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

Work in Progress (Introduction).

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, collecting large volumes of data may not be feasible depending on the domain, while the labeling of such data becomes labor and time intensive, while involving domain experts throughout the process [1].

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 [2]. 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 [3]:

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

2. Selection of an initial data subset. Typically, when there is no a priori labeled dataset, the initial data subset is randomly picked from the unlabeled dataset.

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

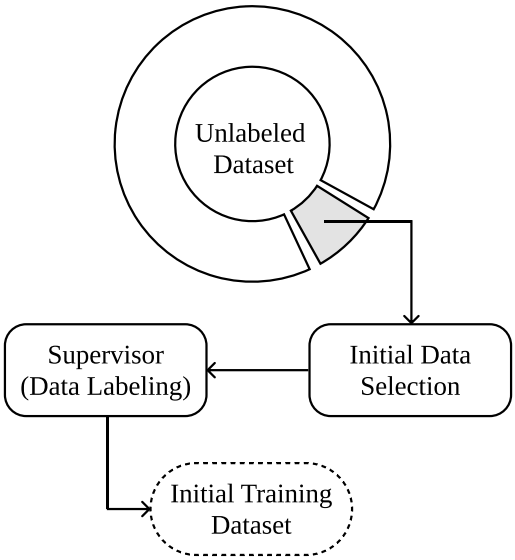

Figure 1: Active Learning initialization.

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 [4, 5]:

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.

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.

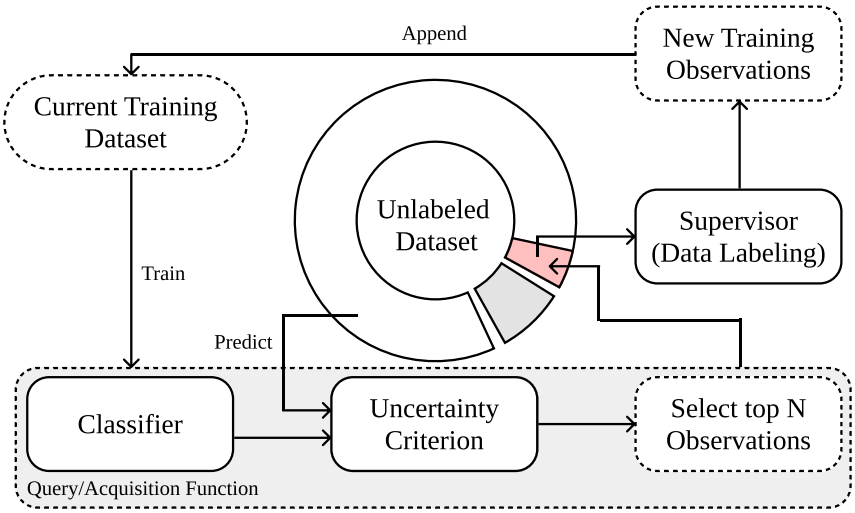

Figure 2: Active Learning iteration. In the first iteration, the training set collected during the initialization process becomes the "Current Dataset".

Two common challenges found in AL implementations is the consistency and efficiency of AL in practical scenarios [6]. 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.

Work in progress (Active Learning Methods — Lit. Review).

Reminders:

- Discuss maximum performance thresholds.

3 Data Augmentation Methods

Work in progress (Data Augmentation Methods — Lit. Review).

4 Proposed Method

Work in progress (Proposed Method)

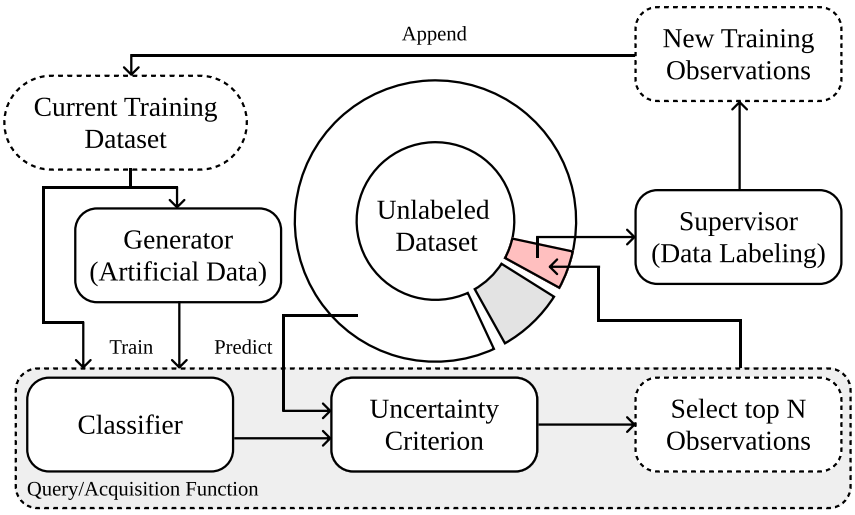

Figure 3: Active Learning — proposed iteration.

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 the optimal classification performance).

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 4. 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.

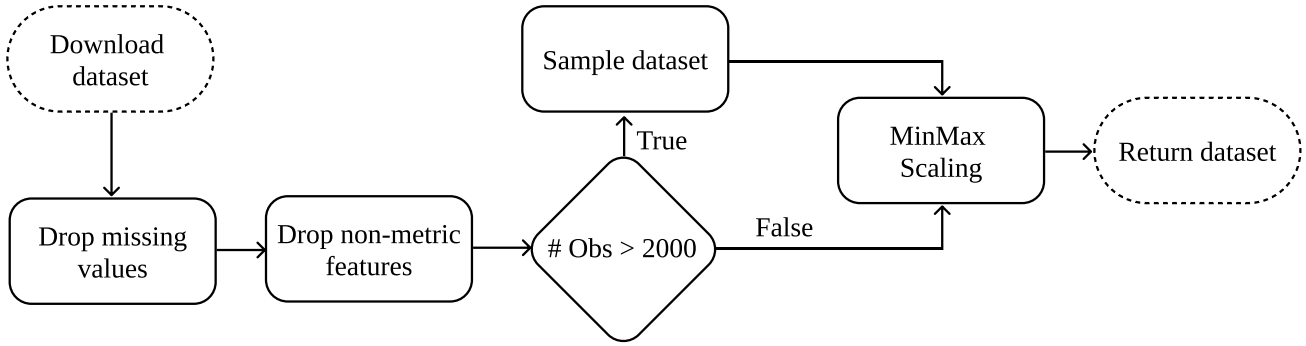

Figure 4: Data preprocessing pipeline.

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) [7], a K-nearest neighbors classifier (KNN) [8], a Random Forest Classifier (RF) [9] and a Logistic Regression (LR) [10]. 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 [3]. 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 [11]. 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 [12]. 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 [11, 13], 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 [13]. 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 [11]:

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

- 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 [6]. 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) [14]. 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 [15]. In this scenario, since the dataset is already labeled, the annotation process is done at zero cost. Figure 5 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 5 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).

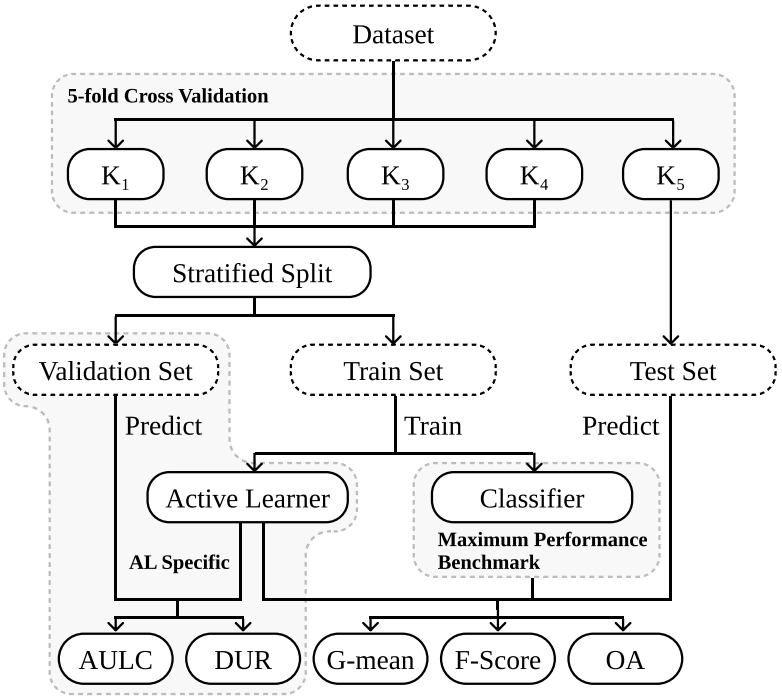

Figure 5: Experimental procedure flowchart.

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 [16], Imbalanced-Learn [17], Geometric-SMOTE [18], 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 [19]. 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.

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.

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.

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 6. A DUR below 1 means that the Proposed/Oversampling method requires less data than the Standard AL method to reach the same performance threshold. For example, running an AL 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).

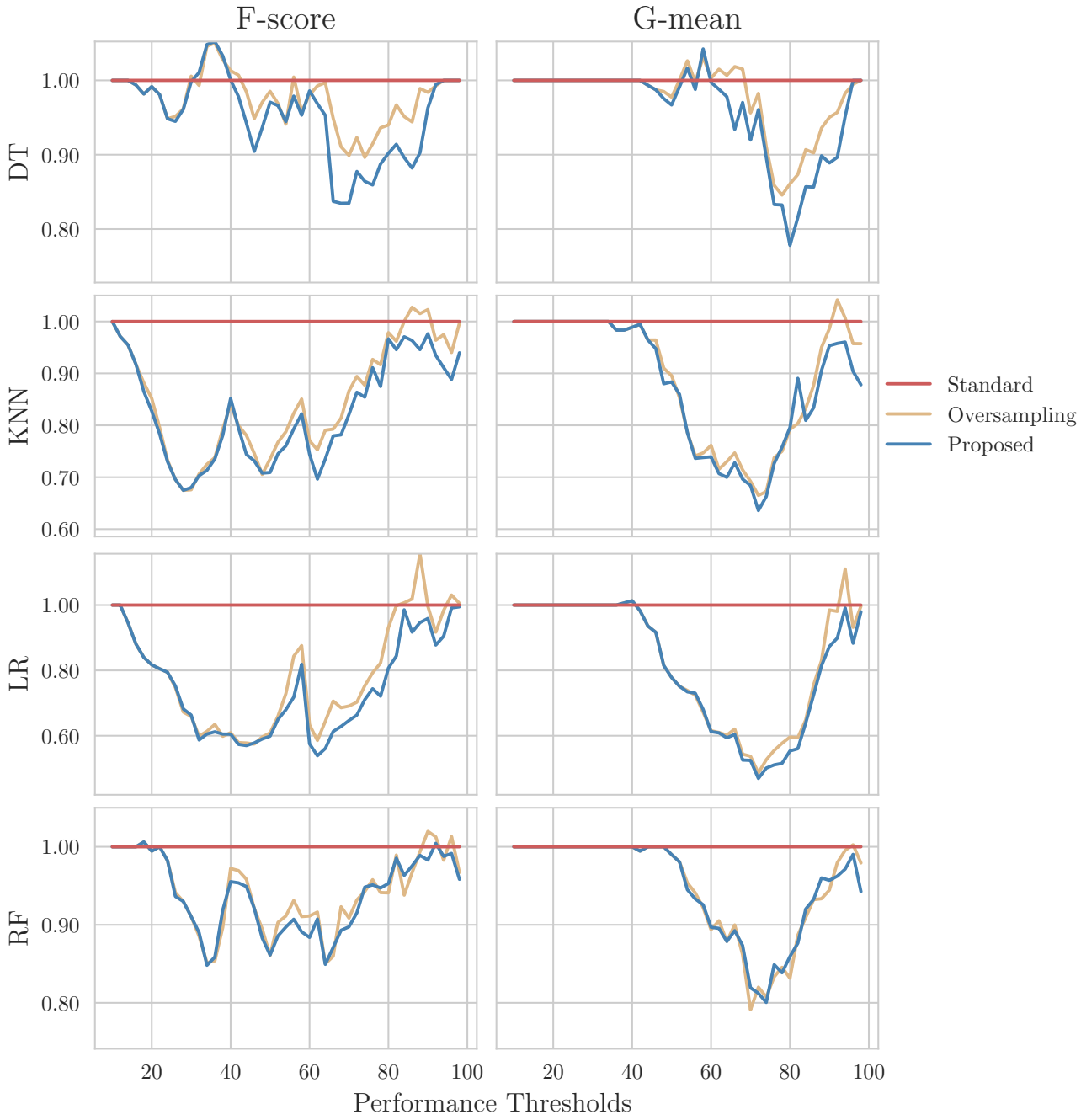

Figure 6: Mean data utilization rates. The y-axis shows the percentage of data (relative to the baseline AL framework) required to reach the different performance thresholds.

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 [19]. Overall, we perform 3 statistical tests. The Friedman test [20] 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 [21] 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 [22] 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 [3] 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 tested method does not perform better than the control method (benchmark 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 6. 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 [3, 6]. This indicates that the production 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

WIP: THE TEXT BELOW IS A DRAFT (i.e., just some notes I wrote to remind myself of important topics to discuss later)

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 [3]. We further extend this study by applying the AL framework discussed in a context agnostic environment, considering 10 different multiclass datasets from different

domains.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

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

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