Improving Active Learning Performance Through the Use of Data Augmentation

Joao Fonseca[1*], Fernando Bacao[1]

[1]NOVA Information Management School, Universidade Nova de Lisboa

*Corresponding Author

Postal Address: NOVA Information Management School, Campus de Campolide, 1070–312 Lisboa, Portugal

Telephone: +351 21 382 8610

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

1 Introduction

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

The importance of training robust ML models with minimal data requirements is substantially increasing [1, 2, 3]. Although the growing amount of valuable data sources and formats being developed and explored is affecting various domains [4], 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) [1]. 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 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 [5].

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 [6]. 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 [7]. 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 [8].

Several studies have pointed out the limitations of AL within an Imbalanced Learning context [9, 10]. 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 [9, 11, 12]. However, classifier or query function-level modifications (See Section 2.1) have limited applicability since a universally good AL strategy has not yet been found [7]. Other methods address imbalanced learning by weighing the observations as the function of the observation's class imbalance ratio [13]. Alternatively, other techniques reduce the imbalanced learning bias by combining Informative and Representative-based query approaches (see Section 2.1) [14]. Another approach to deal with imbalanced data and data scarcity, in general, is generating synthetic data [15]. 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 [16]. However, most recent studies improve the AL performance by modifying the design/choice of the classifier and query functions used.

Recently, data augmentation 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.

1.1 Motivation and contributions

The usage of data augmentation in AL is not new. The literature found on the topic (see Section 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:

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) [17] as a data augmentation method with an optimized generation policy (explained in Section 2.2). 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 2 introduces relevant topics discussed in the paper and describes the related work. Section 3 elucidates the proposed method. Section 4 details the methodology of the study's experiment. Section 5 presents the results obtained from the experiment, as well as a discussion of these results. Section 6 presents the conclusions drawn from this study.

2 Background

In this section we describe the AL problem, data augmentation techniques, and review the literature that combines AL with data augmentation. Table 1 describes the notations used throughout the rest of this study.

Table 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 $\mathcal{D}^{t+1}_{lab}$.
T	Iteration budget.
n	Annotation budget per iteration.

2.1 Active Learning

This paper focuses on pool-based AL methods as defined in [18]. 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 $\mathcal{D}^{t+1}_{pool} = \mathcal{D}^t_{pool} \setminus \mathcal{D}^t_{new}$ [19, 2]. This process is shown in Figure 1. Before the start of the iterative process,

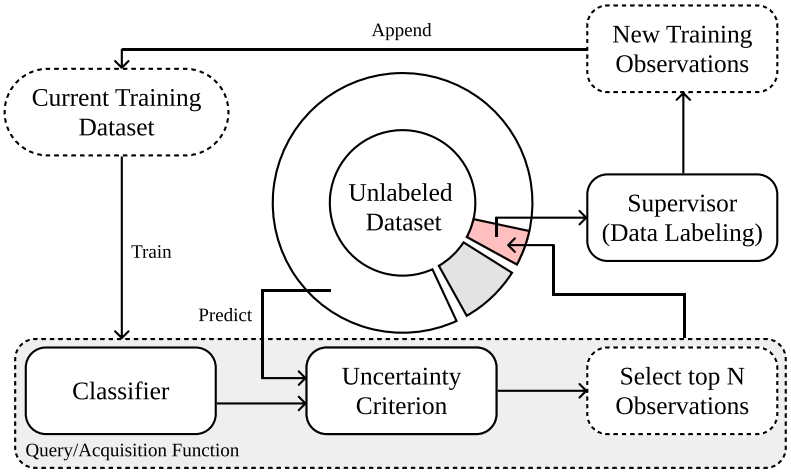

Figure 1: Diagram depicting a typical AL iteration. In the first iteration, the training set collected during the initialization process becomes the "Current Training Dataset".

assuming $\mathcal{D}_{lab}^{t=0} = \emptyset$, the data used to populate $\mathcal{D}_{lab}^{t=1}$ is typically collected randomly from $\mathcal{D} = \mathcal{D}_{pool}^{t=0}$ and is labeled by a supervisor [20, 21, 22].

Research focused on AL has typically been focused on the specification of f_{acq} [23] and domain-specific applications, such as malware detection [24] or Land Use/Land Cover classification [25]. Acquisition functions can be divided into two different categories [26, 27]:

1. Informative-based. These strategies use the classifier's output to assess the importance of each observation towards the performance of the classifier [28].

2. Representative-based. These strategies estimate the optimal set of observations that will optimize the classifier's performance [27].

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 [21] 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 [29]. Another related example is machinery fault diagnostics, where the quality and quantity of the data collected is often recognized as a bottleneck [30]. In this case, ML-based techniques frequently leverage unlabeled data to improve the classification performance [31] and rely on manual data acquisition [32]. 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 [33] or, in the case of text data, semantic

representation networks [34]. 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 [35].

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 [36]. The production of artificial data may be done via the introduction of perturbations on the input [37], feature [38], or output space [36]. Data Augmentation methods may be divided into two categories [39]:

1. Heuristic approaches attempt to generate new and relevant observations by applying a predefined procedure, usually incorporating some degree of randomness [40]. 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 [38]. 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 [41], Translation, Cropping and Flipping are examples of image data-specific augmentation methods. Other methods, such as autoencoders, may be considered domain agnostic.

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 [20]. 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) [42] 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 [16], 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 [43] 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 2.1. The Task-aware VAAL model [44] extends the VAAL model by introducing a ranker, which consists of the Learning Loss module introduced in [21]. 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 [45], 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 [46, 47]. However, these methods were designed for specific applications within that domain and are not necessarily transferable to other domains or tasks.

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 [20].

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 [20] from oversampling techniques to any other data augmentation mechanism and/or policy.

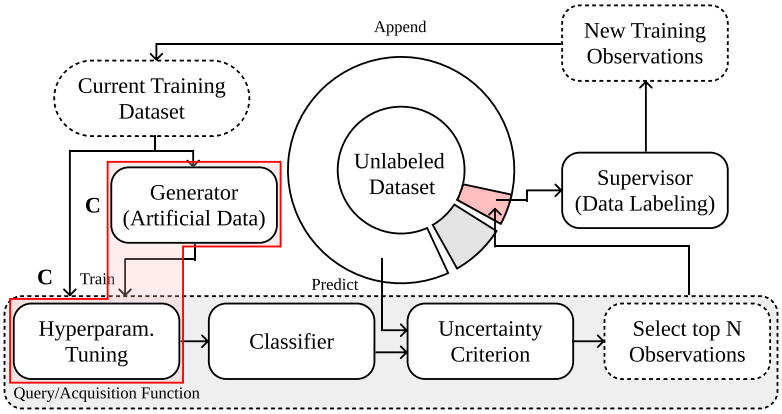

Figure 2: Diagram depicting the proposed AL iteration. The proposed modifications are comprised within the red polygon and marked with a boldface "C."

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 2. 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 [20]. This factor allows the direct comparison of the two frameworks and establishes a causality of the performance variations to the data generation mechanism (*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) a hyperparameter tuning module to estimate the optimal data augmentation policy. This framework was designed to be task-agnostic. Specifically, any data augmentation method

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 $\mathcal{D}_{new}$
    $\mathcal{D}_{pool} \leftarrow \mathcal{D}_{pool} \setminus \mathcal{D}_{new}$
    $\mathcal{D}_{lab} \leftarrow \mathcal{D}_{lab} \cup \mathcal{D}_{new}$

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**
      $x_{center} \leftarrow DataSelection(\mathcal{D}'_{lab}, f_{acq}, f_c^{t-1})$
      $x_{gen} \leftarrow GSMOTE(x_{center}, \mathcal{D}^t_{lab}, \alpha_{trunc}, \alpha_{def})$
      $\mathcal{D}'_{aug} \leftarrow \mathcal{D}'_{aug} \cup \{x_{gen}\}$
  $\mathcal{D}_{aug} \leftarrow \mathcal{D}^{min}_{aug} \cup \mathcal{D}^{maj}_{aug}$
  $\mathcal{D}^t_{train} \leftarrow \mathcal{D}^t_{lab} \cup \mathcal{D}_{aug}$

(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 [21].

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 4.1. The classifiers used in the experiment are defined in Subsection 4.2. The metrics chosen to measure AL performance and overall classification performance are defined in Subsection 4.3. The experimental procedure is described in Subsection 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).

4.1 Datasets

The datasets used to test the proposed method are publicly available in open data repositories. Specifically, they were retrieved from OpenML and the UCI Machine Learning Repository. 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 across all datasets. Table 2 describes the key properties of the 15 preprocessed datasets where the experimental procedure was applied.

Table 2: Description of the datasets collected after data preprocessing. The sampling strategy is similar across datasets. Legend: (IR) Imbalance Ratio

Dataset	Feat.	Inst.	Min. inst.	Maj. inst.	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

The data preprocessing pipeline is depicted as a flowchart in Figure 3. 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

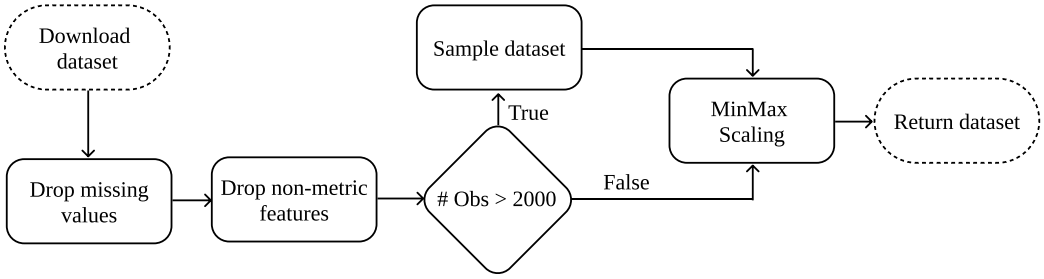

Figure 3: Data preprocessing pipeline.

availability).

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) [48], a K-nearest neighbors classifier (KNN) [49], a Random Forest Classifier (RF) [50] and a Logistic Regression (LR) [51]. 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 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 [20]. 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 3.

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 [52]. 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 [53]. 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 [52, 54], 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 [54]. 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 [52]:

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

- 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 [55]. 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) [56]. 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.

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 [57]. Since the dataset is already labeled, the annotation process is done at zero cost in this scenario. Figure 4 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.

The process shown in Figure 4 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 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 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 [58]. Since mean performance scores are still intuitive

Table 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

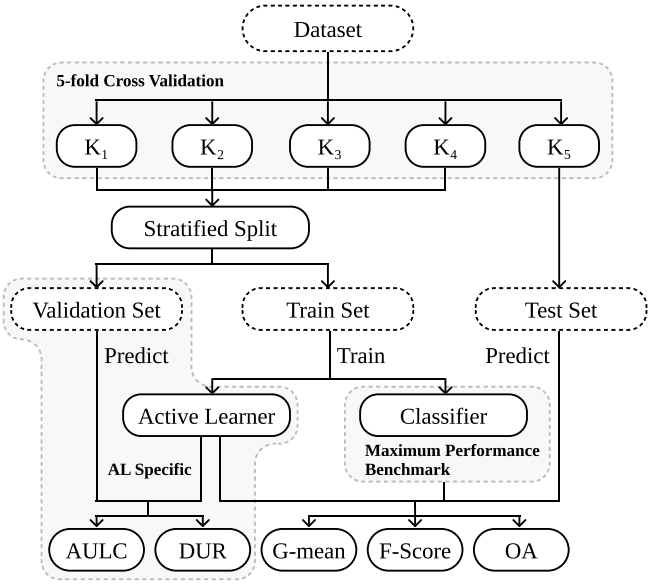

Figure 4: Experimental procedure flowchart. The preprocessed datasets are split into five folds. One of the folds is used to test the best-found classifiers using AL and the classifiers trained using the entire training dataset (containing the remaining folds). The training set is used to run both the 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.

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.

5.1 Results

The average rankings of the AL methods' AULC estimations are shown in Table 4. The proposed method almost always improves AL performance and ensures higher data selection efficiency.

Table 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.

Table 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	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: 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	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 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 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 6: AL algorithms' mean data utilization as a percentage of the training set.

G-mean	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 DUR scores relative to the Standard AL method are shown in Figure 5. 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).

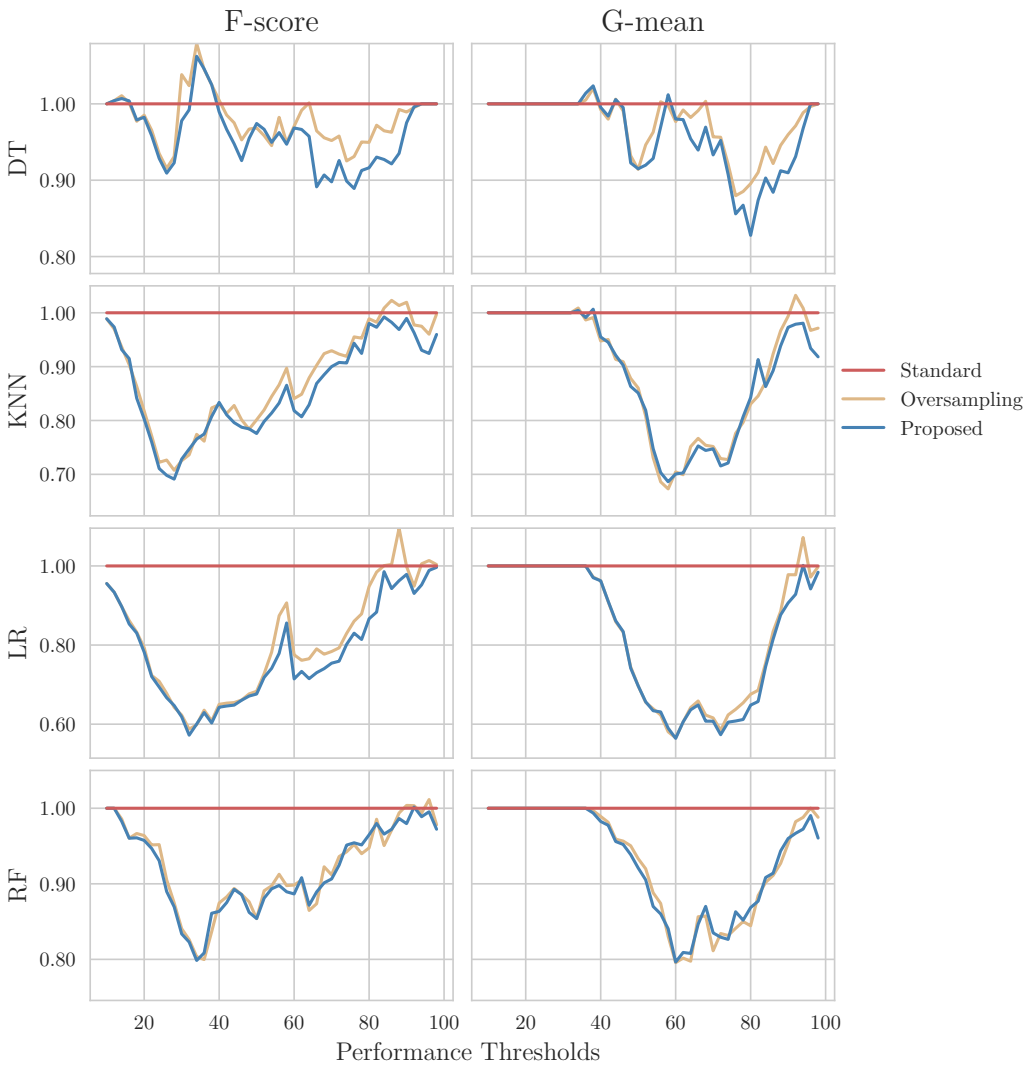

Figure 5: Mean data utilization rates. The y-axis shows the percentage of data (relative to the baseline AL framework) required to reach the different performance thresholds.

The comparison of mean optimal classification scores of AL methods with Classifiers (using the entire training set, without AL) is shown in Table 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).

Table 7: Optimal classification scores. The Maximum Performance (MP) classification scores are calculated using classifiers trained using the entire training set.

Classifier	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**

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 [58]. Overall, we perform three statistical tests. The Friedman test [59] 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 [60] 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 [61] 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 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 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 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

Table 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**

The *p-values* shown in Table 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.

5.3 Discussion

In this paper, we study the application of data augmentation methods through the modification of the

Table 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**

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 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 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, while showing performance improvements between the proposed method and both the standard and oversampling methods. The Friedman test in Table 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 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 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. 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 9 and 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 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 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 5.1), consistent with the findings in [20, 55]. 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.

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.

Data and Software Availability

The experiment was implemented using the Python programming language, along with the Python libraries Scikit-Learn [62], Imbalanced-Learn [63], Geometric-SMOTE [17], Research-Learn and ML-Research libraries. All functions, algorithms, experiments, and results are provided in the project's GitHub 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.

Acknowledgments

This research was supported by three research grants of the Portuguese Foundation for Science and Technology ("Fundação para a Ciência e a Tecnologia"), references SFRH/BD/151473/2021, DSAIPA/DS/0116/2019 and PCIF/SSI/0102/2017.

References

[1] Vishwesh Nath, Dong Yang, Bennett A. Landman, Daguang Xu, and Holger R. Roth. "Diminishing Uncertainty Within the Training Pool: Active Learning for Medical Image Segmentation". In: *IEEE Transactions on Medical Imaging* 40.10 (2021), pp. 2534–2547.

[2] Yuriy Sverchkov and Mark Craven. "A review of active learning approaches to experimental design for uncovering biological networks". In: *PLoS Computational Biology* 13 (6 June 2017). Ed. by Haiyan Huang, e1005466. ISSN: 15537358.

[3] Xiao Li, Da Kuang, and Charles X. Ling. "Active Learning for Hierarchical Text Classification". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7301 LNAI (PART 1 2012), pp. 14–25. ISSN: 03029743.

[4] Yayong Li, Jie Yin, and Ling Chen. "SEAL: Semisupervised Adversarial Active Learning on Attributed Graphs". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.7 (2021), pp. 3136–3147.

[5] Oriane Siméoni, Mateusz Budnik, Yannis Avrithis, and Guillaume Gravier. "Rethinking deep active learning: Using unlabeled data at model training". In: *Proceedings - International Conference on Pattern Recognition* (2020), pp. 1220–1227. ISSN: 10514651.

[6] Jesper E Van Engelen and Holger H Hoos. "A survey on semi-supervised learning". In: *Machine Learning* 109.2 (2020), pp. 373–440.

[7] Ozan Sener and Silvio Savarese. "Active Learning for Convolutional Neural Networks: A Core-Set Approach". In: *International Conference on Learning Representations*. 2018.

[8] Yan Leng, Xinyan Xu, and Guanghui Qi. "Combining active learning and semi-supervised learning to construct SVM classifier". In: *Knowledge-Based Systems* 44 (2013), pp. 121–131.

[9] Hualong Yu, Xibei Yang, Shang Zheng, and Changyin Sun. "Active Learning from Imbalanced Data: A Solution of Online Weighted Extreme Learning Machine". In: *IEEE Transactions on Neural Networks and Learning Systems* 30 (4 Apr. 2019), pp. 1088–1103. ISSN: 21622388.

[10] Hang Zhang, Weike Liu, and Qingbao Liu. "Reinforcement online active learning ensemble for drifting imbalanced data streams". In: *IEEE Transactions on Knowledge and Data Engineering* (2020).

[11] Weiwei Zong, Guang-Bin Huang, and Yiqiang Chen. "Weighted extreme learning machine for imbalance learning". In: *Neurocomputing* 101 (2013), pp. 229–242.

[12] Jiongming Qin, Cong Wang, Qinhong Zou, Yubin Sun, and Bin Chen. "Active learning with extreme learning machine for online imbalanced multiclass classification". In: *Knowledge-Based Systems* 231 (2021), p. 107385.

[13] Weike Liu, Hang Zhang, Zhaoyun Ding, Qingbao Liu, and Cheng Zhu. "A comprehensive active learning method for multiclass imbalanced data streams with concept drift". In: *Knowledge-Based Systems* 215 (2021), p. 106778.

[14] Alaa Tharwat and Wolfram Schenck. "Balancing Exploration and Exploitation: A novel active learner for imbalanced data". In: *Knowledge-Based Systems* 210 (2020), p. 106500.

[15] Haibo He and Edwardo A Garcia. "Learning from imbalanced data". In: *IEEE Transactions on knowledge and data engineering* 21.9 (2009), pp. 1263–1284.

[16] Yoon-Yeong Kim, Kyungwoo Song, JoonHo Jang, and Il-chul Moon. "LADA: Look-Ahead Data Acquisition via Augmentation for Deep Active Learning". In: *Advances in Neural Information Processing Systems* 34 (2021).

[17] Georgios Douzas and Fernando Bacao. "Geometric SMOTE a Geometrically Enhanced Drop-In Replacement for SMOTE". In: *Information Sciences* 501 (Oct. 2019), pp. 118–135.

[18] Julian Katz-Samuels, Jifan Zhang, Lalit Jain, and Kevin Jamieson. "Improved algorithms for agnostic pool-based active classification". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5334–5344.

[19] Tengfei Su, Shengwei Zhang, and Tingxi Liu. "Multi-spectral image classification based on an object-based active learning approach". In: *Remote Sensing* 12 (3 Feb. 2020), p. 504. ISSN: 20724292.

[20] Joao Fonseca, Georgios Douzas, and Fernando Bacao. "Increasing the Effectiveness of Active Learning: Introducing Artificial Data Generation in Active Learning for Land Use/Land Cover Classification". In: *Remote Sensing 2021, Vol. 13, Page 2619* 13.13 (July 2021), p. 2619.

[21] Donggeun Yoo and In So Kweon. "Learning loss for active learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 93–102.

[22] Hamed H Aghdam, Abel Gonzalez-Garcia, Antonio Lopez, and Joost Weijer. "Active learning for deep detection neural networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. Vol. 2019-Octob. 2019, pp. 3671–3679. ISBN: 9781728148038.

[23] Timothy M Hospedales, Shaogang Gong, and Tao Xiang. "Finding rare classes: Active learning with generative and discriminative models". In: *IEEE transactions on knowledge and data engineering* 25.2 (2011), pp. 374–386.

[24] Yuanzhang Li, Xinxin Wang, Zhiwei Shi, Ruyun Zhang, Jingfeng Xue, and Zhi Wang. "Boosting training for PDF malware classifier via active learning". In: *International Journal of Intelligent Systems* 37.4 (2022), pp. 2803–2821.

[25] Jiayi Li, Xin Huang, and Xiaoyu Chang. "A label-noise robust active learning sample collection method for multi-temporal urban land-cover classification and change analysis". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 163 (2020), pp. 1–17.

[26] Cong Su, Zhongmin Yan, and Guoxian Yu. "Cost-effective multi-instance multilabel active learning". In: *International Journal of Intelligent Systems* 36.12 (2021), pp. 7177–7203.

[27] Punit Kumar and Atul Gupta. "Active Learning Query Strategies for Classification, Regression, and Clustering: A Survey". In: *Journal of Computer Science and Technology 2020 35:4* 35 (4 July 2020), pp. 913–945. ISSN: 1860-4749.

[28] Yifan Fu, Xingquan Zhu, and Bin Li. "A survey on instance selection for active learning". In: *Knowledge and information systems* 35.2 (2013), pp. 249–283.

[29] Xiaoxue Wu, Wei Zheng, Xin Xia, and David Lo. "Data quality matters: A case study on data label correctness for security bug report prediction". In: *IEEE Transactions on Software Engineering* (2021).

[30] Wei Zhang, Ziwei Wang, and Xiang Li. "Blockchain-based decentralized federated transfer learning methodology for collaborative machinery fault diagnosis". In: *Reliability Engineering & System Safety* 229 (2023), p. 108885.

[31] Wei Zhang, Xiang Li, Hui Ma, Zhong Luo, and Xu Li. "Open-set domain adaptation in machinery fault diagnostics using instance-level weighted adversarial learning". In: *IEEE Transactions on Industrial Informatics* 17.11 (2021), pp. 7445–7455.

[32] Miao He and David He. "Deep learning based approach for bearing fault diagnosis". In: *IEEE Transactions on Industry Applications* 53.3 (2017), pp. 3057–3065.

[33] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. "Grass: Generative recursive autoencoders for shape structures". In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), pp. 1–14.

[34] Wenfeng Zheng, Xiangjun Liu, and Lirong Yin. "Sentence representation method based on multi-layer semantic network". In: *Applied Sciences* 11.3 (2021), p. 1316.

[35] Kai Zhang, Zhongzheng Wang, Guodong Chen, Liming Zhang, Yongfei Yang, Chuanjin Yao, Jian Wang, and Jun Yao. "Training effective deep reinforcement learning agents for real-time life-cycle production optimization". In: *Journal of Petroleum Science and Engineering* 208 (2022), p. 109766.

[36] Sima Behpour, Kris M. Kitani, and Brian D. Ziebart. "ADA: Adversarial data augmentation for object detection". In: *Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision, WACV 2019* (Mar. 2019), pp. 1243–1252.

[37] Joao Fonseca, Georgios Douzas, and Fernando Bacao. "Improving imbalanced land cover classification with K-Means SMOTE: Detecting and oversampling distinctive minority spectral signatures". In: *Information* 12.7 (2021), p. 266.

[38] Terrance DeVries and Graham 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, Feb. 2017.

[39] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of Big Data* 6.1 (2019), pp. 1–48.

[40] Omid Kashefi and Rebecca 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: Association for Computational Linguistics, Nov. 2020, pp. 200–208.

[41] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. "Random erasing data augmentation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 13001–13008.

[42] Toan Tran, Thanh-Toan Do, Ian Reid, and Gustavo Carneiro. "Bayesian generative active deep learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6295–6304.

[43] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. "Variational adversarial active learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5972–5981.

[44] Kwanyoung Kim, Dongwon Park, Kwang In Kim, and Se Young Chun. "Task-aware variational adversarial active learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8166–8175.

[45] Yu Ma, Shaoxing Lu, Erya Xu, Tian Yu, and Lijian Zhou. "Combining Active Learning and Data Augmentation for Image Classification". In: *Proceedings of the 2020 3rd International Conference on Big Data Technologies*. 2020, pp. 58–62.

[46] Husam Quteineh, Spyridon Samothrakis, and Richard 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)*. 2020, pp. 7400–7410.

[47] Qingqing Li, Zhen Huang, Yong Dou, and Ziwen Zhang. "A Framework of Data Augmentation While Active Learning for Chinese Named Entity Recognition". In: *International Conference on Knowledge Science, Engineering and Management*. Springer. 2021, pp. 88–100.

[48] Chialin Wu. *The decision tree approach to classification*. Purdue University, 1975.

[49] T Cover and P Hart. "Nearest neighbor pattern classification". In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27. ISSN: 0018-9448.

[50] Tin Kam Ho. "Random Decision Forests". In: *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*. ICDAR '95. USA: IEEE Computer Society, 1995, p. 278. ISBN: 0818671289.

[51] John Ashworth Nelder and Robert WM Wedderburn. "Generalized linear models". In: *Journal of the Royal Statistical Society: Series A (General)* 135.3 (1972), pp. 370–384.

[52] László A. Jeni, Jeffrey F. Cohn, and Fernando 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*. 2013, pp. 245–251. ISBN: 9780769550480.

[53] Mehrdad Fatourechi, Rabab K Ward, Steven G Mason, Jane Huggins, Alois Schloegl, and Gary E Birch. "Comparison of evaluation metrics in classification applications with imbalanced datasets". In: *2008 seventh international conference on machine learning and applications*. IEEE. 2008, pp. 777–782.

[54] Miroslav Kubat, Stan Matwin, et al. "Addressing the curse of imbalanced training sets: one-sided selection". In: *Icml*. Vol. 97. Citeseer. 1997, pp. 179–186.

[55] Daniel Kottke, Adrian Calma, Denis Huseljic, Georg Krempl, and Bernhard Sick. "Challenges of reliable, realistic and comparable active learning evaluation". In: *CEUR Workshop Proceedings*. Vol. 1924. Sept. 2017, pp. 2–14.

[56] Tobias Reitmaier and Bernhard Sick. "Let us know your decision: Pool-based active training of a generative classifier with the selection strategy 4DS". In: *Information Sciences* 230 (May 2013), pp. 106–131. ISSN: 0020-0255.

[57] Jean-François Kagy, Tolga Kayadelen, Ji Ma, Afshin Rostamizadeh, and Jana Strnadova. "The Practical Challenges of Active Learning: Lessons Learned from Live Experimentation". In: *arXiv preprint arXiv:1907.00038* (June 2019).

[58] Janez Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *Journal of Machine Learning Research* 7 (2006), pp. 1–30.

[59] Milton Friedman. "The use of ranks to avoid the assumption of normality implicit in the analysis of variance". In: *Journal of the american statistical association* 32.200 (1937), pp. 675–701.

[60] Frank Wilcoxon. "Individual Comparisons by Ranking Methods". In: *Biometrics Bulletin* 1.6 (Dec. 1945), p. 80. ISSN: 00994987.

[61] Sture Holm. "A simple sequentially rejective multiple test procedure". In: *Scandinavian journal of statistics* (1979), pp. 65–70.

[62] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.Oct (2011), pp. 2825–2830. ISSN: ISSN 1533-7928.

[63] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning". In: *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5.