/undersamplers and run the experimental procedure. The original SMOTE-ENC implementation was retrieved from the authors' GitHub repository. The Latex code, Python scripts (including data pulling and preprocessing, experiment setup and analysis of results), as well as the datasets used, are available in this GitHub repository.

6.5. Results and Discussion

In this section, we present the experimental results. We focus on the comparison of classification performance using oversamplers whose generation mechanism is compatible with datasets containing both nominal and continuous features. The experimental results were analyzed in two stages: (1) in Section 6.5.1 we analyze mean rankings and absolute performances and in Section 6.5.2 we show the results of our statistical analysis. Section 6.5.3 discusses the main insights extracted by analyzing the experimental results.

6.5.1. Results

Table 6.3 presents the mean rankings of CV scores between the different combinations of oversamplers, metrics and classifiers. These results were calculated by assigning a ranking score for each oversampler from 1 (best) to 4 (worst) for each dataset, metric and classifier.

Table 6.3.: Mean rankings over the different datasets, folds and runs used in the experiment.

Classifier	Metric	G-SMOTENC	NONE	SMOTENC	ROS	RUS	SMOTE-ENC
DT	OA	1.66 ± 0.13	**1.61 ± 0.27**	3.58 ± 0.20	4.68 ± 0.15	5.42 ± 0.27	4.05 ± 0.23
DT	F-Score	**1.32 ± 0.11**	3.84 ± 0.40	3.13 ± 0.20	4.32 ± 0.19	5.47 ± 0.23	2.92 ± 0.34
DT	G-Mean	**1.68 ± 0.24**	5.84 ± 0.09	2.82 ± 0.21	2.95 ± 0.32	4.26 ± 0.32	3.45 ± 0.30
KNN	OA	2.50 ± 0.17	**1.37 ± 0.28**	4.21 ± 0.25	3.34 ± 0.35	5.68 ± 0.22	3.89 ± 0.15
KNN	F-Score	**1.37 ± 0.16**	3.95 ± 0.35	3.11 ± 0.29	3.47 ± 0.36	5.53 ± 0.23	3.58 ± 0.23
KNN	G-Mean	**1.74 ± 0.17**	5.84 ± 0.12	2.89 ± 0.23	3.76 ± 0.33	3.00 ± 0.45	3.76 ± 0.23
LR	OA	2.74 ± 0.19	**1.37 ± 0.28**	3.08 ± 0.21	4.34 ± 0.30	5.74 ± 0.17	3.74 ± 0.28
LR	F-Score	**2.11 ± 0.24**	4.53 ± 0.35	2.37 ± 0.28	3.47 ± 0.32	5.21 ± 0.27	3.32 ± 0.38
LR	G-Mean	2.13 ± 0.26	6.00 ± 0.00	3.61 ± 0.21	**2.11 ± 0.23**	3.32 ± 0.40	3.84 ± 0.28
RF	OA	1.82 ± 0.11	**1.24 ± 0.09**	3.97 ± 0.16	4.32 ± 0.21	5.92 ± 0.06	3.74 ± 0.22
RF	F-Score	**1.32 ± 0.13**	5.05 ± 0.31	3.16 ± 0.22	3.05 ± 0.31	5.37 ± 0.14	3.05 ± 0.27
RF	G-Mean	**1.68 ± 0.22**	5.79 ± 0.21	3.26 ± 0.28	2.47 ± 0.30	3.89 ± 0.35	3.89 ± 0.19

Table 6.4 presents the mean CV scores. Except for the OA metric, G-SMOTENC either outperformed or matched the remaining oversamplers.

Table 6.4.: Mean scores over the different datasets, folds and runs used in the experiment

Classifier	Metric	G-SMOTENC	NONE	SMOTENC	ROS	RUS	SMOTE-ENC
DT	OA	0.74 ± 0.05	**0.75 ± 0.04**	0.68 ± 0.04	0.66 ± 0.04	0.58 ± 0.04	0.65 ± 0.04
DT	F-Score	**0.56 ± 0.04**	0.52 ± 0.04	0.54 ± 0.04	0.52 ± 0.04	0.48 ± 0.04	0.51 ± 0.04
DT	G-Mean	**0.69 ± 0.03**	0.60 ± 0.02	0.68 ± 0.03	0.67 ± 0.03	0.65 ± 0.03	0.66 ± 0.03
KNN	OA	0.69 ± 0.04	**0.73 ± 0.05**	0.67 ± 0.04	0.69 ± 0.05	0.57 ± 0.04	0.68 ± 0.05
KNN	F-Score	**0.53 ± 0.04**	0.50 ± 0.04	0.52 ± 0.04	0.52 ± 0.04	0.46 ± 0.04	0.51 ± 0.04
KNN	G-Mean	**0.66 ± 0.03**	0.58 ± 0.03	0.64 ± 0.03	0.62 ± 0.03	0.65 ± 0.03	0.63 ± 0.03
LR	OA	0.68 ± 0.05	**0.75 ± 0.04**	0.68 ± 0.05	0.66 ± 0.05	0.58 ± 0.04	0.67 ± 0.04
LR	F-Score	**0.54 ± 0.04**	0.52 ± 0.04	**0.54 ± 0.04**	0.53 ± 0.04	0.48 ± 0.04	0.52 ± 0.04
LR	G-Mean	**0.69 ± 0.02**	0.60 ± 0.03	0.68 ± 0.02	**0.69 ± 0.03**	0.67 ± 0.03	0.67 ± 0.03
RF	OA	0.74 ± 0.04	**0.76 ± 0.04**	0.69 ± 0.04	0.69 ± 0.04	0.59 ± 0.04	0.68 ± 0.05
RF	F-Score	**0.57 ± 0.04**	0.48 ± 0.04	0.55 ± 0.04	0.55 ± 0.04	0.49 ± 0.04	0.53 ± 0.04
RF	G-Mean	**0.70 ± 0.02**	0.57 ± 0.02	0.68 ± 0.03	0.69 ± 0.03	0.68 ± 0.03	0.68 ± 0.02

6.5.2. Statistical Analysis

It is necessary to use methods that account for the multiple comparison problem to conduct an appropriate statistical analysis in an experiment with multiple datasets. Based on the recommendations found in [135], we applied a Friedman test followed by a Holm-Bonferroni test for post-hoc analysis.

In Section 6.4.3 we explained that OA, although easily interpretable, is not an appropriate performance metric for imbalanced learning problems. Therefore, the statistical analysis was developed using the two imbalance-appropriate metrics used in the study: F-Score and G-Mean. Based on the Friedman test [136], there is a statistically significant difference in performance across resampling methods. The results of this test are shown in Table 6.5. The null hypothesis is rejected in all cases.

Table 6.5.: Results for the Friedman test. Statistical significance is tested at a level of $\alpha = 0.05$. The null hypothesis is that there is no difference in the classification outcome across resamplers.

Classifier	Metric	p-value	Significance
DT	F-Score	2.2e-10	True
DT	G-Mean	1.2e-10	True
KNN	F-Score	2.3e-09	True
KNN	G-Mean	9.4e-10	True
LR	F-Score	2.1e-07	True
LR	G-Mean	9.7e-11	True
RF	F-Score	8.5e-12	True
RF	G-Mean	2.0e-10	True

We performed a Holm-Bonferroni test to understand whether the difference in the performance of G-SMOTENC is statistically significant to the remaining resampling methods. The results of this test are shown in Table 6.6. The null hypothesis is rejected in 33 out of 40 trials.

Table 6.6.: Adjusted p-values using the Holm-Bonferroni test. Statistical significance is tested at a level of $\alpha = 0.05$. The null hypothesis is that the benchmark methods perform similarly to the control method (G-SMOTENC).

Classifier	Metric	NONE	SMOTENC	ROS	RUS	SMOTE-ENC
DT	F-Score	**1.5e-04**	**1.5e-04**	**7.3e-06**	**1.2e-06**	1.0e-01
DT	G-Mean	**5.6e-07**	**2.7e-03**	**2.8e-02**	**3.9e-04**	**2.3e-02**
KNN	F-Score	**6.4e-04**	**2.2e-04**	**7.2e-04**	**6.4e-04**	**5.9e-06**
KNN	G-Mean	**1.6e-05**	**9.6e-03**	**6.5e-03**	2.0e-01	**3.5e-03**
LR	F-Score	**4.0e-03**	6.1e-01	**9.2e-03**	**3.6e-04**	5.6e-02
LR	G-Mean	**1.6e-07**	**4.0e-04**	8.6e-01	2.4e-01	**4.7e-03**
RF	F-Score	**1.7e-06**	**2.4e-04**	**8.0e-03**	**1.7e-06**	**8.0e-03**
RF	G-Mean	**3.8e-06**	**8.8e-03**	2.5e-01	**2.3e-02**	**1.7e-03**

6.5.3. Discussion

The results reported in Section 6.5.1 show that G-SMOTENC consistently outperforms the remaining oversampling approaches. Based on the two metrics appropriate for imbalanced learning problems, G-Mean and F-Score, in the average rankings shown in Table 6.3 G-SMOTENC was only outperformed once by a small margin. Unlike the results reported in [214], SMOTE-ENC's performance was rarely superior to SMOTENC's.

The relative difference in the classifiers' performance is better visible in Table 6.4. Using an RF classifier, for example, the impact of using G-SMOTENC compared to no oversampling improves, on average, 13 percentual points on G-mean and nine percentual points using F-Score.

The difference in performance between oversamplers was found to be statistically significant across classifiers and performance metrics in the Friedman test. The p-values of this test are reported in Table 6.5. The superiority of G-SMOTENC was confirmed with the results from the Holm-Bonferroni test shown in Table 6.6. This test showed that G-SMOTENC outperformed with statistical significance the remaining resamplers in 82.5% of the comparisons done.

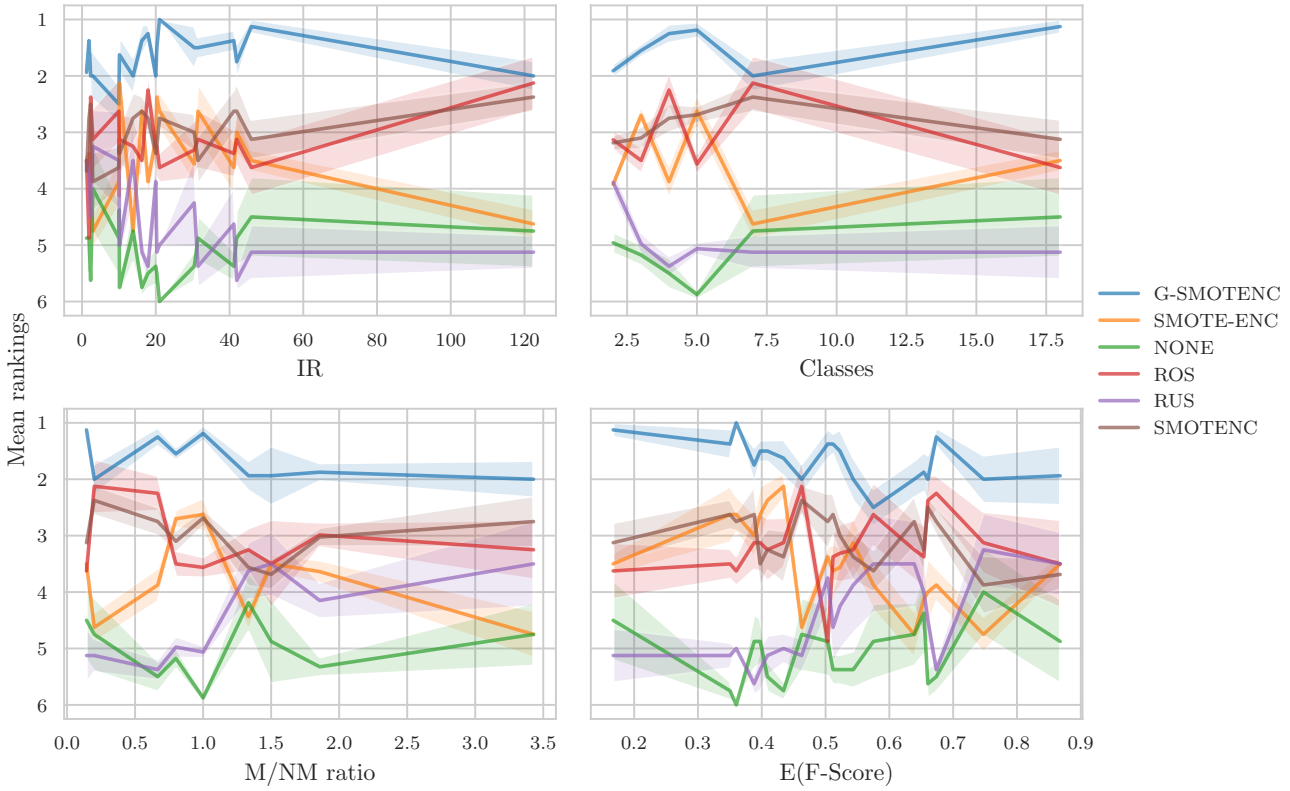


Figure 6.3.: Average ranking of oversamplers over different characteristics of the datasets used in the experiment. Legend: IR — Imbalance Ratio, Classes — Number of classes in the dataset, M/NM ratio — ratio between the number of metric and non-metric features, E(F-Score) — Mean F-Score of dataset across all combinations of classifiers and oversamplers.

The results from this experiment expose some well-known limitations of SMOTE, which become particularly evident with SMOTENC. Specifically, the lack of diversity in the generated data and, on some occasions, the near-duplication of observations discussed in [23] may be a possible explanation for the performance of SMOTENC being comparable to ROS' performance, visible in Figure 6.3. In this figure, three groups of resampling methods with comparable performance are visible: (1) G-SMOTENC, the top-performing method, (2) SMOTENC, ROS and SMOTE-ENC, where SMOTE-ENC has the most inconsistent behavior and (3) RUS and no oversampling, the worst-performing approaches. In addition, G-SMOTENC's superiority seems invariable to the dataset's characteristics, with little overlap with the remaining benchmark methods.

6.6. Conclusion

This paper presented G-SMOTENC, a new oversampling algorithm that combines G-SMOTE and SMOTENC. This oversampling algorithm leverages G-SMOTE's data selection and generation mechanisms into datasets with mixed data types. This was achieved by encoding and generating nominal feature values using SMOTENC's approach. The quality of the data generated with G-SMOTENC was tested over 20 datasets with different imbalance ratios, metric/non-metric feature ratios and number of classes. These results were compared to no oversampling, SMOTENC, Random Oversampling, Random Undersampling

and SMOTE-ENC using a Decision Tree, K-Nearest Neighbors, Logistic Regression and Random Forest as classifiers.

G-SMOTENC can be seen as a drop-in replacement of SMOTENC, since when $\alpha_{trunc} = 1$, $\alpha_{def} = 1$ and $\alpha_{sel} = minority$, SMOTENC is reproduced. G-SMOTENC has three additional hyperparameters that allow for greater customization of the selection and generation mechanisms. However, determining the optimal parameters a priori (*i.e.*, with reduced parameter tuning) is a topic for future work.

The results show that G-SMOTENC performs significantly better when compared to its more popular counterparts (SMOTENC, Random Oversampling and Random Undersampling), as well as a recently proposed oversampling algorithm for mixed data types (SMOTE-ENC). This performance improvement is related to G-SMOTENC's selection mechanism, which finds a safer region for data generation, along with its generation mechanism which increases the diversity of the generated observations compared to SMOTENC. The G-SMOTENC implementation used in this study is available in the open-source Python library "ML-Research" and is fully compatible with the Scikit-Learn ecosystem.

7. Moving Forward

In Chapter 2 we studied the state-of-the-art in data augmentation algorithms, which was necessary to proceed to subsequent steps of the work plan. Based on these findings, in Chapter 3 we used an oversampling method to address a know limitation of imbalanced learning in LULC. An additional limitation we found in oversampling methods was the lack of models capable of addressing datasets with both metric and non-metric features. We plan to address this limitation by modifying a state-of-the-art oversampling method, based on RQ4, as described in Subsection 1.4. This work will be added as a new Chapter between Chapters 2 and 3. At the time of writing, the algorithmic implementation, dataset collection and preprocessing, and experimental results have been completed. The writing of the Chapter is in progress.

The Geometric-SMOTENC oversampler will use the generation mechanism described in [23], while encoding the continuous features, calculating the selected observations' nearest neighbors and generating the categorical feature values for the synthetic data using the method described in [22]. This approach allows the usage of a state-of-the-art oversampling method with categorical features. The experimental results have shown that G-SMOTENC outperforms the oversamplers compatible with this type of data (*i.e.*, Random Oversampling and SMOTENC).

In Chapters 4 and 5 we modify the AL framework as a means to address RQ3. However, the work presented can be further enriched with the application of other methods that leverage information from unlabeled data, specifically semi-supervised and self-supervised learning techniques to form a single framework, the Self-Supervised Semi-Supervised Active Deep Learning (S^4AD-Learning) framework. This work is also under development and is intended to follow Chapter 5. At the time of writing, the algorithmic implementation is in progress.

The last chapter will close the thesis with the conclusions, discussion of the proposed research questions and recommendations for future work based on the results presented in previous chapters.

Bibliography

[1] R. Khatami, G. Mountrakis, and S. V. Stehman, "A meta-analysis of remote sensing research on supervised pixel-based land-cover image classification processes: General guidelines for practitioners and future research," *Remote Sensing of Environment*, vol. 177, pp. 89–100, may 2016.

[2] G. Douzas, F. Bacao, J. Fonseca, and M. Khudinyan, "Imbalanced learning in land cover classification: Improving minority classes' prediction accuracy using the geometric SMOTE algorithm," *Remote Sensing*, vol. 11, p. 3040, dec 2019.

[3] A. P. Tewkesbury, A. J. Comber, N. J. Tate, A. Lamb, and P. F. Fisher, "A critical synthesis of remotely sensed optical image change detection techniques," *Remote Sensing of Environment*, vol. 160, pp. 1–14, 2015.

[4] C. Pelletier, S. Valero, J. Inglada, N. Champion, C. Marais Sicre, and G. Dedieu, "Effect of Training Class Label Noise on Classification Performances for Land Cover Mapping with Satellite Image Time Series," *Remote Sensing*, vol. 9, p. 173, feb 2017.

[5] O. Stromann, A. Nascetti, O. Yousif, and Y. Ban, "Dimensionality Reduction and Feature Selection for Object-Based Land Cover Classification based on Sentinel-1 and Sentinel-2 Time Series Using Google Earth Engine," *Remote Sensing*, vol. 12, no. 1, p. 76, 2020.

[6] F. Alonso-Sarria, C. Valdivieso-Ros, and F. Gomariz-Castillo, "Isolation Forests to Evaluate Class Separability and the Representativeness of Training and Validation Areas in Land Cover Classification," *Remote Sensing*, vol. 11, p. 3000, dec 2019.

[7] W. Feng, W. Huang, H. Ye, and L. Zhao, "Synthetic minority over-sampling technique based rotation forest for the classification of unbalanced hyperspectral data," in *International Geoscience and Remote Sensing Symposium (IGARSS)*, vol. 2018-July, pp. 2651–2654, Institute of Electrical and Electronics Engineers Inc., oct 2018.

[8] O. Siméoni, M. Budnik, Y. Avrithis, and G. Gravier, "Rethinking deep active learning: Using unlabeled data at model training," *Proceedings - International Conference on Pattern Recognition*, pp. 1220–1227, 2020.

[9] A. Fernández, V. López, M. Galar, M. J. del Jesus, and F. Herrera, "Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches," *Knowledge-Based Systems*, vol. 42, pp. 97–110, apr 2013.

[10] Y. Ouali, C. Hudelot, and M. Tami, "An overview of deep semi-supervised learning," *arXiv preprint arXiv:2006.05278*, 2020.

[11] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[12] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21271–21284, 2020.

[13] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[14] S. Budd, E. C. Robinson, and B. Kainz, "A survey on active learning and human-in-the-loop deep learning for medical image analysis," *Medical Image Analysis*, vol. 71, p. 102062, 2021.

[15] S. Behpour, K. M. Kitani, and B. D. Ziebart, "Ada: Adversarial data augmentation for object detection," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1243–1252, IEEE, 2019.

[16] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 13001–13008, 2020.

[17] T. DeVries and G. W. Taylor, "Dataset augmentation in feature space," in *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings*, International Conference on Learning Representations, ICLR, 2 2017.

[18] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[19] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *Plos one*, vol. 16, no. 7, p. e0254841, 2021.

[20] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?," in *2016 international conference on digital image computing: techniques and applications (DICTA)*, pp. 1–6, IEEE, 2016.

[21] O. Kashefi and R. Hwa, "Quantifying the evaluation of heuristic methods for textual data augmentation," in *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, (Online), pp. 200–208, Association for Computational Linguistics, Nov. 2020.

[22] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 6 2002.

[23] G. Douzas and F. Bacao, "Geometric smote a geometrically enhanced drop-in replacement for smote," *Information sciences*, vol. 501, pp. 118–135, 2019.

[24] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*, pp. 878–887, Springer, 2005.

[25] G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE," *Information Sciences*, vol. 465, pp. 1–20, oct 2018.

[26] X. Cao, J. Yao, Z. Xu, and D. Meng, "Hyperspectral Image Classification with Convolutional Neural Network and Active Learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, pp. 4604–4616, jul 2020.

[27] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang, "A survey of deep active learning," *ACM Computing Surveys (CSUR)*, vol. 54, no. 9, pp. 1–40, 2021.

[28] V. K. Shrivastava and M. K. Pradhan, "Hyperspectral remote sensing image classification using active learning," *Studies in Computational Intelligence*, vol. 907, pp. 133–152, 2021.

[29] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A survey of deep active learning," *arXiv preprint arXiv:2009.00236*, 2020.

[30] J. Fonseca, G. Douzas, and F. Bacao, "Improving imbalanced land cover classification with k-means smote: Detecting and oversampling distinctive minority spectral signatures," *Information*, vol. 12, no. 7, p. 266, 2021.

[31] D. Yoo and I. S. Kweon, "Learning loss for active learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 93–102, June 2019.

[32] H. H. Aghdam, A. Gonzalez-Garcia, A. Lopez, and J. Weijer, "Active learning for deep detection neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob, pp. 3671–3679, 2019.

[33] T. Su, S. Zhang, and T. Liu, "Multi-spectral image classification based on an object-based active learning approach," *Remote Sensing*, vol. 12, p. 504, feb 2020.

[34] Y. Sverchkov and M. Craven, "A review of active learning approaches to experimental design for uncovering biological networks," *PLoS Computational Biology*, vol. 13, p. e1005466, 6 2017.

[35] Z. del Rosario, M. Rupp, Y. Kim, E. Antono, and J. Ling, "Assessing the frontier: Active learning, model accuracy, and multi-objective candidate discovery and optimization," *The Journal of Chemical Physics*, vol. 153, p. 024112, 7 2020.

[36] D. Kottke, A. Calma, D. Huseljic, G. Krempl, and B. Sick, "Challenges of reliable, realistic and comparable active learning evaluation," in *CEUR Workshop Proceedings*, vol. 1924, pp. 2–14, sep 2017.

[37] J. Li, X. Huang, and X. Chang, "A label-noise robust active learning sample collection method for multi-temporal urban land-cover classification and change analysis," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 163, no. January, pp. 1–17, 2020.

[38] H. T. Nguyen and A. Smeulders, "Active learning using pre-clustering," in *Proceedings of the twenty-first international conference on Machine learning*, p. 79, 2004.

[39] B. Gu, Z. Zhai, C. Deng, and H. Huang, "Efficient active learning by querying discriminative and representative samples and fully exploiting unlabeled data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 4111–4122, 9 2021.

[40] P. Kumar and A. Gupta, "Active learning query strategies for classification, regression, and clustering: A survey," *Journal of Computer Science and Technology 2020 35:4*, vol. 35, pp. 913–945, 7 2020.

[41] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," *Knowledge and information systems*, vol. 35, no. 2, pp. 249–283, 2013.

[42] A. Samat, P. Gamba, S. Liu, P. Du, and J. Abuduwaili, "Jointly informative and manifold structure representative sampling based active learning for remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, pp. 6803–6817, 11 2016.

[43] S. J. Huang, R. Jin, and Z. H. Zhou, "Active learning by querying informative and representative examples," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 1936–1949, 10 2014.

[44] X. Li, D. Kuang, and C. X. Ling, "Active learning for hierarchical text classification," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7301 LNAI, pp. 14–25, 2012.

[45] D. Ienco, A. Bifet, I. Žliobaite, and B. Pfahringer, "Clustering based active learning for evolving data streams," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8140 LNAI, pp. 79–93, 2013.

[46] K. Brinker, "Incorporating diversity in active learning with support vector machines," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 59–66, 2003.

[47] J. Jia, M. T. Schaub, S. Segarra, and A. R. Benson, "Graph-based semi-supervised & active learning for edge flows," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 761–771, 2019.

[48] B. Settles, "From theories to queries: Active learning in practice," in *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pp. 1–18, JMLR Workshop and Conference Proceedings, 2011.

[49] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *SIGIR'94*, pp. 3–12, Springer, 1994.

[50] T. Luo, K. Kramer, D. B. Goldgof, L. O. Hall, S. Samson, A. Remsen, T. Hopkins, and D. Cohn, "Active learning to recognize multiple types of plankton.," *Journal of Machine Learning Research*, vol. 6, no. 4, 2005.

[51] W. Liu, J. Yang, P. Li, Y. Han, J. Zhao, and H. Shi, "A novel object-based supervised classification method with active learning and random forest for PolSAR imagery," *Remote Sensing*, vol. 10, no. 7, p. 1092, 2018.

[52] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Spectral–spatial classification of hyperspectral data using loopy belief propagation and active learning," *IEEE Transactions on Geoscience and remote sensing*, vol. 51, no. 2, pp. 844–856, 2012.

[53] N. Abe, "Query learning strategies using boosting and bagging," *Proc. of* 15^{th} *Int. Cmf. on Machine Learning (ICML98)*, pp. 1–9, 1998.

[54] P. Melville and R. J. Mooney, "Diverse ensembles for active learning," in *Proceedings of the twenty-first international conference on Machine learning*, p. 74, 2004.

[55] M. Bloodgood, "Support vector machine active learning algorithms with query-by-committee versus closest-to-hyperplane selection," *Proceedings - 12th IEEE International Conference on Semantic Computing, ICSC 2018*, vol. 2018-January, pp. 148–155, 4 2018.

[56] X. Zhou, S. Prasad, and M. Crawford, "Wavelet domain multi-view active learning for hyperspectral image analysis," in *Workshop on Hyperspectral Image and Signal Processing, Evolution in Remote Sensing*, vol. 2014-June, IEEE Computer Society, jun 2014.

[57] G. Fenza, M. Gallo, V. Loia, F. Orciuoli, and E. Herrera-Viedma, "Data set quality in machine learning: Consistency measure based on group decision making," *Applied Soft Computing*, vol. 106, p. 107366, 2021.

[58] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, 2009.

[59] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[60] S. Salman and X. Liu, "Overfitting mechanism and avoidance in deep neural networks," *arXiv preprint arXiv:1901.06566*, 2019.

[61] L. Hu, C. Robinson, and B. Dilkina, "Model generalization in deep learning applications for land cover mapping," *arXiv preprint arXiv:2008.10351*, 2020.

[62] Z. Xie, F. He, S. Fu, I. Sato, D. Tao, and M. Sugiyama, "Artificial neural variability for deep learning: On overfitting, noise memorization, and catastrophic forgetting," *Neural computation*, vol. 33, no. 8, pp. 2163–2192, 2021.

[63] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[64] P. L. Bartlett, A. Montanari, and A. Rakhlin, "Deep learning: a statistical viewpoint," *arXiv preprint arXiv:2103.09177*, 2021.

[65] S. Chun, S. J. Oh, S. Yun, D. Han, J. Choe, and Y. Yoo, "An empirical evaluation on robustness and uncertainty of regularization methods," *arXiv preprint arXiv:2003.03879*, 2020.

[66] D. A. Van Dyk and X.-L. Meng, "The art of data augmentation," *Journal of Computational and Graphical Statistics*, vol. 10, no. 1, pp. 1–50, 2001.

[67] A. J. Ratner, H. R. Ehrenberg, Z. Hussain, J. Dunnmon, and C. Ré, "Learning to compose domain-specific transformations for data augmentation," *Advances in neural information processing systems*, vol. 30, p. 3239, 2017.

[68] U. Meier, D. C. Ciresan, L. M. Gambardella, and J. Schmidhuber, "Better digit recognition with a committee of simple neural nets," in *2011 International Conference on Document Analysis and Recognition*, pp. 1250–1254, IEEE, 2011.

[69] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Handwritten digit recognition with a committee of deep neural nets on gpus," *arXiv preprint arXiv:1103.4487*, 2011.

[70] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. P. Steiner, D. Keysers, J. Uszkoreit, *et al.*, "Mlp-mixer: An all-mlp architecture for vision," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[71] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, G. Izacard, A. Joulin, G. Synnaeve, J. Verbeek, *et al.*, "Resmlp: Feedforward networks for image classification with data-efficient training," *arXiv preprint arXiv:2105.03404*, 2021.

[72] L. Melas-Kyriazi, "Do you even need attention? a stack of feed-forward layers does surprisingly well on imagenet," 2021.

[73] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," *arXiv preprint arXiv:2002.12478*, 2020.

[74] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[75] H. Kaur, H. S. Pannu, and A. K. Malhi, "A systematic review on imbalanced data challenges in machine learning: Applications and solutions," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–36, 2019.

[76] J. Fonseca, G. Douzas, and F. Bacao, "Increasing the Effectiveness of Active Learning: Introducing Artificial Data Generation in Active Learning for Land Use/Land Cover Classification," *Remote Sensing 2021, Vol. 13, Page 2619*, vol. 13, p. 2619, jul 2021.

[77] J. Wang, L. Perez, *et al.*, "The effectiveness of data augmentation in image classification using deep learning," *Convolutional Neural Networks Vis. Recognit*, vol. 11, pp. 1–8, 2017.

[78] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[79] P. Chu, X. Bian, S. Liu, and H. Ling, "Feature space augmentation for long-tailed data," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pp. 694–710, Springer, 2020.

[80] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," *arXiv preprint arXiv:1711.04340*, 2017.

[81] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[82] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[83] N. Paskin, "Toward unique identifiers," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1208–1227, 1999.

[84] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, p. 066111, Dec 2004.

[85] J. K. Pritchard, M. Stephens, and P. Donnelly, "Inference of population structure using multilocus genotype data," *Genetics*, vol. 155, no. 2, pp. 945–959, 2000.

[86] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[87] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010.

[88] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring Network Structure, Dynamics, and Function using NetworkX," in *Proceedings of the 7th Python in Science Conference* (G. Varoquaux, T. Vaught, and J. Millman, eds.), (Pasadena, CA USA), pp. 11–15, 2008.

[89] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: an open source software for exploring and manipulating networks," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 3, 2009.

[90] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[91] P. A. Cicalese, A. Mobiny, P. Yuan, J. Becker, C. Mohan, and H. Van Nguyen, "Stypath: style-transfer data augmentation for robust histology image classification," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 351–361, Springer, 2020.

[92] H. Liu, Z. Dai, D. R. So, and Q. V. Le, "Pay attention to mlps," *arXiv preprint arXiv:2105.08050*, 2021.

[93] M. Drusch, U. Del Bello, S. Carlier, O. Colin, V. Fernandez, F. Gascon, B. Hoersch, C. Isola, P. Laberinti, P. Martimort, A. Meygret, F. Spoto, O. Sy, F. Marchese, and P. Bargellini, "Sentinel-2: ESA's Optical High-Resolution Mission for GMES Operational Services," *Remote Sensing of Environment*, vol. 120, pp. 25–36, may 2012.

[94] S. Fritz, L. See, C. Perger, I. McCallum, C. Schill, D. Schepaschenko, M. Duerauer, M. Karner, C. Dresel, J. C. Laso-Bayas, M. Lesiv, I. Moorthy, C. F. Salk, O. Danylo, T. Sturn, F. Albrecht, L. You, F. Kraxner, and M. Obersteiner, "A global dataset of crowdsourced land cover and land use reference data," *Scientific Data*, vol. 4, pp. 1–8, jun 2017.

[95] M. A. Wulder, N. C. Coops, D. P. Roy, J. C. White, and T. Hermosilla, "Land cover 2.0," *International Journal of Remote Sensing*, vol. 39, no. 12, pp. 4254–4284, 2018.

[96] A. B. Gavade and V. S. Rajpurohit, "Systematic analysis of satellite image-based land cover classification techniques: literature review and challenges," *International Journal of Computers and Applications*, pp. 1–10, feb 2019.

[97] R. Wang, J. Zhang, J.-W. Chen, L. Jiao, and M. Wang, "Imbalanced Learning-based Automatic SAR Images Change Detection by Morphologically Supervised PCA-Net," *IEEE Geoscience and Remote Sensing Letters*, jun 2019.

[98] W. Feng, W. Huang, and W. Bao, "Imbalanced Hyperspectral Image Classification With an Adaptive Ensemble Method Based on SMOTE and Rotation Forest With Differentiated Sampling Rates," *IEEE Geoscience and Remote Sensing Letters*, pp. 1–5, 2019.

[99] L. Abdi and S. Hashemi, "To Combat Multi-Class Imbalanced Problems by Means of Over-Sampling Techniques," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, pp. 238–251, jan 2016.

[100] A. E. Maxwell, T. A. Warner, and F. Fang, "Implementation of machine-learning classification in remote sensing: An applied review," *International Journal of Remote Sensing*, vol. 39, no. 9, pp. 2784–2817, 2018.

[101] J. Luengo, D. García-Gil, S. Ramírez-Gallego, S. García, and F. Herrera, "Imbalanced Data Preprocessing for Big Data," in *Big Data Preprocessing*, pp. 147–160, Cham: Springer International Publishing, 2020.

[102] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez, and F. Herrera, "Big data preprocessing: methods and prospects," *Big Data Analytics*, vol. 1, p. 9, dec 2016.

[103] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data," *Expert Syst. Appl.*, vol. 73, p. 220–239, May 2017.

[104] N. Japkowicz, "Concept-learning in the presence of between-class and within-class imbalances," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2056, pp. 67–77, Springer Verlag, 2001.

[105] T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," *ACM SIGKDD Explorations Newsletter*, vol. 6, pp. 40–49, jun 2004.

[106] R. Blagus and L. Lusa, "Class prediction for high-dimensional class-imbalanced data.," *BMC bioinformatics*, vol. 11, p. 523, oct 2010.

[107] A. Mellor, S. Boukir, A. Haywood, and S. Jones, "Exploring issues of training data imbalance and mislabelling on random forest performance for large area land cover classification using the ensemble margin," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 155–168, jul 2015.

[108] Y. H. Shao, W. J. Chen, J. J. Zhang, Z. Wang, and N. Y. Deng, "An efficient weighted Lagrangian twin support vector machine for imbalanced data classification," *Pattern Recognition*, vol. 47, pp. 3158–3167, sep 2014.

[109] T. Lee, K. B. Lee, and C. O. Kim, "Performance of Machine Learning Algorithms for Class-Imbalanced Process Fault Detection Problems," *IEEE Transactions on Semiconductor Manufacturing*, vol. 29, pp. 436–445, nov 2016.

[110] C. Huang, Y. Li, C. C. Loy, and X. Tang, "Learning deep representation for imbalanced classification," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 5375–5384, 2016.

[111] Y. Cui, M. Jia, T. Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 9260–9269, 2019.

[112] Q. Dong, S. Gong, and X. Zhu, "Class Rectification Hard Mining for Imbalanced Deep Learning," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 1869–1878, 2017.

[113] A. Sharififar, F. Sarmadian, and B. Minasny, "Mapping imbalanced soil classes using Markov chain random fields models treated with data resampling technique," *Computers and Electronics in Agriculture*, vol. 159, pp. 110–118, apr 2019.

[114] K. O. Hounkpatin, K. Schmidt, F. Stumpf, G. Forkuor, T. Behrens, T. Scholten, W. Amelung, and G. Welp, "Predicting reference soil groups using legacy data: A data pruning and Random Forest approach for tropical environment (Dano catchment, Burkina Faso)," *Scientific Reports*, vol. 8, pp. 1–16, dec 2018.

[115] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, pp. 221–232, nov 2016.

[116] M. P. Ferreira, F. H. Wagner, L. E. Aragão, Y. E. Shimabukuro, and C. R. de Souza Filho, "Tree species classification in tropical forests using visible to shortwave infrared WorldView-3 images and texture analysis," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 149, pp. 119–131, mar 2019.

[117] S. E. Jozdani, B. A. Johnson, and D. Chen, "Comparing Deep Neural Networks, Ensemble Classifiers, and Support Vector Machine Algorithms for Object-Based Urban Land Use/Land Cover Classification," *Remote Sensing*, vol. 11, p. 1713, jul 2019.

[118] C. Bogner, B. Seo, D. Rohner, and B. Reineking, "Classification of rare land cover types: Distinguishing annual and perennial crops in an agricultural catchment in South Korea," *PLoS ONE*, vol. 13, jan 2018.

[119] M. Zhu, B. Wu, Y. N. He, and Y. Q. He, "Land Cover Classification Using High Resolution Satellite Image Based On Deep Learning," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-3/W10, pp. 685–690, 2020.

[120] T. W. Cenggoro, S. M. Isa, G. P. Kusuma, and B. Pardamean, "Classification of imbalanced land-use/land-cover data using variational semi-supervised learning," in *Proceedings - 2017 International Conference on Innovative and Creative Information Technology: Computational Intelligence and IoT, ICITech 2017*, vol. 2018-Janua, pp. 1–6, IEEE, nov 2018.

[121] L. Ma and S. Fan, "CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests," *BMC Bioinformatics*, vol. 18, p. 169, mar 2017.

[122] G. Douzas and F. Bacao, "Self-Organizing Map Oversampling (SOMO) for imbalanced data set learning," *Expert Systems with Applications*, vol. 82, pp. 40–52, oct 2017.

[123] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322–1328, IEEE, jun 2008.

[124] R. C. Holte, L. Acker, B. W. Porter, *et al.*, "Concept learning and the problem of small disjuncts.," in *IJCAI*, vol. 89, pp. 813–818, Citeseer, 1989.

[125] M. S. Santos, P. H. Abreu, P. J. García-Laencina, A. Simão, and A. Carvalho, "A new cluster-based oversampling method for improving survival prediction of hepatocellular carcinoma patients," *Journal of Biomedical Informatics*, vol. 58, pp. 49–59, dec 2015.

[126] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, "220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3," *Purdue University Research Repository*, sep 2015.

[127] J. A. Nelder and R. W. Wedderburn, "Generalized linear models," *Journal of the Royal Statistical Society: Series A (General)*, vol. 135, no. 3, pp. 370–384, 1972.

[128] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[129] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[130] P. Olofsson, G. M. Foody, S. V. Stehman, and C. E. Woodcock, "Making better use of accuracy data in land change studies: Estimating accuracy and area and quantifying uncertainty using stratified estimation," *Remote Sensing of Environment*, vol. 129, pp. 122–131, feb 2013.

[131] R. G. Pontius and M. Millones, "Death to Kappa: Birth of quantity disagreement and allocation disagreement for accuracy assessment," 2011.

[132] L. A. Jeni, J. F. Cohn, and F. De La Torre, "Facing imbalanced data - Recommendations for the use of performance metrics," in *Proceedings - 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, ACII 2013*, pp. 245–251, 2013.

[133] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[134] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.

[135] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[136] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the american statistical association*, vol. 32, no. 200, pp. 675–701, 1937.

[137] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, p. 80, dec 1945.

[138] S. Nagai, K. N. Nasahara, T. K. Akitsu, T. M. Saitoh, and H. Muraoka, "Importance of the Collection of Abundant Ground-Truth Data for Accurate Detection of Spatial and Temporal Variability of Vegetation by Satellite Remote Sensing," in *Biogeochemical Cycles: Ecological Drivers and Environmental Impact*, pp. 223–244, American Geophysical Union (AGU), feb 2020.

[139] Y. Huang, Z. xin CHEN, T. YU, X. zhi HUANG, and X. fa GU, "Agricultural remote sensing big data: Management and applications," *Journal of Integrative Agriculture*, vol. 17, pp. 1915–1931, sep 2018.

[140] X. Wang and H. Xie, "A review on applications of remote sensing and geographic information systems (GIS) in water resources and flood risk management," *Water (Switzerland)*, vol. 10, p. 608, may 2018.

[141] H. Costa, P. Benevides, F. Marcelino, and M. Caetano, "Introducing automatic satellite image processing into land cover mapping by photo-interpretation of airborne data," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 29–34, 2020.

[142] E. F. Vermote, S. Skakun, I. Becker-Reshef, and K. Saito, "Remote sensing of coconut trees in tonga using very high spatial resolution worldview-3 data," *Remote Sensing*, vol. 12, no. 19, p. 3113, 2020.

[143] D. Costantino, M. Pepe, G. Dardanelli, and V. Baiocchi, "Using Optical Satellite and Aerial Imagery for Automatic Coastline Mapping," *Geographia Technica*, pp. 171–190, sep 2020.

[144] V. Růžička, S. D'Aronco, J. D. Wegner, and K. Schindler, "Deep active learning in remote sensing for data efficient change detection," *arXiv preprint arXiv:2008.11201*, 2020.

[145] S.-J. Liu, H. Luo, and Q. Shi, "Active Ensemble Deep Learning for Polarimetric Synthetic Aperture Radar Image Classification," *IEEE Geoscience and Remote Sensing Letters*, pp. 1–5, 2020.

[146] E. Pasolli, H. L. Yang, and M. M. Crawford, "Active-metric learning for classification of remotely sensed hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, pp. 1925–1939, apr 2016.

[147] G. Cawley, "Baseline Methods for Active Learning," *Proceedings of Active Learning and Experimental Design workshop In conjunction with AISTATS*, vol. 16, pp. 47–57, apr 2011.

[148] X. Li and Y. Guo, "Active learning with multi-label svm classification," in *In IJCAI*, pp. 1479–1485, 08 2013.

[149] D. Tuia, E. Pasolli, and W. J. Emery, "Using active learning to adapt remote sensing image classifiers," *Remote Sensing of Environment*, vol. 115, no. 9, pp. 2232–2242, 2011.

[150] X. Wu, W. Li, D. Hong, J. Tian, R. Tao, and Q. Du, "Vehicle detection of multi-source remote sensing data using active fine-tuning network," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 167, pp. 39–53, sep 2020.

[151] H. Bi, F. Xu, Z. Wei, Y. Xue, and Z. Xu, "An Active Deep Learning Approach for Minimally Supervised PolSAR Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, pp. 9378–9395, nov 2019.

[152] S. K. Roy, G. Krishna, S. R. Dubey, and B. B. Chaudhuri, "HybridSN: Exploring 3-D-2-D CNN Feature Hierarchy for Hyperspectral Image Classification," *IEEE Geoscience and Remote Sensing Letters*, pp. 1–5, jun 2019.

[153] I. Muslea, S. Minton, and C. A. Knoblock, "Active learning with multiple views," *Journal of Artificial Intelligence Research*, vol. 27, pp. 203–233, oct 2006.

[154] W. Di and M. M. Crawford, "View generation for multiview maximum disagreement based active learning for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, pp. 1942–1954, may 2012.

[155] Z. Zhang, E. Pasolli, H. L. Yang, and M. M. Crawford, "Multimetric Active Learning for Classification of Remote Sensing Data," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, pp. 1007–1011, jul 2016.

[156] D. Tuia, F. Ratle, F. Pacifici, M. F. Kanevski, and W. J. Emery, "Active learning methods for remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, pp. 2218–2232, 7 2009.

[157] L. Copa, D. Tuia, M. Volpi, and M. Kanevski, "Unbiased query-by-bagging active learning for VHR image classification," in *Image and Signal Processing for Remote Sensing XVI* (L. Bruzzone, ed.), vol. 7830, p. 78300K, SPIE, oct 2010.

[158] B. Demir, C. Persello, and L. Bruzzone, "Batch-mode active-learning methods for the interactive classification of remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, pp. 1014–1031, mar 2011.

[159] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Hyperspectral image segmentation using a new bayesian approach with active learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, pp. 3947–3960, oct 2011.

[160] T. Luo, K. Kramer, D. Goldgof, L. O. Hall, S. Samson, A. Remsen, and T. Hopkins, "Learning to recognize plankton," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, pp. 888–893, 2003.

[161] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Spectral-spatial classification of hyperspectral data using loopy belief propagation and active learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 2, pp. 844–856, 2013.

[162] S. Ertekin, J. Huang, and C. L. Giles, "Active learning for class imbalance problem," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'07*, (New York, New York, USA), pp. 823–824, ACM Press, 2007.

[163] T. K. Ho, "Random decision forests," in *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, (USA), p. 278, IEEE Computer Society, 1995.

[164] M. Kubat, S. Matwin, *et al.*, "Addressing the curse of imbalanced training sets: one-sided selection," in *Icml*, vol. 97, pp. 179–186, Citeseer, 1997.

[165] T. Reitmaier and B. Sick, "Let us know your decision: Pool-based active training of a generative classifier with the selection strategy 4ds," *Information Sciences*, vol. 230, pp. 106–131, 5 2013.

[166] J.-F. Kagy, T. Kayadelen, J. Ma, A. Rostamizadeh, and J. Strnadova, "The practical challenges of active learning: Lessons learned from live experimentation," *arXiv preprint arXiv:1907.00038*, 6 2019.

[167] V. Nath, D. Yang, B. A. Landman, D. Xu, and H. R. Roth, "Diminishing uncertainty within the training pool: Active learning for medical image segmentation," *IEEE Transactions on Medical Imaging*, vol. 40, no. 10, pp. 2534–2547, 2021.

[168] Y. Li, J. Yin, and L. Chen, "Seal: Semisupervised adversarial active learning on attributed graphs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 3136–3147, 2021.

[169] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020.

[170] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," in *International Conference on Learning Representations*, 2018.

[171] Y. Leng, X. Xu, and G. Qi, "Combining active learning and semi-supervised learning to construct svm classifier," *Knowledge-Based Systems*, vol. 44, pp. 121–131, 2013.

[172] H. Yu, X. Yang, S. Zheng, and C. Sun, "Active learning from imbalanced data: A solution of online weighted extreme learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, pp. 1088–1103, 4 2019.

[173] H. Zhang, W. Liu, and Q. Liu, "Reinforcement online active learning ensemble for drifting imbalanced data streams," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[174] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, 2013.

[175] J. Qin, C. Wang, Q. Zou, Y. Sun, and B. Chen, "Active learning with extreme learning machine for online imbalanced multiclass classification," *Knowledge-Based Systems*, vol. 231, p. 107385, 2021.

[176] A. Tharwat and W. Schenck, "Balancing exploration and exploitation: A novel active learner for imbalanced data," *Knowledge-Based Systems*, vol. 210, p. 106500, 2020.

[177] Y.-Y. Kim, K. Song, J. Jang, and I.-c. Moon, "Lada: Look-ahead data acquisition via augmentation for deep active learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[178] J. Katz-Samuels, J. Zhang, L. Jain, and K. Jamieson, "Improved algorithms for agnostic pool-based active classification," in *International Conference on Machine Learning*, pp. 5334–5344, PMLR, 2021.

[179] T. M. Hospedales, S. Gong, and T. Xiang, "Finding rare classes: Active learning with generative and discriminative models," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 2, pp. 374–386, 2011.

[180] Y. Li, X. Wang, Z. Shi, R. Zhang, J. Xue, and Z. Wang, "Boosting training for pdf malware classifier via active learning," *International Journal of Intelligent Systems*, vol. 37, no. 4, pp. 2803–2821, 2022.

[181] J. Li, X. Huang, and X. Chang, "A label-noise robust active learning sample collection method for multi-temporal urban land-cover classification and change analysis," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 163, pp. 1–17, 2020.

[182] C. Su, Z. Yan, and G. Yu, "Cost-effective multi-instance multilabel active learning," *International Journal of Intelligent Systems*, vol. 36, no. 12, pp. 7177–7203, 2021.

[183] X. Wu, W. Zheng, X. Xia, and D. Lo, "Data quality matters: A case study on data label correctness for security bug report prediction," *IEEE Transactions on Software Engineering*, 2021.

[184] W. Zhang, Z. Wang, and X. Li, "Blockchain-based decentralized federated transfer learning methodology for collaborative machinery fault diagnosis," *Reliability Engineering & System Safety*, vol. 229, p. 108885, 2023.

[185] W. Zhang, X. Li, H. Ma, Z. Luo, and X. Li, "Open-set domain adaptation in machinery fault diagnostics using instance-level weighted adversarial learning," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7445–7455, 2021.

[186] M. He and D. He, "Deep learning based approach for bearing fault diagnosis," *IEEE Transactions on Industry Applications*, vol. 53, no. 3, pp. 3057–3065, 2017.

[187] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, and L. Guibas, "Grass: Generative recursive autoencoders for shape structures," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–14, 2017.

[188] W. Zheng, X. Liu, and L. Yin, "Sentence representation method based on multi-layer semantic network," *Applied Sciences*, vol. 11, no. 3, p. 1316, 2021.

[189] K. Zhang, Z. Wang, G. Chen, L. Zhang, Y. Yang, C. Yao, J. Wang, and J. Yao, "Training effective deep reinforcement learning agents for real-time life-cycle production optimization," *Journal of Petroleum Science and Engineering*, vol. 208, p. 109766, 2022.

[190] T. Tran, T.-T. Do, I. Reid, and G. Carneiro, "Bayesian generative active deep learning," in *International Conference on Machine Learning*, pp. 6295–6304, PMLR, 2019.

[191] S. Sinha, S. Ebrahimi, and T. Darrell, "Variational adversarial active learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5972–5981, 2019.

[192] K. Kim, D. Park, K. I. Kim, and S. Y. Chun, "Task-aware variational adversarial active learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8166–8175, 2021.

[193] Y. Ma, S. Lu, E. Xu, T. Yu, and L. Zhou, "Combining active learning and data augmentation for image classification," in *Proceedings of the 2020 3rd International Conference on Big Data Technologies*, pp. 58–62, 2020.

[194] H. Quteineh, S. Samothrakis, and R. Sutcliffe, "Textual data augmentation for efficient active learning on tiny datasets," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7400–7410, 2020.

[195] Q. Li, Z. Huang, Y. Dou, and Z. Zhang, "A framework of data augmentation while active learning for chinese named entity recognition," in *International Conference on Knowledge Science, Engineering and Management*, pp. 88–100, Springer, 2021.

[196] C. Wu, *The decision tree approach to classification.* Purdue University, 1975.

[197] M. Fatourechi, R. K. Ward, S. G. Mason, J. Huggins, A. Schloegl, and G. E. Birch, "Comparison of evaluation metrics in classification applications with imbalanced datasets," in *2008 seventh international conference on machine learning and applications*, pp. 777–782, IEEE, 2008.

[198] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, pp. 65–70, 1979.

[199] S. Tyagi and S. Mittal, "Sampling approaches for imbalanced data classification problem in machine learning," in *Proceedings of ICRIC 2019*, pp. 209–221, Springer, 2020.

[200] P. Vuttipittayamongkol, E. Elyan, and A. Petrovski, "On the class overlap problem in imbalanced data classification," *Knowledge-based systems*, vol. 212, p. 106631, 2021.

[201] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information sciences*, vol. 250, pp. 113–141, 2013.

[202] S. Das, S. Datta, and B. B. Chaudhuri, "Handling data irregularities in classification: Foundations, trends, and future challenges," *Pattern Recognition*, vol. 81, pp. 674–693, 2018.

[203] H. Kaur, H. S. Pannu, and A. K. Malhi, "A systematic review on imbalanced data challenges in machine learning: Applications and solutions," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–36, 2019.

[204] A. Fernández, V. LóPez, M. Galar, M. J. Del Jesus, and F. Herrera, "Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches," *Knowledge-based systems*, vol. 42, pp. 97–110, 2013.

[205] A. N. Tarekegn, M. Giacobini, and K. Michalak, "A review of methods for imbalanced multi-label classification," *Pattern Recognition*, vol. 118, p. 107965, 2021.

[206] J. Lumijärvi, J. Laurikkala, and M. Juhola, "A comparison of different heterogeneous proximity functions and euclidean distance," in *MEDINFO 2004*, pp. 1362–1366, IOS Press, 2004.

[207] A. Fernández, S. Garcia, F. Herrera, and N. V. Chawla, "Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary," *Journal of artificial intelligence research*, vol. 61, pp. 863–905, 2018.

[208] X. Liang, A. Jiang, T. Li, Y. Xue, and G. Wang, "Lr-smote — an improved unbalanced data set oversampling based on k-means and svm," *Knowledge-Based Systems*, vol. 196, p. 105845, 2020.

[209] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Pacific-Asia conference on knowledge discovery and data mining*, pp. 475–482, Springer, 2009.

[210] A. Salazar, L. Vergara, and G. Safont, "Generative adversarial networks and markov random fields for oversampling very small training sets," *Expert Systems with Applications*, vol. 163, p. 113819, 2021.

[211] A. Koivu, M. Sairanen, A. Airola, and T. Pahikkala, "Synthetic minority oversampling of vital statistics data with generative adversarial networks," *Journal of the American Medical Informatics Association*, vol. 27, no. 11, pp. 1667–1674, 2020.

[212] W. Jo and D. Kim, "Obgan: Minority oversampling near borderline with generative adversarial networks," *Expert Systems with Applications*, vol. 197, p. 116694, 2022.

[213] L. Gonog and Y. Zhou, "A review: generative adversarial networks," in *2019 14th IEEE conference on industrial electronics and applications (ICIEA)*, pp. 505–510, IEEE, 2019.

[214] M. Mukherjee and M. Khushi, "Smote-enc: A novel smote-based method to generate synthetic data for nominal and continuous features," *Applied System Innovation*, vol. 4, no. 1, p. 18, 2021.

[215] K. Ambai and H. Fujita, "Multivariate normal distribution based over-sampling for numerical and categorical features," in *Advancing Technology Industrialization Through Intelligent Software Methodologies, Tools and Techniques: Proceedings of the 18th International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques (SoMeT)*, vol. 318, p. 107, 2019.

[216] K. Ambai and H. Fujita, "Mndo: Multivariate normal distribution based over-sampling for binary classification.," in *Advancing Technology Industrialization Through Intelligent Software Methodologies, Tools and Techniques: Proceedings of the 17th International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques (SoMeT)*, pp. 425–438, 2018.

[217] S. Park and H. Park, "Combined oversampling and undersampling method based on slow-start algorithm for imbalanced network traffic," *Computing*, vol. 103, no. 3, pp. 401–424, 2021.

[218] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[219] A. Bansal and A. Jain, "Analysis of focussed under-sampling techniques with machine learning classifiers," in *2021 IEEE/ACIS 19th International Conference on Software Engineering Research, Management and Applications (SERA)*, pp. 91–96, IEEE, 2021.

[220] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *International journal of pattern recognition and artificial intelligence*, vol. 23, no. 04, pp. 687–719, 2009.

[221] N. Rout, D. Mishra, and M. K. Mallick, "Handling imbalanced data: a survey," in *International proceedings on advances in soft computing, intelligent systems and applications*, pp. 431–443, Springer, 2018.

[222] N. Japkowicz, "Assessment metrics for imbalanced learning," *Imbalanced learning: Foundations, algorithms, and applications*, pp. 187–206, 2013.

A. Improving Imbalanced Land Cover Classification with K-means SMOTE: Detecting and Oversampling Distinctive Minority Spectral Signatures

Dataset	Classifier	Metric	NONE	ROS	SMOTE	B-SMOTE	K-SMOTE
Botswana	LR	Accuracy	0.920	0.917	0.920	0.921	**0.927**
Botswana	LR	F-score	0.913	0.909	0.913	0.914	**0.921**
Botswana	LR	G-mean	0.952	0.950	0.952	0.952	**0.956**
Botswana	KNN	Accuracy	0.875	0.862	0.881	0.869	**0.889**
Botswana	KNN	F-score	0.859	0.850	0.873	0.859	**0.879**
Botswana	KNN	G-mean	0.924	0.918	0.930	0.923	**0.933**
Botswana	RF	Accuracy	0.873	0.884	0.877	0.877	**0.890**
Botswana	RF	F-score	0.865	0.877	0.872	0.870	**0.883**
Botswana	RF	G-mean	0.925	0.933	0.929	0.928	**0.936**
PC	LR	Accuracy	0.954	0.955	0.955	0.950	**0.956**
PC	LR	F-score	0.944	0.947	0.947	0.941	**0.948**
PC	LR	G-mean	0.968	0.972	0.972	0.966	**0.973**
PC	KNN	Accuracy	**0.926**	0.920	0.923	0.924	**0.926**
PC	KNN	F-score	**0.915**	0.909	0.913	0.913	**0.915**
PC	KNN	G-mean	0.953	0.955	**0.957**	0.954	**0.957**
PC	RF	Accuracy	0.938	0.941	0.940	0.938	**0.942**
PC	RF	F-score	0.928	0.932	0.931	0.928	**0.933**
PC	RF	G-mean	0.959	0.964	**0.965**	0.961	**0.965**
KSC	LR	Accuracy	0.904	0.905	0.905	0.899	**0.909**
KSC	LR	F-score	0.868	0.873	0.874	0.862	**0.877**
KSC	LR	G-mean	0.928	0.932	0.932	0.924	**0.934**
KSC	KNN	Accuracy	0.855	0.859	0.862	0.857	**0.865**
KSC	KNN	F-score	0.808	0.819	**0.827**	0.810	0.826
KSC	KNN	G-mean	0.893	0.901	**0.906**	0.895	0.905
KSC	RF	Accuracy	0.860	0.859	0.863	0.859	**0.868**
KSC	RF	F-score	0.817	0.815	0.826	0.816	**0.832**
KSC	RF	G-mean	0.898	0.899	0.905	0.898	**0.907**
SA	LR	Accuracy	0.979	0.981	0.983	0.979	**0.984**
SA	LR	F-score	0.976	0.979	**0.982**	0.977	**0.982**
SA	LR	G-mean	0.985	0.988	**0.990**	0.987	0.989
SA	KNN	Accuracy	0.987	0.979	0.982	0.983	**0.988**
SA	KNN	F-score	0.986	0.979	0.981	0.982	**0.987**
SA	KNN	G-mean	0.992	0.989	0.990	0.991	**0.993**
SA	RF	Accuracy	0.980	0.983	0.984	0.979	**0.985**
SA	RF	F-score	0.979	0.982	0.983	0.978	**0.984**
SA	RF	G-mean	0.987	0.988	0.989	0.986	**0.990**

Dataset	Classifier	Metric	NONE	ROS	SMOTE	B-SMOTE	K-SMOTE
PU	LR	Accuracy	**0.905**	0.897	0.897	0.891	0.904
PU	LR	F-score	0.890	0.894	0.894	0.888	**0.898**
PU	LR	G-mean	0.932	0.947	0.947	0.942	**0.949**
PU	KNN	Accuracy	**0.895**	0.867	0.865	0.873	**0.895**
PU	KNN	F-score	**0.891**	0.868	0.868	0.874	**0.891**
PU	KNN	G-mean	0.940	0.935	0.936	0.936	**0.941**
PU	RF	Accuracy	**0.912**	0.908	0.907	0.908	0.911
PU	RF	F-score	**0.909**	0.906	0.906	0.908	**0.909**
PU	RF	G-mean	0.946	0.946	0.948	0.948	**0.949**
Salinas	LR	Accuracy	**0.990**	**0.990**	0.989	**0.990**	**0.990**
Salinas	LR	F-score	0.985	**0.986**	0.985	0.985	**0.986**
Salinas	LR	G-mean	0.992	**0.993**	0.992	0.992	**0.993**
Salinas	KNN	Accuracy	**0.970**	0.967	0.969	0.967	**0.970**
Salinas	KNN	F-score	0.959	0.957	**0.960**	0.957	**0.960**
Salinas	KNN	G-mean	0.977	0.978	**0.981**	0.976	**0.981**
Salinas	RF	Accuracy	0.984	0.983	0.983	0.983	**0.985**
Salinas	RF	F-score	0.979	0.979	0.977	0.978	**0.980**
Salinas	RF	G-mean	0.989	0.989	0.989	0.989	**0.990**
IP	LR	Accuracy	0.687	0.681	0.680	0.678	**0.692**
IP	LR	F-score	0.662	0.663	0.659	0.659	**0.674**
IP	LR	G-mean	0.798	0.801	0.798	0.797	**0.807**
IP	KNN	Accuracy	**0.644**	0.602	0.589	0.557	0.632
IP	KNN	F-score	0.593	0.591	0.603	0.560	**0.604**
IP	KNN	G-mean	0.757	0.764	**0.782**	0.751	0.781
IP	RF	Accuracy	0.742	0.747	0.747	0.740	**0.752**
IP	RF	F-score	0.673	0.704	0.713	0.701	**0.714**
IP	RF	G-mean	0.806	0.826	0.835	0.831	**0.838**

Table A.1.: Mean cross-validation scores for each dataset. Legend: IP - Indian Pines, KSC - Kennedy Space Center, PC - Pavia Center, PU - Pavia University, SA - Salinas A.

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa