Data Augmentation Improves the Performance of Active Learning Methods

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

Work in progress (Abstract). Highlights below.

- Proposed method outperforms the remaining frameworks
- Proposed method is more robust to the choice of domain, classifiers and performance metrics being chosen
- Proposed method outperforms classifiers trained using the full training dataset. To the best of our knowledge, this is the first AL method able to do so reliably.

1 Introduction

The importance of leveraging unlabeled data along with labeled data to improve machine learning tasks is substantially growing. The increasing amount of valuable data sources and formats being developed and explored is affecting various domains [1]. On the one hand, because this data is often originally unlabeled, only a small amount of the data being produced and stored can be useful for Machine Learning (ML) tasks. On the other hand, labeling data for a specific ML project is often difficult, especially when data-intensive ML techniques are involved (*e.g.,* Deep Learning classifiers) [2]. In this scenario, labeling the full dataset becomes impractical, time-consuming and expensive. Two different approaches of ML attempt to address this problem: Semi-Supervised Learning (SSL) and Active Learning (AL) [3]. Even though they address the same problem, the two follow opposite approaches. SSL focuses on observations with the most certain predictions, whereas AL focuses on observations with the least certain predictions.

Semi-supervised learning

Active learning

Imbalanced learning within AL [4]

Work in Progress (Introduction).

In this study, we introduce two modifications of the AL framework: (1) The exploration of a different data generation strategy (*i.e.,* data augmentation using G-SMOTE beyond the typical oversampling strategy) and (2) Add parameter optimization within the iterative process of an AL procedure.

These modifications were developed with the goal of reducing the amount of iterations required to produce a classifier with a performance as good as classifiers trained with the entire training dataset (*i.e.,* without the need of AL). This is done via the collection of a reduced data subset containing the most informative observations for the training phase of the classifier. With our contribution, data selection in AL iterations target observations that optimize the quality of the artificial data produced. The substitution of labeled data with artificial data is especially useful in this context, since it allows the reduction of user interaction necessary to reach a sufficiently informative dataset.

The AL framework discussed in this study was recently proposed within the Remote Sensing domain for Land Use/Land Cover classification [5]. We further extend this study by applying the AL framework discussed in a context agnostic environment, considering 10 different multiclass datasets from different domains.

NOTE: Add figure similar to one found in (but in order to explain the advantages of data augmentation): Kumar, P., Gupta, A. (2020). Active Learning Query Strategies for Classification, Regression, and Clustering: A Survey. Journal of Computer Science and Technology

NOTE: A nice, but short introduction: Bshouty, N. H., Gentile, C. (Eds.). (2007). Learning Theory. Lecture Notes in Computer Science.

The scope of this paper is to propose a generalization and improvement of the AL framework proposed in the previous paper. In addition, we replace the oversampling strategy used in the previous paper and replace it with a varying levels of data augmentation, producing larger amounts of data than the method used in the original paper.

2 Active Learning Methods

Supervised Machine Learning (ML) algorithms typically perform well in contexts where labeled data is abundant and easily accessible **[CITATION]**. However, in a practical setting, finding this data is frequently a challenging task. Specifically, 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 [6].

Active Learning (AL) is a method that addresses this problem via the labeling of the most informative 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 [7]. In a scenario with limited budget, time or availability of labeled data, AL allows the implementation of near-optimal classifiers with minimal effort **[CITATION]**.

AL methods may be divided into 2 different stages, initialization and iteration. Figure 1 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 the following steps [5]:

1. Collection of an unlabeled dataset. The procedure to carry out this step depends on the context.

2. Selection of an initial data subset. Typically, when there is no a priori labeled dataset (**i.e.,** cold start problem), the initial data subset is randomly picked from the unlabeled dataset.

3. Data labeling. The supervisor is presented with the data subset, where the supervisor's goal is to label each observation. Some of the research refers to the supervisor as the oracle.

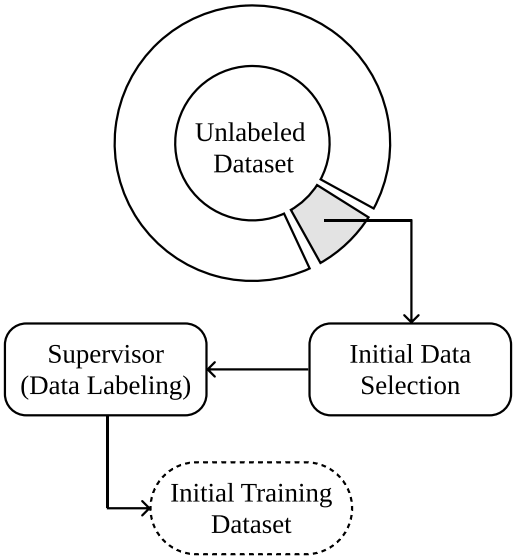

Figure 1: Active Learning initialization.

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 2 and is composed of the following steps [8, 9]:

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 [10] **[CITATION]**.

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. The selected observations from the unlabeled dataset are presented to the supervisor, which is responsible for manually labeling the observations.

4. Update the current training dataset with the new training observations. The new (labeled) training observations are added to the training dataset and the iteration is completed.

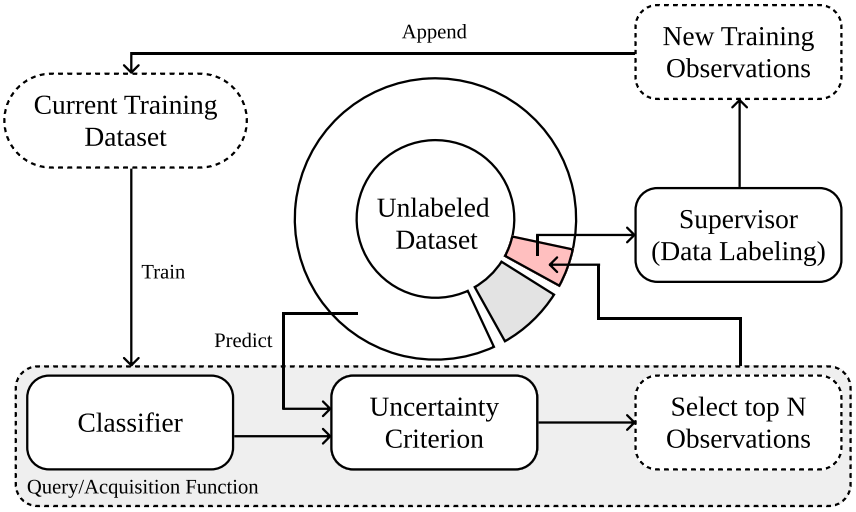

Figure 2: Active Learning iteration. In the first iteration, the training set collected during the initialization process becomes the "Current Training Dataset".

Two common challenges found in AL implementations is the consistency and efficiency of AL in practical scenarios [11]. 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 [**CITATION**]: (1) Human error during data labeling, (2) Non-informative initial training dataset and (3) Lack of an appropriate uncertainty criterion. However, AL research typically focuses on either domain-specific applications of AL, or in the improvement of the AL iterative process [**CITATION**]. Specifically, most of the methodological research found in AL focuses in the improvement of the Query function [**CITATION**]. Query functions can be divided into two different categories [12, 13]:

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 [14].

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. As pointed out in [13], 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.

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. However, a few studies attempted to do so. Specifically, in [15] 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 [3] 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. In [5], the authors introduce the generator element in the AL framework (discussed in Section 4) using an oversampling method, showing that this method effectively addresses the limitations of imbalanced learning. However, this method was implemented specifically in the Remote Sensing domain and used an oversampling strategy without consideration for the actual amount of artificial data generated, which may limit its performance.

2.1 Query Strategies

Representative query strategies are less efficient in data selection than Informative query strategies [13]. However, many studies attempt to improve its efficiency using 3 main approaches. For example, [16, 17, 18] used a density-based approach using clustering algorithms to select the observations closest to the centroid of each cluster. The diversity-based approach was developed to avoid the selection of redundant observations in batch-mode learning [19]. Other representative strategies were also developed for graph-based structures [20] which attempt to find the most representative nodes and edges of a graph network. However, recent research often use representative approaches alongside informative approaches [12, 21].

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) [13]. 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 [10]. Specifically, Settles [22] observed that in some datasets informative query strategies fail to outperform the random selection of observations. Generally, the Random Selection query method method is used as a baseline method. 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 [23]. 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 [24]. 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 [25, 26].

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 [27]. 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 [28].

The Query by Committee (QBC) strategy was developed to address ensemble classifiers. It is a dis-

agreement 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 (described previously) 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 [29], while the performance between the two strategies appears to differ significantly across various scenarios [30].

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 [13]. Although, since this method is known to lead to the excessive selection of observations in dense regions [31], it was improved in various ways. Specifically, in [31] the authors extend this strategy by applying the manifold-preserving graph reduction algorithm beyond the normal Margin Sampling method.

Reminders:

- Discuss maximum performance thresholds.
- There is no uncertainty method that accounts for within-iteration data augmentation.

3 Data Augmentation Methods

Data Augmentation methods expand the training dataset by introducing new and informative observations [32]. The production of artificial data may be done via the introduction of perturbations on the input [33], feature [34] or output space [32]. Data Augmentation methods may be divided into Heuristic and Neural Network-based approaches [35]. In addition, they may also be distinguished based on its generation strategy, whether local (considers a local/specific subset of the dataset) or global (considers the overall distribution of the training dataset) **[CITATION]**. Figure 3 shows the general taxonomy of Data Augmentation methods. Finding the appropriate Data Augmentation method generally depends on the domain [34], although some studies discuss which methods are more appropriate according to the domain [35, 36, 37].

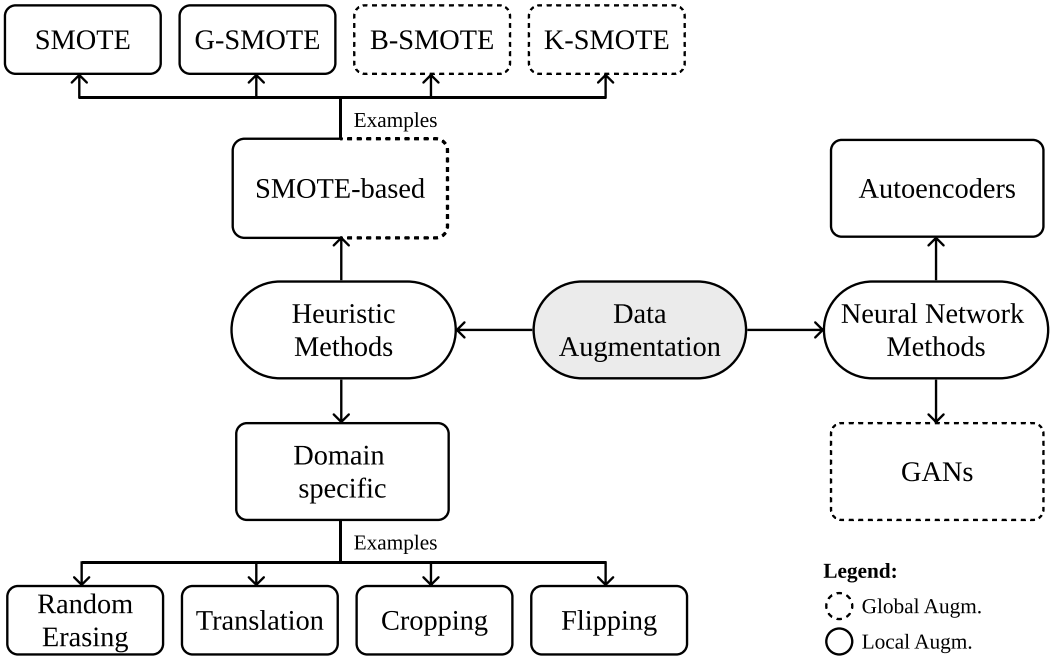

Figure 3: Data Augmentation Taxonomy.

Heuristic approaches attempt to generate new and relevant observations through the application a predefined procedure, usually incorporating some degree of randomness [38]. 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 [34]. 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 paper (the paper's contributions are described in Sections 1 and 4), the computational power available for this experiment and the breadth of datasets used in our experimental procedure, we will focus on domain-agnostic heuristic data augmentation methods.

While some techniques may depend on the domain, others may be considered domain-agnostic. For example, Random Erasing [33], 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) [39], 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 [40]. 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 4.

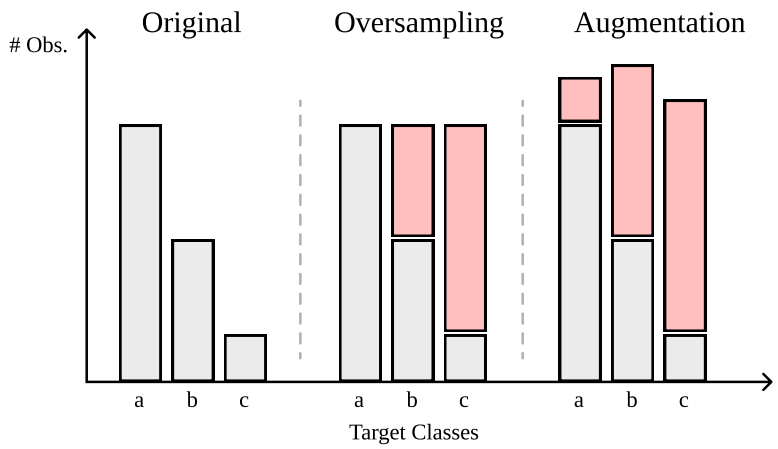

Figure 4: Examples of data augmentation Strategies. The salmon-colored bars represent artificial data using the normal oversampling (center group) and an example of augmentation (right group) strategies.

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 [41].

The SMOTE method generates artificial data via the linear interpolation between a random observation and one of its k-nearest neighbors (also randomly selected) [39]. 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 introduces little variability into the training dataset [42]. Borderline-SMOTE (B-SMOTE) [43] 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) [44]. 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.

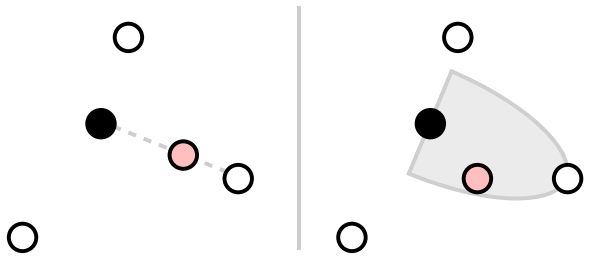

Figure 5: Examples of data generation using SMOTE and G-SMOTE. In this example, both G-SMOTE's deformation and truncation parameters assume values around 0.5.

Geometric-SMOTE (G-SMOTE) [42] modifies SMOTE's generation mechanism. Instead of generating an observation as 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 5. Reference [41] found that G-SMOTE outperforms various state-of-the-art oversamplers.

4 Proposed Method

Based on the literature found on AL, most of the contributions and novel implementations of AL algorithms 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 full training set (when that happens). Finally, in [5] the authors also found a significant variability of the data selection efficiency during different runs of the AL iterative process. In this study the authors proposed a new element within the AL framework, the generator, which was able to marginally reduce the variability previously identified. However, this modification was applied in a Land Use/Land Cover context which contains specific characteristics that are not necessarily found in other supervised learning problems. Specifically, high dimensional datasets containing target classes with little data variability (*i.e.,* cohesive spectral signatures within classes) due to their geographical proximity. Furthermore, the implementation of the generator was done using a simple oversampling strategy, which limits the possibility of employing other techniques with an undefined target amount of data generated at each iteration.

Our work builds upon the work developed in [5], with the following contributions:

1. We improve the AL framework by introducing a parameter tuning stage using only the labeled dataset available at the current iteration (*i.e.,* no hold-out set needed).

2. We generalize the usage of the generator from just using oversampling techniques to any other data augmentation mechanism and/or strategy.

3. We analyze the performance of the proposed and oversampling frameworks over 10 different datasets of different domains, while comparing them with the standard AL framework.

The proposed iterative process of the AL framework is depicted in Figure 6. The generator element becomes an additional source of data and is expected to introduce additional data variability into the training dataset. This 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 strategy at each iteration. In our implementation, this step uses the current training dataset to perform an exhaustive search over specified parameters of the generator, tested over a 5-fold cross validation method. The best augmentation strategy found is used to train the iteration's classifier in the following step.

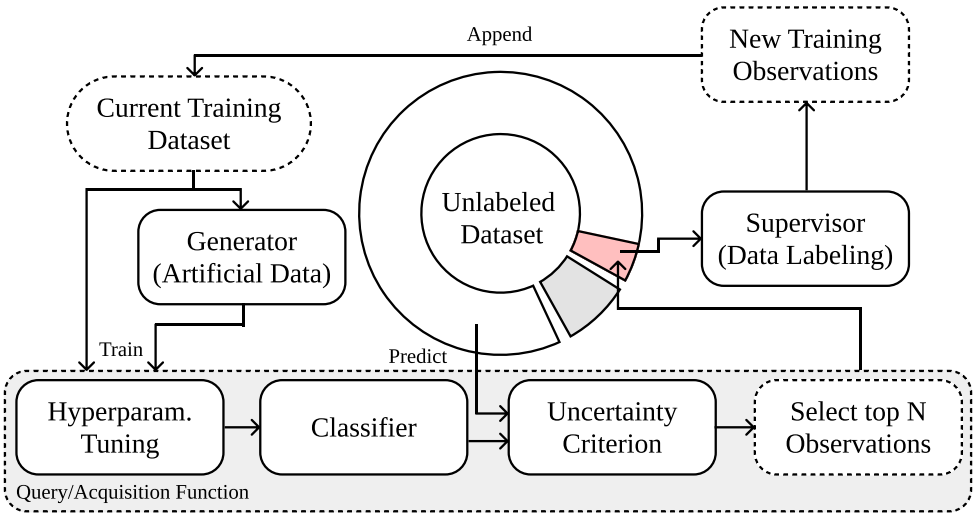

Figure 6: Active Learning — proposed iteration.

To show the effectiveness of data augmentation in an AL implementation, we implemented a simple modification of the G-SMOTE algorithm. This modification facilitates the usage of G-SMOTE beyond its original oversampling purposes. In this paper, the data augmentation strategies used ensure the all the class frequencies are balanced. Furthermore, the amount of artificial data produced for each class is defined by the *augmentation factor*, 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 [5]. This allows the direct comparison of the two frameworks and establish a causality of the performance variations to the data generation mechanism (*i.e.,* augmentation vs normal oversampling) and hyperparameter tuning steps.

This framework was designed to be task-agnostic. Specifically, any data augmentation method (domain specific or not) may be used, as well as any other parameter search method. It is also expected to be compatible with other AL modifications, including the ones that do not affect solely the classifier or uncertainty criterion, such as the one proposed in [15].

5 Methodology

This section describes the different elements included in the experimental procedure. The datasets used were acquired in open data repositories and its sources and preprocessing steps are defined in Subsection 5.1. The choice of classifiers used in the experiment are defined in Subsection 5.2. The metrics chosen to measure AL performance and overall classification performance are defined in Subsection 5.3. The experimental procedure is described in Subsection 5.4. The implementation of the experiment and resources used to do so are described in Subsection 5.5.

The methodology developed serves 2 purposes: (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.,* number of

AL iterations required to reach similar classification performances).

5.1 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. The datasets were chosen considering different domains of application, 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 1 describes the key properties of the 10 preprocessed datasets where the experimental procedure was applied.

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
BASEBALL	15	1320	57	1196	20.98	3

Table 1: Description of the datasets collected after data preprocessing. The sampling strategy is similar across datasets. Legend: (IR) Imbalance Ratio

The data preprocessing pipeline is depicted as a flowchart in Figure 7. The missing values are removed from each dataset by removing the corresponding observations. This 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.

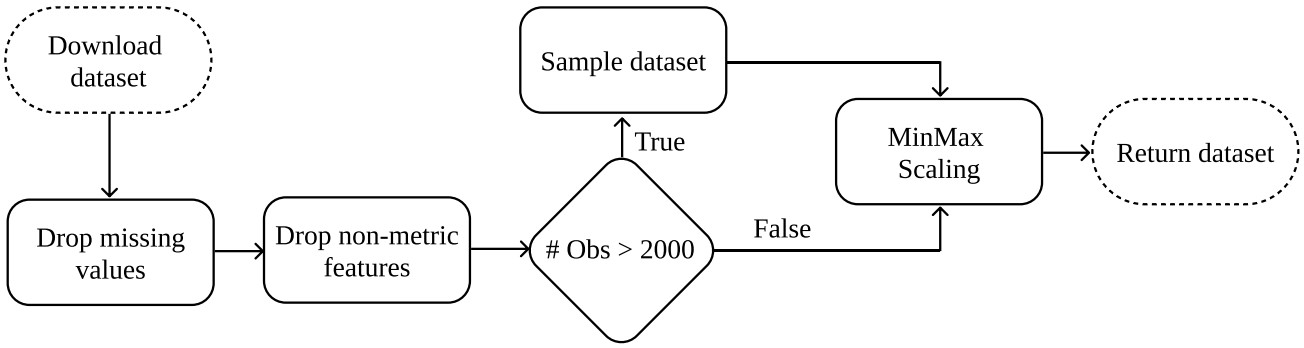

Figure 7: Data preprocessing pipeline.

The preprocessed datasets were stored into a SQLite database file and is available along with the experiment's source code in the GitHub repository of the project (see Subsection 5.5).

5.2 Machine Learning Algorithms

We used a total of 4 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) [45], a K-nearest neighbors classifier (KNN) [46], a Random Forest Classifier (RF) [47] and a Logistic Regression (LR) [48]. 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 sampling strategy 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 sampling strategy 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 4.

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 normal oversampling method, as proposed in [5]. 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 4.

5.3 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 quantify a

model's classification performance [49]. 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 [50]. However, these metrics remain popular choices for the evaluation of classification performance. Other performance metrics like $Precision = \frac{TP}{TP+TN}$, $Recall = \frac{TP}{TP+FN}$ 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 at a per-class basis instead. In a multiple dataset with varying amount of target classes and meanings, comparing the performance of different models using these metrics becomes impractical.

Based on the recommendations found in [49, 51], we used 2 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 [51]. 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:

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

- 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 [49]:

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

- 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 strategy allows the production of classifiers with high performance on unseen data while using as least non-artificial training data as possible. To measure the performance of the different AL setups, we follow the recommendations found in [11]. 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. To facilitate the interpretability of this metric, 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).

- Data Utilization Rate (DUR) [52]. 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 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 [53]. In this scenario, since the dataset is already labeled, the annotation process is done at zero cost. Figure 8 depicts the experiment designed for one dataset over a single run.

A single run starts with the splitting of a preprocessed dataset in 5 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 5 times using a different partition as the Test set each time. For each training process, a Validation set is created and is used to measure the data selection efficiency (*i.e.,* AULC and DUR using the classification performance metrics, specific to AL). The AL simulations and the classifiers' training occur within the Train 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 8 is repeated over 3 runs using different random seeds over the 10 different datasets collected. The final scores of each AL configuration and classifier correspond to the average of the 3 runs and 5-fold Cross Validation estimations (*i.e.,* the mean score of 15 fits, across 10 datasets).

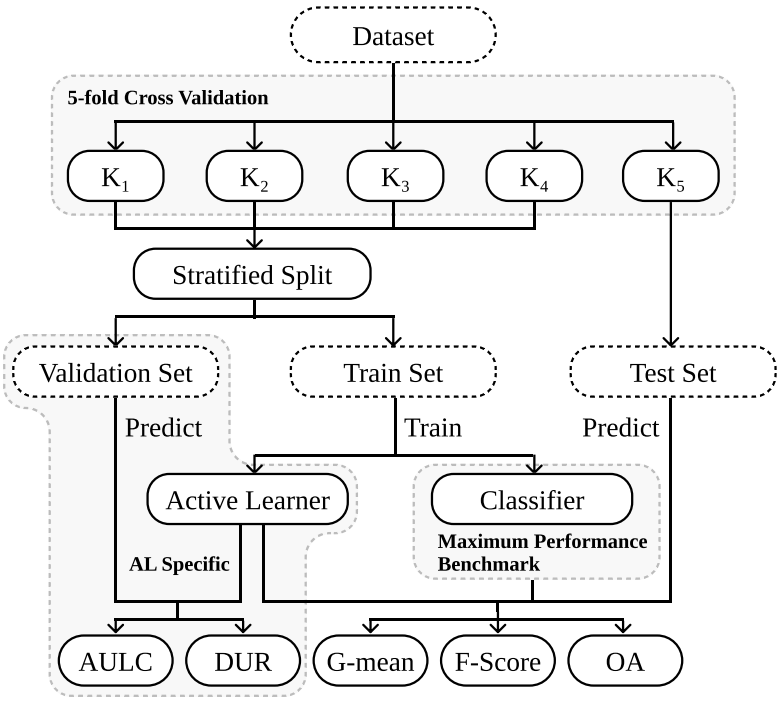

Figure 8: Experimental procedure flowchart.

The hyperparameters defined for the AL frameworks, Classifiers and Generators are shown in Table 2. In the Generators table, we distinguish the G-SMOTE algorithm working as a normal oversampling method from G-SMOTE-AUGM, which performs 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.

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

Table 2: Hyper-parameter definition for the active learners, classifiers and generators used in the experiment.

5.5 Software Implementation

The experiment was implemented using the Python programming language, along with the Python libraries Scikit-Learn [54], Imbalanced-Learn [55], Geometric-SMOTE [42], Research-Learn and ML-Research libraries. All functions, algorithms, experiments and results are provided in the GitHub repository of the project.

6 Results & Discussion

In a multiple dataset experiment, the analysis of results should not rely uniquely on the average performance scores across datasets. 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 the use of the mean ranking scores to extend the analysis [56]. 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 3 different runs of 5-fold Cross Validation (15 performance estimations per dataset) for each combination of dataset, AL configuration, classifier and performance metric.

6.1 Results

The average ranking of the AULC estimations of AL methods are shown in Table 3. The proposed method almost always improves AL performance and ensures higher data selection efficiency.

Classifier	Evaluation Metric	Standard	Oversampling	Proposed
DT	Accuracy	2.50 ± 0.81	2.20 ± 0.4	**1.30 ± 0.64**
DT	F-score	2.50 ± 0.81	2.10 ± 0.3	**1.40 ± 0.8**
DT	G-mean	2.70 ± 0.64	2.00 ± 0.45	**1.30 ± 0.64**
KNN	Accuracy	2.40 ± 0.8	1.90 ± 0.54	**1.70 ± 0.9**
KNN	F-score	2.60 ± 0.66	1.80 ± 0.4	**1.60 ± 0.92**
KNN	G-mean	2.80 ± 0.4	1.70 ± 0.46	**1.50 ± 0.81**
LR	Accuracy	2.60 ± 0.66	2.10 ± 0.54	**1.30 ± 0.64**
LR	F-score	2.80 ± 0.4	2.00 ± 0.45	**1.20 ± 0.6**
LR	G-mean	2.80 ± 0.4	2.00 ± 0.45	**1.20 ± 0.6**
RF	Accuracy	2.60 ± 0.66	1.90 ± 0.54	**1.50 ± 0.81**
RF	F-score	2.60 ± 0.66	2.00 ± 0.45	**1.40 ± 0.8**
RF	G-mean	2.80 ± 0.4	**1.60 ± 0.49**	**1.60 ± 0.8**

Table 3: Mean rankings of the AULC metric over the different datasets (10), folds (5) and runs (3) used in the experiment. The proposed method always improves the results of the original framework and on average almost always improves the results of the oversampling framework.

Table 4 shows the average AULC scores, grouped by classifier, Evaluation Metric and AL framework. The variation in performance across active learners is consistent with the mean rankings found in Table 3,

while showing significant AULC score differences between the proposed AL method and the oversampling AL method.

Classifier	Evaluation Metric	Standard	Oversampling	Proposed
DT	Accuracy	0.733 ± 0.092	0.732 ± 0.087	**0.740 ± 0.087**
DT	F-score	0.695 ± 0.088	0.698 ± 0.09	**0.705 ± 0.092**
DT	G-mean	0.804 ± 0.065	0.811 ± 0.06	**0.816 ± 0.062**
KNN	Accuracy	0.816 ± 0.091	0.818 ± 0.088	**0.822 ± 0.091**
KNN	F-score	0.775 ± 0.102	0.784 ± 0.108	**0.788 ± 0.111**
KNN	G-mean	0.852 ± 0.084	0.866 ± 0.072	**0.869 ± 0.074**
LR	Accuracy	0.802 ± 0.091	0.812 ± 0.088	**0.821 ± 0.086**
LR	F-score	0.749 ± 0.112	0.773 ± 0.116	**0.784 ± 0.115**
LR	G-mean	0.839 ± 0.093	0.870 ± 0.065	**0.875 ± 0.064**
RF	Accuracy	0.861 ± 0.076	0.861 ± 0.075	**0.862 ± 0.077**
RF	F-score	0.823 ± 0.105	0.827 ± 0.105	**0.829 ± 0.105**
RF	G-mean	0.886 ± 0.077	**0.895 ± 0.063**	**0.895 ± 0.065**

Table 4: 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 total of XXXX% instances of the XXXX% instances that compose the training sets.

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 shows the results obtained for these scores starting from a G-mean score of 0.6 and was filtered to show only the thresholds ending with 0 or 6. In most cases, the proposed method reduces the amount of data annotation required to reach each G-mean score threshold.

G-mean Score	Classifier	Standard	Oversampling	Proposed
0.60	DT	3.2%	**3.1%**	3.2%
0.60	KNN	3.6%	2.6%	**2.5%**
0.60	LR	3.9%	**2.2%**	**2.2%**
0.60	RF	2.4%	**2.1%**	**2.1%**
0.66	DT	4.6%	4.6%	**4.2%**
0.66	KNN	4.9%	3.7%	**3.5%**
0.66	LR	5.7%	3.2%	**3.1%**
0.66	RF	3.0%	2.8%	**2.7%**
0.70	DT	6.6%	6.1%	**5.8%**
0.70	KNN	8.5%	5.0%	**4.7%**
0.70	LR	9.5%	4.6%	**4.3%**
0.70	RF	4.5%	**3.2%**	3.3%
0.76	DT	16.5%	13.0%	**12.7%**
0.76	KNN	17.8%	9.7%	**9.0%**
0.76	LR	16.6%	10.0%	**7.8%**
0.76	RF	10.1%	**5.5%**	**5.5%**
0.80	DT	36.1%	30.4%	**27.1%**
0.80	KNN	22.7%	18.0%	**17.8%**
0.80	LR	25.2%	16.0%	**14.2%**
0.80	RF	15.5%	**9.0%**	9.5%
0.86	DT	60.5%	56.7%	**54.5%**
0.86	KNN	39.9%	**37.0%**	37.8%
0.86	LR	32.6%	27.5%	**27.0%**
0.86	RF	28.0%	**25.7%**	**25.7%**
0.90	DT	72.5%	70.7%	**67.8%**
0.90	KNN	49.9%	50.3%	**49.3%**
0.90	LR	52.5%	53.8%	**49.3%**
0.90	RF	44.6%	**42.6%**	43.5%
0.96	DT	100.0%	**99.5%**	100.0%
0.96	KNN	79.4%	75.6%	**71.6%**
0.96	LR	87.5%	83.1%	**79.8%**
0.96	RF	63.6%	64.2%	**63.1%**

Table 5: Mean data utilization of AL algorithms, as a percentage of the training set.

The DUR scores relative to the Standard AL method are shown in Figure 9. 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 strategy using the KNN classifier requires 69.6% 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 30.4% less data).

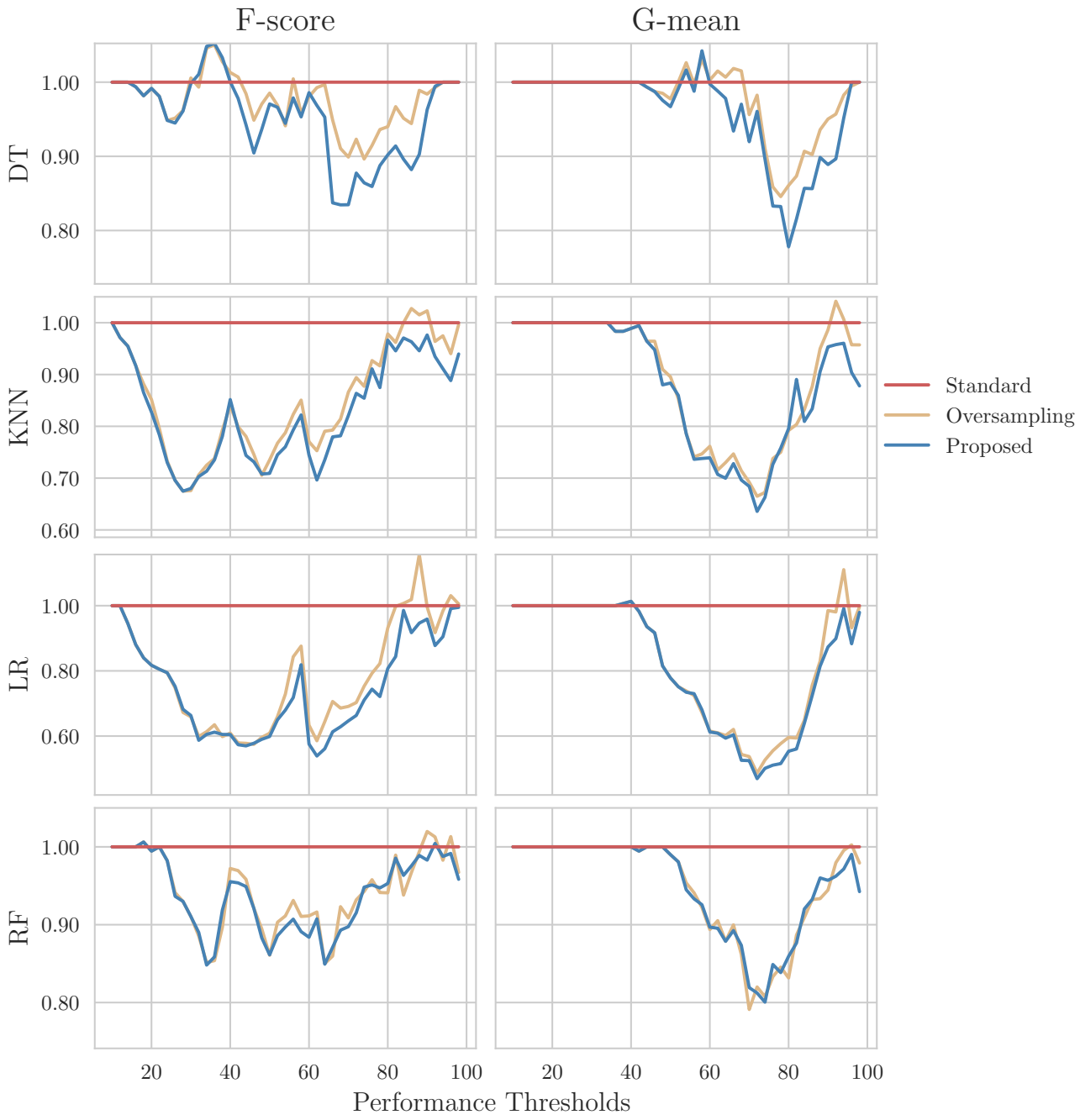

Figure 9: Mean data utilization rates. The y-axis shows the percentage of data (relative to the baseline AL framework) required to reach the different performance thresholds.

The mean optimal classification scores of AL methods and Classifiers (fully labeled training set, without AL) is shown in Table 6. The proposed AL method produces classifiers that are almost always able to outperform classifiers using the full training set (*i.e.,* the ones labeled as MP).

Classifier	Evaluation Metric	MP	Standard	Oversampling	Proposed
DT	Accuracy	0.809 ± 0.086	0.802 ± 0.089	0.806 ± 0.089	**0.812 ± 0.087**
DT	F-score	0.774 ± 0.107	0.772 ± 0.096	0.775 ± 0.101	**0.781 ± 0.103**
DT	G-mean	0.853 ± 0.081	0.854 ± 0.069	0.860 ± 0.067	**0.864 ± 0.068**
KNN	Accuracy	0.882 ± 0.085	**0.883 ± 0.087**	0.877 ± 0.087	0.881 ± 0.093
KNN	F-score	0.848 ± 0.116	0.849 ± 0.115	0.847 ± 0.118	**0.852 ± 0.121**
KNN	G-mean	0.896 ± 0.094	0.899 ± 0.09	0.904 ± 0.078	**0.907 ± 0.08**
LR	Accuracy	0.855 ± 0.074	**0.870 ± 0.073**	0.858 ± 0.077	**0.870 ± 0.076**
LR	F-score	0.812 ± 0.113	0.835 ± 0.105	0.825 ± 0.106	**0.838 ± 0.106**
LR	G-mean	0.875 ± 0.099	0.895 ± 0.075	0.899 ± 0.059	**0.907 ± 0.059**
RF	Accuracy	0.897 ± 0.08	0.905 ± 0.078	0.904 ± 0.078	**0.906 ± 0.077**
RF	F-score	0.867 ± 0.107	**0.877 ± 0.103**	0.875 ± 0.108	**0.877 ± 0.108**
RF	G-mean	0.911 ± 0.081	0.917 ± 0.078	0.923 ± 0.067	**0.925 ± 0.065**

Table 6: Optimal classification scores. The Maximum Performance (MP) classification scores are calculated using classifiers trained using the entire training set.

6.2 Statistical Analysis

When checking for statistical significance in a multiple dataset context it is important to account for the multiple comparison problem. Consequently, our statistical analysis focuses on the recommendations found in [56]. Overall, we perform 3 statistical tests. The Friedman test [57] is used to understand whether there is a statistically significant difference in performance between the 3 AL frameworks. As post hoc analysis, the Wilcoxon signed-rank test [58] was used 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 [59] method was used to check for statistical significance between the methods using data generators and the Standard AL framework across classifiers and evaluation metrics.

Table 7 contains 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.

Classifier	Evaluation Metric	p-value	Significance
DT	Accuracy	2.1e-17	True
DT	F-score	2.5e-24	True
DT	G-mean	2.8e-16	True
KNN	Accuracy	1.1e-46	True
KNN	F-score	1.8e-66	True
KNN	G-mean	6.4e-42	True
LR	Accuracy	9.9e-59	True
LR	F-score	2.0e-76	True
LR	G-mean	2.2e-59	True
RF	Accuracy	5.7e-42	True
RF	F-score	4.6e-55	True
RF	G-mean	1.3e-38	True

Table 7: 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.

Table 8 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 normal oversampling strategy proposed in [5] with statistical significance in 9 out of 10 datasets.

Dataset	Oversampling	Standard
Baseball	5.0e-01	3.4e-01
Gas Drift	**3.7e-26**	**4.6e-57**
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**
Texture	**1.5e-22**	**6.7e-57**
Vehicle	**7.4e-11**	**7.9e-13**
Waveform	**8.9e-08**	**2.6e-02**

Table 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 oversampling or standard framework.

The *p-values* shown in Table 9 refer to the results of the Holm-Bonferroni test. The proposed method's superior performance was statistically significant for any combination of classifier and evaluation metric. Simultaneously, the proposed method established statistical significance in the 3 scenarios where the oversampling AL method failed to do so.

Classifier	Evaluation Metric	Oversampling	Proposed
DT	Accuracy	**4.5e-05**	**1.6e-10**
DT	F-score	**1.9e-07**	**2.7e-10**
DT	G-mean	**2.5e-06**	**3.1e-09**
KNN	Accuracy	5.5e-02	**1.1e-05**
KNN	F-score	**6.7e-11**	**6.3e-14**
KNN	G-mean	**8.3e-06**	**1.3e-07**
LR	Accuracy	8.1e-02	**3.4e-06**
LR	F-score	**7.1e-06**	**2.0e-20**
LR	G-mean	**2.2e-07**	**1.1e-11**
RF	Accuracy	2.0e-01	**2.8e-02**
RF	F-score	**2.2e-05**	**8.1e-07**
RF	G-mean	**2.0e-04**	**2.0e-04**

Table 9: 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).

6.3 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.

In Table 3 we found that the proposed method was able to outperform the Standard AL framework in all scenarios. The mean rankings are consistent with the mean AULC scores found in Table 4, while showing significant performance differences between the proposed method and both the standard and oversampling methods. The Friedman test in Table 7 showed that the difference in the performance of these AL frameworks is statistically significant, regardless of the classifier or performance metric being used.

The proposed method showed more consistent data utilization requirements to most of the assessed G-mean score thresholds when compared to the remaining AL methods, as seen in Table 5. 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 9. In most cases, this method was able 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 8 and 9 showed that the proposed method's superiority was statistically significant in all datasets except one (Baseball) and established statistical significance when compared to the Standard AL method for all combinations of classifier and performance metric, including when the Oversampling AL method failed to do so. These results show that the Proposed method increased the reliability of the new AL framework while improving the quality of the final classifier while requiring less data.

To the best of our knowledge, the method proposed in this paper was the first AL approach to consistently outperform the maximum performance threshold. Specifically, in Table 6, the performance of the classifiers originating from the proposed method was able to outperform classifiers trained using the full training dataset in all 12 scenarios except one. This shows that using a meaningful subset of the training dataset along 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 preprocessing method to further optimize classification performance.

This study introduces 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 amount 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 into 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 in 100% of scenarios. The exploration of other, more complex, data augmentation techniques might further improve its performance through the production of 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 into 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 found significant standard errors throughout our experimental results (see Subsection 6.1), which is consistent with the findings in [5, 11]. This 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.

7 Conclusion

Work in Progress (Conclusion).

FUTURE WORK: Data augmentation considering the global structure of the unlabeled dataset.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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, 7 2020.

[7] 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.

[8] 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, 2 2020.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] J. Zhou and S. Sun, "Improved margin sampling for active learning," *Communications in Computer and Information Science*, vol. 483, pp. 120–129, 11 2014.

[32] S. Behpour, K. M. Kitani, and B. D. Ziebart, "Ada: Adversarial data augmentation for object detection," *Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision, WACV 2019*, pp. 1243–1252, 3 2019.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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 2016*, Institute of Electrical and Electronics Engineers Inc., 12 2016. Shows domain-specific augmentation methods outperform generic data augmentation.

[38] 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.

[39] 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.

[40] 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.

[41] 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, no. 24, p. 3040, 2019.

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

[43] 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, Berlin, Heidelberg, 2005.

[44] 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, 10 2018.

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

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[57] 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.

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

[59] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, pp. 65–70, 1979.