Instance-Selection-Inspired Undersampling Strategies for Bias Reduction in Small and Large Language Models for Binary Text Classification

Guilherme Fonseca[1], Gabriel Prenassi[2], Washington Cunha[1], Marcos André Gonçalves[1], Leonardo Rocha[2]
[1]Universidade Federal de Minas Gerais, [2]Universidade Federal de São João del Rei,
{guilhermefonseca, washingtoncunha, mgoncalv}@dcc.ufmg.br,
prenassigabriel@aluno.ufsj.edu.br, lcrocha@ufsj.edu.br

Abstract

Skewness in imbalanced datasets affects Automatic Text Classification (ATC), leading to classifier bias toward the majority classes. This work examines undersampling methods to mitigate such bias in Small and Large Language Model (SLMs and LLMs) classifiers. Based on the limitations found in existing solutions, we propose two novel undersampling methods inspired by state-of-the-art **Instance Selection** techniques, relying on calibrated confidences and semantic difficulty estimates. We compare them against 19 baselines across 13 datasets, evaluating: (i) effectiveness, (ii) class imbalance bias, (iii) efficiency, (iv) scalability, and (v) consistency. Results show our methods uniquely reduce classifier bias (up to 56%) across all datasets without effectiveness loss while improving efficiency (1.6x speedup), scalability and reducing carbon emissions (up to 50%).

1 Introduction

Automatic Text Classification (ATC) has experienced a fast (r)evolution in recent years, led by advances in deep learning based on Transformers (Devlin et al., 2018; Liu et al., 2019; Touvron et al., 2023). These strategies have benefited from applications that constantly produce large volumes of labeled data (e.g., social networks).

Despite high effectiveness promoted by data abundance, Transformer-based classifiers, including Small and Large Language Models (SLMs and LLMs), can still be negatively impacted by issues such as class imbalance and model bias towards the majority class (in this paper, referred to as 'class imbalanced bias') (Cunha et al., 2020). In addition to ethical issues related to bias (Ferrer et al., 2021), there are several scenarios in which the minority class is indeed the class of interest (e.g., health).

An ATC task very influenced by class imbalance is sentiment analysis, our focus in this work. For products (Luiz et al., 2018) and points of interest (POI) (Werneck et al., 2021), users commonly make their consumption decisions by analyzing other users' comments. Some users take positive reviews more heavily (e.g., positive aspects of a hotel), while others emphasize negative comments (why not buying a particular product?). In both cases, the class of interest may be the minority one, and the classification results of a biased model may negatively influence the user's decision.

Experimental results (Appendix A and B) reveal that fine-tuned Transformer models – ranging from smaller language models (SLMs) such as RoBERTa (Liu et al., 2019), BART (Lewis et al., 2020) and BERT (Devlin et al., 2018)), to Large Language Models (LLMs) such as Llama3.1 (Dubey et al., 2024) – are still impacted by class imbalance. Indeed, although they demonstrate greater effectiveness and resilience to biases generated by class imbalance[1] compared to traditional classification algorithms (e.g., KNN, Random Forest, SVMs, XGBoost, as detailed in Appendix A), in highly imbalanced datasets - specifically when the ratio between the total number of documents in the majority class and the minority class exceeds five, the SLMs and LLMs model's still exhibit high bias values (detailed in Appendix B), with considerable room for improvement.

Two main approaches are typically employed to deal with data imbalance. **Oversampling** generates new synthetic samples for the minority class, aiming to balance it with the majority class (Han et al., 2005). In the ATC task, synthetic textual data generation presents specific challenges, such as ensuring linguistic quality and preserving semantic fidelity to the original data, making its application less straightforward than in other

[1]Measured in terms of TPRGap, a metric that captures the class imbalanced bias based on the absolute difference between the TPR - True Positive Rate - of the classes (Czarnowska et al., 2021).

domains (Douzas et al., 2022). This approach also leads to a considerable increase in model training time due to dataset expansion, posing relevant issues for current language models.

Undersampling (*US* in short), in turn, reduces instances of the majority class towards balancing. To the best of our knowledge, no previous study addresses how undersampling methods interact with SLMs and LLMs applied to ATC tasks. Thus, our **first** research questions, which we aim to empirically answer, are (*RQ1*): *Are undersampling methods, applied to SLMs and LLMs for ATC, capable of reducing classification bias caused by class imbalance? What is the impact of this combination on model effectiveness? Are the results consistent between SLMs and LLMs?*

To answer these questions, we carried out a systematic literature review of the main undersampling methods. We identified and implemented 14 popular undersampling methods. For completeness's sake, we expanded this set with 5 recently proposed methods. We investigated the performance of these **19** undersampling techniques in conjunction with RoBERTa, a SLM considered state-of-the-art in sentiment analysis (Cunha et al., 2024), and Llama3.1, an open-source LLM that has been widely used in recent research (Reis et al., 2024).

In our experiments, we observed that of all tested undersampling strategies, only 3 – Condensed Nearest Neighbors (CNN) (Hart, 1968), Near Miss 1 (NM1) (Mani and Zhang, 2003) and Near Miss 2 (NM2) (Mani and Zhang, 2003) – were able to reduce model bias without effectiveness losses. But, even these methods were not able to scale when applied to relatively moderate datasets.

Based on these findings, we propose two new undersampling techniques that seek to achieve five simultaneous objectives: *(i) to reduce classifier bias towards the majority class; while (ii) maintaining (or improving) effectiveness and being (iii) efficient; (iv) scalable for large datasets; and (v) consistent when applied to both SLMs and LLMs.* None of the previously identified US methods have been evaluated based on all five perspectives simultaneously – most studies typically consider only one or two of these aspects in their assessments. Motivated by the identified gaps, our novel proposals take inspiration from solutions from another NLP research area – Instance Selection (IS) (Cunha et al., 2023b, 2024), especially redundancy-oriented IS methods (Cunha et al., 2023a). Despite having different objectives, IS and US are related, as both deal with techniques aimed at selecting a subset of representative (training) data.

Our first proposed strategy, **E2SC_US**, is based on a state-of-the-art IS approach (Cunha et al., 2023a) and aims to remove redundant (i.e., very similar to other) training instances from the majority class. We assess redundancy in the majority class according to the degree of confidence associated with classifying training instances, using a calibrated classifier (Rajaraman et al., 2022). Our second strategy, called *Undersampling Based on Redundance* (**UBR**), also removes redundant instances from the majority class. Different from E2SC_US, UBR explores the *semantic difficulty* of an (training) instance being correctly classified (Mujumdar et al., 2023).

Accordingly, our second main research question is *RQ2: How do the new proposed undersampling methods – E2SC_US and UBR – when applied to SLMs and LLMs for ATC – compare to the literature methods, considering: (i) effectiveness; (ii) class imbalance bias reduction; (iii) efficiency; (iv) scalability; and (v) consistency*?

Our experiments, encompassing 21 undersampling methods, 13 datasets, and 5 evaluation criteria, demonstrate that our methods are the only ones able to achieve the five objectives simultaneously, with a slight advantage for UBR in terms of bias reduction and for E2SC_US in speedup, especially when considering large datasets. Indeed, our methods were the only ones capable of executing on all datasets, including large ones, reducing the total execution time of LLM-based classification algorithms without effectiveness losses and with a significant bias reduction. Another positive side effect of our solutions is the promotion of Green Computing by significantly reducing carbon emissions.

In sum, the main contributions of this work are:

- A systematic mapping of the literature on US methods, identifying, implementing and evaluating 14 popular and 5 recent methods (19 in total);
- Two novel undersampling strategies inspired by state-of-the-art Instance Selection techniques;
- A comprehensive evaluation of our new US proposals applied to SLM and LLM-based classifiers from five perspectives: (1) effectiveness; (2) efficiency (time); (3) generalizability (class imbalance bias); (4) scalability; and (5) SLMs-LLMs consistency.

2 Related Work

We turned to the Google Scholar search engine to submit the query and generate our initial set of articles. Google Scholar was chosen due to its broad coverage, including major digital libraries from publishers such as ACM, IEEE, and Elsevier, as well as preprint repositories like ArXiv. The search string used was "Undersampling". To maximize the scope of the search, the search engine did not apply any location or year filters. Based on this, we initially collected a total of 500 unique articles that, in some way, applied some undersampling strategy. We manually analyzed the 500 articles to identify the most relevant ones for our study. An article was considered relevant if it used undersampling techniques to reduce imbalance and the specific undersampling method was explicitly referenced (cited). We identified 139 relevant articles and listed all the utilized undersampling techniques, finding a total of 31 distinct techniques. A comprehensive table with descriptions of all identified strategies is available online[2]. In our evaluation, we chose to consider those methods used in more than one of the relevant works, which resulted in the 14 methods described below:

- **Tomek Links (TL)** (Tomek, 1976b): Given two examples e_i and e_j of distinct classes, with $d(e_i, e_j)$ representing the distance between e_i and e_j, a pair $A(e_i, e_j)$ is called a Tomek link if there is no example e_l such that $d(e_i, e_l) < d(e_i, e_j)$ or $d(e_j, e_l) < d(e_i, e_j)$. If two examples form a Tomek link, either one was manually misclassified or both belong to class boundaries and can be removed.
- **Condensed Nearest Neighbors (CNN)** (Hart, 1968): Dataset S is initialized with one example from the majority class and all examples from the minority one, Set T is created with elements not belonging to S. Each example of T is classified by KNN using S as training set. If KNN gets the example class right, it remains in T; otherwise, the example is removed from T and placed in S. This process repeats until no more changes to S occurs. In the end, elements of T are discarded.
- **One-Sided Selection (OSS)** (Kubat et al., 1997): It combines TL and a CNN variation. As in CNN, a set S is initialized with all minority class instances and one of the majority class and a set T with the rest of the elements. Then, instances in T are classified with a KNN trained in S, and misclassified instances are placed in S. In the end, TL is used in S to identify ambiguous pairs at class boundaries.
- **Edited Nearest Neighbours (ENN)** (Wilson, 1972): inserts all instances of the original set T into the solution set S, using KNN iteratively to classify all instances x given that $x \in S$ and that x belongs to the majority class (considering tset $\{S - \{x\}\}$ as possible neighbors). Finally, it removes the incorrectly classified instances.
- **Repeated Edited Nearest Neighbours(RENN)** (Tomek, 1976a): ENN applied successively until no more points can be removed.
- **ALL k-NN** (Tomek, 1976a): ENN applied successively, but with each application, the number of neighbors to be considered increases.
- **Neighbourhood Cleaning Rule (NCR)** (Laurikkala, 2001): Uses KNN to classify all instances. If the predicted class is different from the real class and the instance belongs to the majority class, it is eliminated. NCR also removes the nearest neighbors of misclassified instances of the minority class from the majority class.
- **Near Miss (NM)** (Mani and Zhang, 2003): NearMiss-1 (**NM1**) removes majority class instances with the smallest average distance to (k instances of) the minority class. NearMiss-2 (**NM2**) selects majority class elements whose average distance to the k furthest points from the minority class is the lowest. NearMiss-3 (**NM3**) keeps k majority class instances closest to the minority class.
- **(SBC)** (Yen and Lee, 2006): The training is divided into N clusters. For each cluster, the number of selected instances is calculated based on the number of majority and minority classes samples that exist in the cluster. Examples from the majority class are randomly selected and the algorithm combines the selected instances from each cluster with those from the minority class to form a new set.
- **(IHT)** (Smith et al., 2014): It uses a classifier c to obtain the instance hardness (IH) of each instance. The IH of an instance is given by $IH(< x_i, y_i >) = 1 - p(y_i|x_i, c)$ where $p(y_i|x_i, c)$ denotes the classifier´s probability of instance x_i belonging to class y_i. IHT removes samples from the majority class with a low probability of belonging to the majority class.
- **(CC-NN)** (Lin et al., 2017): Majority class instances are divided into N clusters, with N being the minority class size. The closest neighbor to the centroid of each cluster belonging to the majority class is chosen to compose with the instances of the minority class, the final set.

[2] https://github.com/guilherme8426/ACL2025_Undersampling

- (OBU) (Vuttipittayamongkol et al., 2018): Uses Fuzzy c-means to divide data into 2 clusters. The one with most instances of the minority class is called CM. The algorithm removes all instances of the majority class whose degree of membership in CM is less than α (a hyperparameter).

Undersampling in ATC has been under-studied in recent years, likely due to limited understanding of its interaction with Transformer-based language models, which typically benefit from more data. Nonetheless, we included the recent methods discussed below in our analyses and experimentation.

AKCS (Zhou and Sun, 2024) clusters majority-class instances using an adaptive k-means to identify a dataset optimal k. Instances furthest from the centroids are discarded. In (Kumar et al., 2024), four algorithms — **ENUB, ENUT, ENUC, and ENUR** — eliminate majority-class instances in overlapping regions where different classes share the same space, based on entropy. The methods differ in how KNN identifies instances for removal.

Time complexity for all the above methods can be found in Table 16 (Appendix I).

Differently from previous work, our novel proposals aim at producing **practical** solutions for the class imbalance bias problem that *simultaneously* preserve classification effectiveness (as most of the above methods *do not*), and can be run fast in large datasets with reasonable computational resources. None of the previous studies evaluate their methods considering all the aforementioned criteria, focusing on only one or two of them in their analyses.

3 Proposed Undersampling Methods

We present two novel approaches to undersampling inspired by solutions for the *Instance Selection (IS) problem (Cunha et al., 2023b, 2024; Pasin et al., 2024)*. IS and undersampling (US) deal with techniques aimed at selecting a subset of training data. They target, however, different goals – IS focuses on improving efficiency without effectiveness loss, while the US aims at reducing the majority class bias while maintaining effectiveness.

Although the final objectives are different, we depart from the hypothesis that there is an underlying relationship between both tasks, especially for IS methods based on redundancy reduction (Cunha et al., 2023b), which can be adapted to remove redundant majority class instances [3].

[3]We focus on binary classification problems leaving extensions to multi-label problems for future work.

3.1 E2SC_US

The E2SC (Cunha et al., 2023a) IS method works in two steps. First, it calculates the probability of each instance being removed from training. These probabilities are obtained through the confidence of a calibrated classifier (Rajaraman et al., 2022)[4], in case, KNN. Its main hypothesis is that high confidence of a calibrated classifier positively correlates with redundancy in the training data for the sake of building a classification model. Accordingly, in the next step, E2SC randomly samples the training set, weighted by confidence, keeping mostly hard-to-classify instances (probably located in the decision borders), and partially removing the easiest ones. To estimate the optimal reduction rate (second step), training instances are randomly sampled and weighted by the assigned probabilities.

Our first proposal consists basically of two modifications of E2SC, called E2SC_US (pseudocode in **Algorithm 1**), following the principles of the original approach, but using logistic regression (LR) in place of KNN. LR, a more calibrated and faster classifier (details in Appendix G), is used to obtain the removal probability of each instance (lines 6 and 7). In the second modification, instead of removing a proportion (β) of instances of all classes, E2SC_US only removes majority class instances at a pre-fixed removal rate defined as $\text{MIN}(0.5, (\#maj - \#min)/(\#maj + \#min))$. A maximum of 50% of removal or up to achieving the minority class size (lines 9 and 10) is the defined limit. The 50% removal limit is based on (Cunha et al., 2023a), which determined this as the empirical reduction limit, after which it is not possible to avoid effectiveness losses. Experiments varying the maximum removal rate are in Appendix J.

Algorithm 1: E2SC_US Algorithm

```
Input: X
Output: selectedInstance
1  $X_{Maj} \leftarrow getInstances(X, class = majority)$;
2  $X_{Min} \leftarrow getInstances(X, class = minority)$;
3  $classifier \leftarrow logisticRegression(X)$;
4  $trust \leftarrow \{\}$ ;
5  for $x_i \in X_{Maj}$ do
6      $trust \leftarrow trust \bigcup getTrust(x_i, classifier)$ ;
7  end
8  $\alpha \leftarrow normalize(trust)$ ;
9  $N \leftarrow min(\|X_{Maj}\| - \|X_{Min}\|, 0.5\,\|X\|)$
   $selected \leftarrow randomSampler(X_{Maj}, \alpha, N)$ ;
10 $selectedInstance \leftarrow X_{Min} \bigcup selected$ ;
```

[4]Classifier whose class probability predictions correlate well with accuracy.

3.2 UBR

UBR (Undersampling Based on Redundancy), our second approach, is also based on redundancy. For this proposal, instead of using a calibrated classifier to estimate the removal probabilities of training instances, we adapt the concept of semantic difficulty (SD) of an instance being correctly classified (Mujumdar et al., 2023) and use this to estimate the likelihood of an instance also being redundant. We hypothesize that instances with low semantic difficulty are redundant and, thus, good candidates for removal from the majority class. Our method uses SD to estimate a distribution $\alpha(x)$ that assigns a probability of x_i being removed.

In (Mujumdar et al., 2023), difficult instances are defined as: (i) samples with high semantic similarity and different labels within their neighborhood and (ii) samples with low (average) semantic similarity with their neighborhood, but with neighbors belonging to the same class. Given a set of documents $X = \{x_1, x_2, ..., x_M\}$ and set of classes $Y = \{y_1, y_2, ..., y_K\}$, the semantic difficulty metric (SD) of x_i is defined as $SD(x_i) = \sum_{j \epsilon M} fp(cos_{sim}(x_i, x_j))$. with $cos_{sim}(x_i, x_j)$ being the cosine similarity between x_i and x_j. fp is a penalty function (Equation 1), responsible for penalizing difficult instances according to the definitions (i) $y_i = y_j$ and (ii) $y_i \neq y_j$. fp penalizes using a *z-shaped sigmoid* if $y_i = y_j$ and a *s-shaped sigmoid* otherwise.

$$fp(x) = \begin{cases} \frac{1}{1+e^{-(5-10x)}} & , y_i = y_j \\ \frac{1}{1+e^{(5-10x)}} & , y_i \neq y_j \end{cases} \quad (1)$$

More formally, in UBR (pseudocode in **Algorithm 2**), we receive as input a set X of training instances, dividing it into two subsets, X_{Maj} (line 1) and X_{Min} (line 2), where X_{Maj} consists of X instances belonging to the majority class and X_{Min} of X instances belonging to the minority one. We create a vector D with the SD of each instance of X_{Maj} based on its K nearest neighbors. For time optimization and memory usage issues, we use an approximate KNN (Ponomarenko et al., 2014) (lines 4-7). The distribution $\alpha(x)$ is obtained by inverting and normalizing D so that elements with a high value have a low probability in $\alpha(x)$ and elements with a low value in D have a high removal probability (line 8-9). In the end, we sample $N = \text{MIN}(\|X_{Maj}\| - \|X_{Min}\|, 0.5\,\|X\|)$ from X_{Maj}, weighted by the probability distribution $\alpha(x)$ (line 10). The new training set is composed of X_{Maj} elements not selected for removal, together with X_{Min} elements (line 11). Time complexity for our proposals is found in Appendix I.

Algorithm 2: UBR Algorithm

```
   Input: X
   Output: selectedInstance
1  $X_{Maj} \leftarrow getInstances(X, class = majority)$;
2  $X_{Min} \leftarrow getInstances(X, class = minority)$;
3  $D \leftarrow \{\}$ ;
4  for $x_i \,\epsilon\, X_{Maj}$ do
5      $nn \leftarrow nearestNeighbors(x_i, X)$ ;
6      $D \leftarrow D \bigcup calculateDS(x_i, nn)$ ;
7  end
8  $\alpha \leftarrow normalize(D)$ ;
9  $N \leftarrow min(\|X_{Maj}\| - \|X_{Min}\|, 0.5\,\|X\|)$
    $selected \leftarrow randomSampler(X_{Maj}, \alpha, N)$ ;
10 $selectedInstance \leftarrow X_{Min} \bigcup selected$ ;
```

4 Experimental Setup

We consider 13 datasets[5] with varying imbalance levels. Table 1 shows the datasets along with the total number of documents belonging to the majority and minority classes, dataset identifier, and imbalance ratio (IR), defined as (Orriols-Puig and Bernadó-Mansilla, 2009), $IR = \frac{class\ majority}{class\ minority}$ - the higher the IR, the more imbalanced the dataset. All adopted datasets have two classes, as this work focuses on binary classification problems.

dataset	Size	# Majority	# Minority	IR	Identifier
sentistrength_twitter	2,289	1,340	949	1.41	A
vader_amazon	3,610	2,128	1,482	1.44	B
english_dailabor	1,227	739	488	1.51	C
debate	1,979	1,249	730	1.71	D
sentistrength_youtube	2,432	1,665	767	2.17	E
vader_twitter	4,196	2,897	1,299	2.23	F
tweet_semevaltest	3,060	2,223	837	2.66	G
sentistrength_digg	782	572	210	2.72	H
sentistrength_myspace	834	702	132	5.32	I
sentistrength_bbc	752	653	99	6.60	J
luxury_beauty	30,394	27,803	2,591	10.73	K
cds_reviews	1,333,070	1,243,212	89,858	13.84	L
digital_music	162,989	158,985	4,004	39.71	M

Table 1: Datasets used in the experiments. Column "Identifier" is the reference to be used for the dataset.

We use SLMs and LLMs classifiers. SLM representative is **RoBERTa** (Liu et al., 2019), considered SOTA in sentiment analysis (Cunha et al., 2025; Zanotto et al., 2021). We carried out an experimental comparison with two other SLMs, BART and BERT, with RoBERTa standing out (see Appendix A). For LLMs, we employ **Llama3.1** (Llama-3.1-8B), an open-source LLM that has been widely used in NLP research (Reis et al., 2024). Effectiveness comparison between Llama3.1 and RoBERTa is found in Appendix B. We fine-tuned both SLMs and LLMs, adapting the pre-trained models to each dataset domain by utilizing the texts and training data labels.

[5]Datasets A–J in (Ribeiro et al., 2016) and datasets K–M in `https://cseweb.ucsd.edu/~jmcauley/datasets/`

Fine-tuning involves learning an appended fully connected head layer (Dense) that captures the label distribution, connecting the "CLS" token representations with labels to perform ATC. Models hyperparameters are found in Appendix H.

Our assessment considers five perspectives: (1) classification effectiveness; (2) model bias towards the majority class; (3) efficiency (time and CO_2 emissions); (4) scalability; and (5) consistency between SLMs and LLMs. Effectiveness is evaluated using Macro Average F1 (MacroF1), which is more suitable for skewed tasks than MicroF1 (a.k.a. accuracy). Class imbalance bias is captured by (Czarnowska et al., 2021): $\text{TPRGap} = \sum_{i,j \in T} \frac{|\text{TPR}(i) - \text{TPR}(j)|}{N}$ where $\text{TPR}(i)$ is the true positive rate of class i, T is the number of classes, N is a normalization equal to the number of pairs of compared classes $\binom{T}{2}$. Efficiency is measured based on the cost of each method in terms of the total time required to build the model and perform classification. Speedup is calculated as $S = \frac{T_{wo}}{T_w}$, where T_w is the total time spent building the model plus the classification time, using some undersampling approach, and T_{wo} is the total time spent on execution (model and classification) without the undersampling phase. Emission of CO_2 is the estimated value of greenhouse gases, converted to their equivalent amount of carbon dioxide, spent for training a model and classification, calculated based on (Lannelongue et al., 2021). Scalability is related to the adequate use of computational resources (i.e., memory, CPU, GPU) and the ability to deal with large datasets. To assess consistency, we compare the results between SLMs and LLMs, considering the other four previous perspectives.

Hardware used in the experiments is detailed in Appendix E. Datasets were divided using cross-validation with 5 (K to M) or 10 partitions (others). Larger datasets were divided into fewer partitions due to the costs of the training process, especially for LLMs. Statistical comparisons were performed using a T-Test with Bonferroni correction. Codes found at `https://github.com/guilherme8426/ACL2025_Undersampling`.

5 Experimental Results

We divided the analyses of results into five perspectives: (1) effectiveness, (2) class imbalance bias reduction, (3) efficiency, (4) scalability, and (5) consistency of results between SLMs and LLMs. We start by discussing the results of the first three analyses, considering only the smaller datasets (with less than 30,000 instances (A-J)) together with the SLM (RoBERTa) classifier. We consider our methods and all 19 baselines in these analyses. Due to space limitations, we present the detailed discussion of the results of AKCS, ENUB, ENUT, ENUC, and ENUR in Appendix C since they did not perform satisfactorily regarding effectiveness and bias reduction.

For (4) and (5), we include the results of the Llama3.1 classifier, considering also the three largest datasets. Due to cost issues, we consider only our proposals and the best baselines based on the SLM results in these last analyses. In any case, the LLM results for all 5 evaluation perspectives are found in Tables 5 and 6, all discussed in Section 5.4 when compared with the SLM. The large datasets results for RoBERTa are in Appendix F.

5.1 Effectiveness

In Table 2, we present the effectiveness results obtained by applying US methods together with the RoBERTa classifier. The "NoUnder" column presents the result without undersampling. For all methods that allow hyperparameterization regarding the number of instances to be removed (UBR, E2SC_US, NM1, NM2, IHT and CC_NN), we limit the removal to a maximum of 50%, as recent IS work (Cunha et al., 2023a) points out that this is the empirical limit of possible reduction without loss of effectiveness. The other methods were not modified, following their own removal policy.

Considering only the smaller datasets (A-J), we observe that only half of the 16 tested methods achieve a statistical tie with the classification without *undersampling* (i.e., with the complete imbalanced training) in all analyzed datasets. This includes our two proposals – UBR and E2SC_US – and methods CNN, NM1, NM2, CC_NN, TL and OSS. These techniques balance the dataset without losing effectiveness. Other methods did not obtain good results, causing losses in 3 (IHT, OBU), 2 (SBC, RENN and ALLKNN) and 1 (NM3, NCR, ENN) dataset(s), respectively. A detailed F1 behavior analysis for each class after applying the US methods is found in Appendix L.

5.2 Class imbalance Bias Reduction

In Table 3, we present the results for the TPRGap metric, which measures models' bias. The "NoUnder" column presents the result without undersampling, while the others present the TPRGap of the models with undersampling. The background colors of the cells represent how much

dataset	NoUnder	CNN	NM1	NM2	CC_NN	SBC	NM3	OBU	NCR	IHT	TL	OSS	ENN	RENN	ALLKNN	UBR	E2SC_US
A	88.6(0.7)	88.2(1.5)	88.9(0.8)	89.0(0.8)	88.7(1.4)	87.6(1.5)	88.5(1.0)	85.1(1.4)	87.2(2.0)	87.7(1.2)	**89.2(1.2)**	88.3(0.6)	85.6(0.9)	85.6(0.9)	87.2(2.1)	88.9(1.5)	88.8(1.1)
B	89.0(0.7)	88.1(2.0)	88.3(1.2)	88.2(0.7)	**90.0(1.2)**	87.9(1.2)	51.3(13.5)	85.6(1.4)	89.4(1.4)	87.8(1.5)	88.7(0.9)	88.7(0.9)	83.4(9.0)	83.1(8.9)	70.2(5.1)	88.4(1.0)	89.1(1.1)
C	93.3(1.1)	93.7(1.1)	94.2(1.5)	94.1(1.1)	**94.5(1.4)**	89.3(3.3)	93.9(1.4)	92.3(1.6)	92.4(1.4)	92.0(1.2)	93.4(1.7)	94.3(1.5)	92.5(1.7)	92.5(1.7)	91.7(2.0)	94.3(1.0)	93.4(1.3)
D	**89.3(1.2)**	88.2(1.0)	88.0(1.5)	86.3(1.5)	81.7(13.8)	87.7(1.2)	87.7(2.0)	80.7(1.6)	86.7(1.5)	84.3(3.2)	88.4(1.5)	89.1(1.4)	83.4(2.1)	83.4(2.1)	83.6(3.4)	88.8(1.2)	88.7(1.7)
E	89.7(1.9)	89.6(1.5)	88.4(1.8)	89.3(1.9)	89.3(1.6)	79.2(4.2)	89.0(1.7)	88.2(2.3)	68.7(7.2)	82.3(1.4)	**89.9(1.6)**	**89.9(1.6)**	55.2(3.5)	55.2(3.5)	58.6(3.5)	89.3(1.5)	89.4(1.7)
F	**94.2(1.0)**	93.0(1.0)	92.6(1.1)	92.5(1.2)	93.0(1.0)	92.0(0.9)	92.9(1.3)	86.7(2.4)	93.2(1.0)	87.9(1.1)	93.4(1.4)	93.8(1.1)	92.9(0.8)	91.5(1.0)	93.1(0.9)	93.1(1.3)	93.1(1.2)
G	90.1(1.5)	89.8(1.8)	89.7(2.1)	89.2(1.8)	88.8(1.5)	88.1(0.9)	88.9(1.6)	88.3(1.7)	90.1(1.5)	83.0(2.0)	90.3(1.1)	**90.6(1.6)**	89.2(1.3)	86.8(1.9)	88.7(1.8)	89.0(1.0)	88.5(1.5)
H	83.8(5.0)	83.0(5.0)	80.5(4.3)	81.3(4.1)	82.4(4.6)	81.0(4.6)	81.0(3.9)	77.6(3.3)	84.0(4.9)	74.1(4.5)	**87.2(4.7)**	85.4(3.7)	81.2(5.3)	75.8(4.4)	77.3(5.5)	82.2(3.6)	80.6(4.7)
I	83.2(3.4)	81.5(3.6)	81.0(4.7)	80.8(5.5)	79.9(9.5)	60.2(2.9)	79.4(5.3)	81.8(6.0)	**84.5(4.6)**	74.5(3.7)	83.2(4.8)	**84.5(4.6)**	84.1(3.3)	80.7(3.9)	82.4(4.1)	83.4(4.6)	81.5(5.2)
J	**81.0(4.5)**	77.8(4.3)	78.0(4.1)	75.0(5.0)	76.7(3.7)	71.0(4.5)	74.0(3.2)	71.7(4.4)	79.8(5.1)	73.3(4.7)	78.7(4.6)	77.2(5.0)	77.8(5.9)	79.4(5.0)	77.4(6.1)	77.9(4.6)	78.7(4.2)

Table 2: RoBERTa MacroF1 using US. **Bold** cells are the largest numeric values for a dataset. Green cells represent results statistically equivalent to not using undersampling (NoUnder). Numbers in parentheses represent 95% confidence intervals.

dataset	NoUnder	CNN	NM1	NM2	CC_NN	SBC	NM3	OBU	NCR	IHT	TL	OSS	ENN	RENN	ALLKNN	UBR	E2SC_US
A	0.063	0.047	0.004	0.005	0.011	0.039	0.011	0.114	0.077	0.072	0.035	0.036	0.141	0.141	0.098	0.015	**0.002**
B	0.065	0.027	0.026	0.013	0.023	**0.012**	0.692	0.100	0.048	0.075	0.060	0.060	0.166	0.188	0.445	0.028	0.034
C	0.041	0.002	0.003	0.018	0.011	0.149	0.022	0.019	0.038	0.059	0.029	0.025	0.061	0.061	0.062	0.008	**0.000**
D	0.076	0.006	0.043	0.019	0.070	0.013	0.020	0.097	0.032	0.107	0.052	0.042	0.143	0.143	0.108	**0.002**	0.019
E	0.096	0.032	0.012	0.010	0.024	0.245	0.026	0.023	0.403	0.197	0.093	0.090	0.634	0.634	0.593	0.002	**0.001**
F	0.160	0.012	0.003	**0.001**	0.012	0.017	0.012	0.094	0.018	0.122	0.063	0.062	0.004	0.037	0.007	0.002	0.006
G	0.102	0.016	**0.000**	0.021	0.024	0.035	0.001	0.007	0.028	0.158	0.085	0.077	0.007	0.072	0.021	0.013	0.006
H	0.190	**0.006**	0.027	0.036	0.037	0.029	0.063	0.066	0.018	0.212	0.118	0.131	0.046	0.158	0.136	0.026	0.040
I	0.333	0.112	0.176	0.193	0.247	0.352	0.111	0.264	0.256	**0.085**	0.336	0.304	0.262	0.216	0.248	0.126	0.222
J	0.350	0.199	0.209	0.210	0.329	**0.047**	0.064	0.173	0.237	0.059	0.361	0.400	0.324	0.287	0.324	0.184	0.215
Mean	0.148	0.046	0.050	0.053	0.079	0.094	0.102	0.096	0.116	0.114	0.123	0.123	0.179	0.194	0.204	0.041	0.054

Table 3: TPRGap of the models generated by the RoBERTa classifier in conjunction with undersampling approaches. The greener the cell, the greater the bias reduction. The redder it is, the greater the increase in bias.

the US methods were able to reduce the model bias compared to "NoUnder". The greener, the greater the bias reduction; the redder, the greater the bias.

Focusing again on average bias in the smaller datasets, the methods that achieved the highest bias reduction in all ten datasets were UBR (average bias of 0.041) with a reduction of approximately 3.6 times compared to NoUnder (0.148), followed by CNN (0.046), NM1 (0.050), NM2 (0.053) and E2SC_US (0.054). CC_NN (0.079), despite having a smaller average bias than NoUnder, presents a smaller reduction compared to the other methods, in addition to presenting a bias practically equal to NoUnder in 2 datasets (D and J). Despite good effectiveness, TL and OSS demonstrate low performance concerning bias, worsening the model in 1 and 2 datasets, respectively, and having a total average TPRGap close to NoUnder – 0.123 each.

Analyzing beyond average values, we observed that E2SC_US presents the best result in 3 out of 10 datasets, while CNN, NM1, NM2, and UBR stand out in only one. However, when comparing the methods pairwisely, we found that UBR outperforms E2SC_US, CNN, NM1 in 6 of 10 datasets and NM2 in 7. UBR stands out in bias reduction, presenting the best average and individual results.

As a final remark, Table 11 (Appendix D) shows the imbalance ratio obtained after applying our methods. A perfect class balance is achieved in 8 out of 13 datasets. Datasets I to M do not reach a perfect IR because we limited the reduction to 50%.

5.3 Efficiency

We analyze US methods concerning their efficiency, verifying the impact of the new pre-processing step on the total processing time (model training + classification). Table 4 presents the speedup produced by the US methods, calculated as the ratio of the total time using some undersampling approach to the total time without the US phase.

We observe in Table 4 that UBR, CNN, NM1, NM2, and E2SC_US, methods that in previous analyses proved superior in effectiveness and class imbalance bias reduction, also achieve good efficiency. The speedups achieved were 1.434, 1.528, 1.479, 1.384, and 1.557, respectively, which were very good. Together with SBC (1.865) and NM3 (1.670), these are the methods with the highest speedup gains. We should remind, however, that both – NM3 and SBC – generate losses in effectiveness for some datasets (Table 2). Among the methods capable of reducing bias without effectiveness losses, E2SC_US is the one that obtained the highest speedup, being the highlight from this perspective. Appendix K further discusses the performance gains achieved by our US methods. Appendix M presents Table 4 with 95% confidence intervals.

5.4 Consistency and Scalability

As seen, considering the SLM, from 16 methods, only 5 were able to reduce class imbalance bias without effectiveness losses and with efficiency gains in the smaller datasets – our two new methods (UBR and E2SC_US) and three from the literature (CNN, NM1, and NM2). We will concentrate on these five methods for the LLM analyses. Table 5

dataset	CNN	NM1	NM2	CC_NN	SBC	NM3	OBU	NCR	IHT	TL	OSS	ENN	RENN	ALLKNN	UBR	E2SC_US
A	1.243	1.076	1.135	1.193	1.216	1.096	1.374	1.210	1.181	0.929	0.961	**1.415**	1.376	1.230	1.149	1.094
B	1.005	1.163	1.114	1.186	1.364	**1.588**	1.342	0.988	1.230	0.962	0.990	1.476	1.513	1.452	1.192	1.273
C	1.361	1.237	1.096	1.202	**1.638**	1.247	1.214	1.223	1.115	0.873	0.885	1.070	1.039	1.120	1.048	1.400
D	1.519	1.469	1.450	1.500	1.637	1.493	1.282	1.516	1.402	1.185	1.150	**1.966**	1.854	1.447	1.557	1.712
E	1.176	1.312	1.312	1.467	**1.769**	1.400	1.381	1.607	1.182	0.937	0.948	1.663	1.611	1.641	1.384	1.569
F	1.310	**1.530**	1.374	1.521	1.444	1.414	1.329	1.122	1.298	0.946	0.995	1.270	1.220	1.159	1.277	1.518
G	1.494	1.659	1.619	1.867	1.728	1.757	1.607	1.319	1.668	1.055	1.067	1.395	1.771	1.587	1.806	**1.950**
H	1.510	**1.923**	1.601	1.612	1.564	1.519	1.331	1.401	1.634	1.034	1.014	1.537	1.516	1.751	1.728	1.635
I	2.283	1.730	1.523	1.608	**2.861**	2.128	1.302	0.877	1.535	1.003	0.800	0.990	0.998	0.977	1.622	1.617
J	2.377	1.691	1.621	1.442	**3.433**	3.054	1.411	1.487	1.602	1.107	0.972	1.335	1.220	1.341	1.577	1.802
Mean	1.528	1.479	1.384	1.460	1.865	1.670	1.357	1.275	1.385	1.003	0.978	1.412	1.412	1.370	1.434	1.557

Table 4: Speedup results in the total cost (time) for generating the models using the RoBERTa classifier in conjunction with undersampling approaches. The greener the cell, the greater the reduction in total training time compared to the approach without undersampling. The redder, the longer the time.

Size	dataset	Macro-F1						TPR-Gap						SpeedUp				
		NoUnder	CNN	NM1	NM2	UBR	E2SC_US	NoUnder	CNN	NM1	NM2	UBR	E2SC_US	CNN	NM1	NM2	UBR	E2SC_US
Small	A	91.9(1.0)	90.6(1.7)	91.1(1.3)	91.6(1.5)	91.8(1.2)	**92.1(1.4)**	0.038	0.034	**0.006**	0.021	**0.006**	0.013	**1.390**	1.198	1.198	1.190	1.196
	B	93.2(0.8)	92.7(1.5)	90.9(5.1)	92.8(0.6)	**93.5(0.7)**	92.8(0.6)	0.028	0.030	**0.005**	0.027	0.010	0.006	**1.214**	1.210	1.210	1.208	1.208
	C	97.6(1.1)	97.6(0.7)	97.4(1.2)	**98.2(0.7)**	97.9(1.1)	98.0(0.9)	0.017	0.007	0.005	0.008	**0.000**	0.006	**1.502**	1.241	1.242	1.242	1.240
	D	**91.2(1.0)**	91.0(1.0)	90.6(1.3)	87.8(1.7)	91.0(1.6)	90.8(1.7)	0.076	0.028	**0.019**	0.044	0.022	0.028	1.306	1.333	**1.335**	1.332	1.334
	E	**93.9(1.4)**	92.9(1.5)	91.4(1.8)	91.5(1.8)	92.0(1.7)	92.4(1.6)	0.065	0.008	0.032	0.003	0.018	**0.002**	1.338	**1.557**	1.554	1.544	1.548
	F	**95.2(0.7)**	94.5(1.3)	94.0(1.0)	94.1(0.8)	94.3(1.0)	94.9(1.0)	0.053	0.005	0.012	0.001	**0.000**	0.010	**1.656**	1.586	1.586	1.581	1.586
	G	**91.7(1.8)**	90.6(1.6)	89.8(1.6)	90.2(1.7)	90.0(1.3)	90.3(1.4)	0.090	**0.001**	0.002	0.014	0.004	0.007	1.652	1.764	**1.771**	1.762	1.767
	H	**83.9(5.4)**	81.9(2.7)	80.0(2.7)	78.9(3.5)	79.7(2.3)	80.2(3.4)	0.228	0.041	0.062	0.019	0.028	**0.005**	1.631	1.743	**1.744**	1.741	1.739
	I	**88.8(3.7)**	81.7(4.8)	82.9(3.3)	85.9(3.3)	86.0(3.8)	84.9(2.8)	0.223	**0.120**	0.146	0.161	0.131	0.182	**2.468**	1.861	1.860	1.861	1.856
	J	**76.3(6.0)**	70.9(5.9)	71.0(6.5)	69.5(6.4)	71.7(5.5)	73.2(5.8)	0.501	0.342	**0.290**	0.358	0.311	0.353	**2.211**	1.833	1.837	1.841	1.846
Large	K	**96.7(0.9)**	-	96.1(0.9)	95.8(1.2)	95.8(0.7)	96.5(0.8)	0.059	-	0.030	0.031	**0.029**	0.032	-	**1.923**	1.920	**1.923**	1.922
	L	84,0(2,5)	-	*	*	88.5(6.4)	**93.4(4.6)**	0.383	-	*	*	0.171	**0.098**	-	*	*	**1.927**	1.926
	M	92.1(1.7)	-	91.7(0.8)	88.2(4.3)	91.2(0.6)	**92.4(0.5)**	0.200	-	0.139	**0.118**	0.126	0.138	-	1.924	1.922	**1.926**	1.925
	Average							0.151	-	-	-	0.066	0.068	-	-	-	1.621	1.623

Table 5: MacroF1, TPRGap, and Speedup of the models generated by Llama3.1 along with undersampling approaches. The color scale is the same of previous tables. In large datasets (K, L, M), cells with "-" represent methods with an execution time with undersampling greater than the classification time without it and were disregarded. Cells with "*" represent methods that could not be executed due to a memory allocation failure during the undersampling step.

presents effectiveness (MacroF1), bias (TPRGap), and efficiency (Speedup) results obtained when applying these 5 US methods together with the LLM – Llama3.1 – classifier considering all datasets, including the largest ones (K, L and M). Color scales follow the previous patterns.

Analyzing the US methods concerning scalability for the largest datasets (K, L, and M), we observe that CNN, despite being a highlight in the SLM analyses, was unable to scale for any of the larger sets, presenting very high undersampling times, comparable to running the LLM (LLama3.1) classification without any undersampling.

This behavior is explained by its time complexity (O(n^3), with n being the number of instances) (Cunha et al., 2021). NM1 and NM2 also did not scale for the largest dataset (L, with more than 1 million documents) due to excessive memory usage. The only methods capable of scaling for very large datasets with high imbalance were our proposals.

Regarding effectiveness, we observed that even with the change of classifier to Llama3.1, UBR and E2SC_US maintained their effectiveness compared to the classifier that does not use undersampling in all 13 datasets. In particular, E2SC_US proved to be statistically superior to the model trained without undersampling in the L dataset. CNN and NM1, despite not generalizing to all datasets, present effectiveness statistically similar to NoUnder in the datasets where they have been able to run. NM2, in turn, loses (statistically) in one dataset (D).

Regarding bias, as seen in Table 9 in Appendix B, LLMs present lower bias than SLMs. Even in this case, UBR and E2SC_US manage to reduce model bias in all datasets, especially UBR, which achieved, on average, the highest reduction, generating an average TPRGap of (0.066) – a 56% reduction in bias compared to NoUnder (0.151) Furthermore, UBR achieved the best bias reduction results in 9 out of 13 datasets when compared to E2SC_US. We also note that CNN, in dataset B, increased model bias, a behavior that did not occur in the analyses with the SLM. Our methods also achieved excellent efficiency – a speedup of 1.621 (UBR) and 1.623 (E2SC_US) on average.

5.5 arbon Emission

One consequence of reducing total model training time is the cutting of carbon emissions. Table 6 presents the carbon emissions (Lannelongue et al., 2021) generated during model training after applying the US methods. We present the results for Llama3.1 in the largest datasets.

We can observe that UBR and E2SC_US achieve a significant reduction of approximately 50% in carbon emissions compared to the model trained with-

dataset	NoUnder	UBR	E2SC_US
K	0.652	0.339	0.339
L	28.659	14.857	14.879
M	3.478	1.805	1.806
Avarage	10.930	5.667	5.675

Table 6: Carbon Emission Results (CO_2), in Kg, for generating models using Llama3.1 with undersampling.

out undersampling. This reduction is especially relevant in the context of sustainability, considering the popularity of current SLMs and LLMs and numerous research groups/companies that train, through fine-tuning or using an in-context process, such models on a daily basis.

Summarizing, both – UBR and E2SC_US – have proven to be very effective in reducing model bias without compromising effectiveness, compared to language models, either small or large, trained without the application of undersampling. In addition, our US methods can significantly reduce the total training time, with a corresponding cutoff in carbon emissions during training.

5.6 Discussion – The Role of Randomness

To better understand why UBR and E2SC_US reduce bias while some methods increase it, we further analyze Table 3. In datasets such as A and E, NCR, ENN, ALLKNN, and RENN increase dataset bias. These methods remove the majority class training examples misclassified by KNN without weighting their importance. This unweighted approach can eliminate key patterns, reduce data variability, and prevent models from learning excluded patterns. As a result, bias may increase due to reinforcement of majority class existing patterns. On the other hand, other KNN-based strategies, such as CNN, which include explicit criteria to prevent the removal of important patterns, can significantly reduce bias. This strategy, however, incurs a higher computational cost (O(n^3)) for CNN case).

Accordingly, we built our proposals by introducing randomness weighted by the semantic difficulty criteria (UBR) or by (logistic regression) confidence (E2SC-US) to prevent the removal of important patterns at a low cost. The weighted randomness process adopted in UBR is computationally cheap and effective (Andrade et al., 2024, 2023). In the case of E2SC_US, we leverage a computationally cheap, yet calibrated and effective classifier (logistic regression), justifying the improved performance compared to UBR. In short, our solutions combine the strengths of the best existing approaches while avoiding their weaknesses.

dataset	NoUnder	UBR	UBR_moreDiff	E2SC_US	E2SC_minCnf
A	0.063	0.015	0.030	**0.002**	0.092
B	0.065	**0.028**	0.041	0.034	0.063
C	0.041	0.008	0.022	**0.000**	0.067
D	0.076	**0.002**	0.026	0.019	0.092
E	0.096	0.002	0.049	**0.001**	0.206
F	0.160	**0.002**	0.089	0.006	0.138
G	0.102	0.013	0.020	**0.006**	0.138
H	0.190	0.026	**0.001**	0.040	0.217
I	0.333	**0.126**	0.200	0.222	0.336
J	0.350	**0.184**	0.421	0.215	0.262
Mean	0.148	0.041	0.090	0.054	0.161

Table 7: TPRGap generated by RoBERTa in conjunction with the original and modified undersampling approaches.

To confirm our conjectures, we conducted an extra experiment (Table 7) showing TPRGap for variations of our methods using RoBERTa. We removed the random factor and applied only the elimination of documents with higher semantic difficulty (UBR_moreDiff) or lower confidence (E2SC_minCnf). Bias increased compared to the original methods, reinforcing that non-random removal, especially in specific data groups, can hinder generalization and increase bias.

6 Conclusions and Future Work

The impact of class imbalance and model bias towards the majority class in SLM and LLM text classifiers remains underexplored. Existing US solutions have limitations, prompting us to propose two novel undersampling techniques (UBR and E2SC_US), drawing inspiration from the Instance Selection field. We evaluated 19 baselines across 13 datasets using RoBERTa (SLM) and Llama3.1 (LLM), assessing effectiveness, bias reduction, efficiency, scalability, and consistency. Results showed our methods were the only ones to significantly reduce class imbalance bias (up to 56%) without sacrificing effectiveness. They also improved efficiency (1.6x speedup or more) and cut carbon emissions by 50%. UBR was the most consistent in bias reduction, while E2SC_US excelled in efficiency. Future work will extend these methods to multi-class, hierarchical, and extreme (XM)TC, expand experiments to other LLMs, and compare US with oversampling methods.

Acknowledgements

This work was supported by CNPq, CAPES, Instituto Nacional de Ciência e Tecnologia em Inteligência Artificial Responsável para Linguística Computacional, Tratamento e Disseminação de Informação (INCT-TILD-IAR), FAPEMIG, AWS, Google, NVIDIA, CIIASaúde, and FAPESP.

Limitations

Despite relevant contributions, our study has some limitations. One of them is that our evaluation targeted specifically the sentiment classification task. First, it is worth noting that although the methods are proposed for binary classification, important problems such as spam, fake news, misinformation detection, hate speech, and sentiment analysis, among many others, can all be mapped to binary classification problems, which makes the proposed approach broadly relevant. However, although we have considered a large set of datasets, increasing the number of dataset domains and extending our analysis to include tasks such as topic classification, multi-class, and hierarchical ATC would provide new and valuable insights.

As mentioned in Section 3, we limited the removal rate parameter of our proposals to a maximum of 50%. We adopted this value considering that Cunha et al. (2023a) empirically pointed out that this is the limit of reduction where it was still possible to avoid effectiveness losses. However, this limit was set in the context of a specific instance selection solution (E2SC) and it would be interesting to further study this parameter by investigating proposals and heuristics for learning optimal dataset-specific domain reduction rates.

References

Claudio Andrade, Fabiano M Belém, Washington Cunha, Celso França, Felipe Viegas, Leonardo Rocha, and Marcos André Gonçalves. 2023. On the class separability of contextual embeddings representations–or "the classifier does not matter when the (text) representation is so good!". *IP&M*.

Claudio Andrade, Washington Cunha, Guilherme Fonseca, Ana Pagano, Luana Santos, Adriana Pagano, Leonardo Rocha, and Marcos Gonçalves. 2024. Explaining the hardest errors of contextual embedding based classifiers. In *CONLL'24*, pages 419–434.

Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. 1992. A training algorithm for optimal margin classifiers. In *5th COLT*.

Breiman. 2001. Random forests. *Machine learning*.

Glenn W Brier. 1950. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd KDD*.

Washington Cunha, Sérgio D. Canuto, Felipe Viegas, Marcos André Gonçalves, Leonardo Rocha, et al. 2020. Extended pre-processing pipeline for text classification: On the role of meta-feature representations, sparsification and selective sampling. *IP&M*.

Washington Cunha, Celso França, Guilherme Fonseca, Leonardo Rocha, and Marcos André Gonçalves. 2023a. An effective, efficient, and scalable confidence-based instance selection framework for transformer-based text classification. In *the 46th ACM SIGIR*.

Washington Cunha, Vítor Mangaravite, Christian Gomes, Sérgio Canuto, Felipe Viegas, Celso França, Martins, Jussara M Almeida, et al. 2021. On the cost-effectiveness of neural and non-neural approaches and representations for text classification: A comprehensive comparative study. *IP&M*.

Washington Cunha, Alejandro Moreo, Andrea Esuli, Fabrizio Sebastiani, Leonardo Rocha, and Marcos André Gonçalves. 2024. A noise-oriented and redundancy-aware instance selection framework. *ACM Trans. Inf. Syst.* Just Accepted.

Washington Cunha, Leonardo Rocha, and Marcos André Gonçalves. 2025. A thorough benchmark of automatic text classification: From traditional approaches to large language models. *arXiv preprint arXiv:2504.01930*.

Washington Cunha, Felipe Viegas, Celso França, Thierson Rosa, Leonardo Rocha, and Marcos André Gonçalves. 2023b. A comparative survey of instance selection methods applied to nonneural and transformer-based text classification. *ACM CSUR*.

Paula Czarnowska, Yogarshi Vyas, and Kashif Shah. 2021. Quantifying social biases in nlp: A generalization and empirical comparison of extrinsic fairness metrics. *TACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Georgios Douzas, Maria Lechleitner, and Fernando Bacao. 2022. Improving the quality of predictive models in small data gsdot: A new algorithm for generating synthetic data. *Plos one*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Xavier Ferrer, Tom van Nuenen, Jose M. Such, Mark Coté, and Natalia Criado. 2021. Bias and discrimination in ai: A cross-disciplinary perspective. *IEEE Technology and Society Magazine*.

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*. Springer.

Peter Hart. 1968. The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in NeurIPS*.

Miroslav Kubat, Stan Matwin, et al. 1997. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*. Citeseer.

Anil Kumar, Dinesh Singh, and Rama Shankar Yadav. 2024. Entropy and improved k-nearest neighbor search based under-sampling (enu) method to handle class overlap in imbalanced datasets. *Concurrency and Computation*, 36(2):e7894.

Loïc Lannelongue, Jason Grealey, and Michael Inouye. 2021. Green algorithms: quantifying the carbon footprint of computation. *Advanced science*.

Jorma Laurikkala. 2001. Improving identification of difficult small classes by balancing class distribution. In *8th Conference on AIME*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

Wei-Chao Lin, Chih-Fong Tsai, Ya-Han Hu, and Jing-Shang Jhang. 2017. Clustering-based undersampling in class-imbalanced data. *Info. Sciences*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Washington Luiz, Felipe Viegas, Rafael Alencar, Fernando Mourão, Thiago Salles, Dárlinton Carvalho, Marcos Andre Gonçalves, and Leonardo Rocha. 2018. A feature-oriented sentiment rating for mobile app reviews. WWW '18.

Inderjeet Mani and I Zhang. 2003. knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*. ICML.

Shashank Mujumdar, Stuti Mehta, Hima Patel, and Suman Mitra. 2023. Identifying semantically difficult samples to improve text classification. *arXiv preprint arXiv:2302.06155*.

Albert Orriols-Puig and Ester Bernadó-Mansilla. 2009. Evolutionary rule-based systems for imbalanced data sets. *Soft Computing*.

Andrea Pasin, Washington Cunha, Marcos André Gonçalves, and Nicola Ferro. 2024. A quantum annealing instance selection approach for efficient and effective transformer fine-tuning. In *Proceedings of the 2024 ACM SIGIR ICTIR*, pages 205–214.

Alexander Ponomarenko, Nikita Avrelin, Bilegsaikhan Naidan, and Leonid Boytsov. 2014. Comparative analysis of data structures for approximate nearest neighbor search. *Data analytics*.

Sivaramakrishnan Rajaraman, Prasanth Ganesan, and Sameer Antani. 2022. Deep learning model calibration for improving performance in class-imbalanced medical image classification tasks. *PloS one*.

Zilma Silveira Nogueira Reis, Adriana Silvina Pagano, Isaias Jose Ramos de Oliveira, Cristiane dos Santos Dias, et al. 2024. Evaluating large language model–supported instructions for medication use: First steps toward a comprehensive model. *Mayo Clinic Proceedings: Digital Health*, 2(4):632–644.

Filipe N Ribeiro, Matheus Araújo, Pollyanna Gonçalves, Marcos André Gonçalves, and Fabrício Benevenuto. 2016. Sentibench-a benchmark comparison of state-of-the-practice sentiment analysis methods. *EPJ DS*.

Michael R Smith, Tony Martinez, and Christophe Giraud-Carrier. 2014. An instance level analysis of data complexity. *Machine learning*.

Ivan Tomek. 1976a. An experiment with the edited nearest-neighbor rule.

Ivan Tomek. 1976b. Two modifications of cnn. *IEEE Transactions on Systems, Man, and Cybernetics*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, et al. 2023. Llama: Open and efficient foundation language models.

Pattaramon Vuttipittayamongkol, Eyad Elyan, Andrei Petrovski, and Chrisina Jayne. 2018. Overlap-based undersampling for improving imbalanced data classification. In *19th IDEAL*. Springer.

Heitor Werneck, Nícollas Silva, Matheus Viana, Adriano Pereira, Fernando Mourão, and Leonardo Rocha. 2021. Points of interest recommendations: Methods, evaluation, and future directions. *Inf. Syst.*

Dennis L Wilson. 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems*.

Raymond E Wright. 1995. Logistic regression.

Show-Jane Yen and Yue-Shi Lee. 2006. Undersampling approaches for improving prediction of the minority class in an imbalanced dataset. In *ICIC*.

Bruna Stella Zanotto, Ana Paula Beck da Silva Etges, Renata Ruschel, Washington Luiz, et al. 2021. Stroke outcome measurements from electronic medical records: cross-sectional study on the effectiveness of neural and nonneural classifiers. *JMIR Med. Infor.*

Qian Zhou and Bo Sun. 2024. Adaptive k-means clustering based under-sampling methods to solve the class imbalance problem. *Data Info. Management*.

dataset	RoBERTa		BART		BERT		SVM		LR		RF		XGB		LGBM		KNN	
	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap
A	88.6(0.7)	0.063	**89.3(1.1)**	0.048	84.3(1.7)	0.050	71.8(2.5)	0.187	71.4(2.5)	0.228	67.0(2.4)	0.370	64.9(2.4)	0.417	63.2(3.2)	0.375	64.3(3.1)	0.478
B	**89.0(0.7)**	0.065	88.3(1.4)	0.079	86.9(0.8)	0.057	72.1(1.5)	0.257	72.9(1.5)	0.209	68.6(1.8)	0.422	67.6(1.4)	0.247	69.4(2.7)	0.250	65.9(2.8)	0.498
C	93.3(1.1)	0.041	**93.7(1.2)**	0.036	89.1(2.2)	0.063	79.3(3.1)	0.111	80.4(2.1)	0.117	75.7(2.8)	0.221	75.0(1.5)	0.171	68.9(5.9)	0.208	76.9(2.1)	0.166
D	**89.3(1.2)**	0.076	89.1(1.1)	0.085	85.5(2.0)	0.146	76.4(2.1)	0.213	75.6(3.6)	0.223	73.3(3.0)	0.271	71.5(1.8)	0.302	72.8(3.4)	0.296	72.7(3.2)	0.330
E	**89.7(1.9)**	0.096	88.9(1.7)	0.117	86.1(1.8)	0.134	79.0(1.7)	0.215	78.6(2.0)	0.245	72.6(4.0)	0.257	71.1(1.7)	0.472	71.7(3.7)	0.385	73.3(3.4)	0.263
F	**94.2(1.0)**	0.160	93.7(1.0)	0.053	88.0(1.3)	0.108	82.0(1.0)	0.238	80.4(1.4)	0.320	72.6(1.9)	0.526	74.0(1.4)	0.476	74.0(2.8)	0.372	75.4(1.3)	0.454
G	**90.1(1.5)**	0.102	90.1(1.6)	0.099	86.4(1.9)	0.125	70.9(2.0)	0.404	71.9(1.7)	0.399	64.5(2.0)	0.615	64.0(1.9)	0.660	63.9(4.4)	0.633	60.8(1.5)	0.654
H	**83.8(5.0)**	0.190	81.6(5.6)	0.198	79.1(3.6)	0.290	67.0(5.6)	0.539	63.0(6.8)	0.619	54.9(5.1)	0.731	56.7(5.2)	0.666	56.4(5.3)	0.598	55.2(6.4)	0.800
I	**83.2(3.4)**	0.333	83.0(5.8)	0.283	79.8(4.9)	0.350	68.1(3.8)	0.610	63.1(6.4)	0.733	59.8(4.6)	0.809	59.2(3.5)	0.803	58.4(10.5)	0.763	56.5(4.4)	0.875
J	**81.0(4.5)**	0.350	78.0(6.1)	0.410	76.4(4.4)	0.447	50.5(5.0)	0.944	53.3(4.7)	0.914	54.3(5.7)	0.888	53.4(5.3)	0.898	55.7(11.2)	0.833	46.4(0.1)	0.998
K	**93.1(0.6)**	0.156	92.6(1.4)	0.149	90.6(1.3)	0.203	82.6(0.3)	0.412	83.1(1.1)	0.408	80.1(1.0)	0.522	76.1(1.5)	0.594	79.4(0.8)	0.525	65.4(2.6)	0.788
L	**92.0(0.1)**	0.166	91.1(0.1)	0.192	89.8(0.2)	0.225	81.6(0.4)	0.416	82.0(0.3)	0.430	80.6(0.7)	0.519	73.4(0.1)	0.644	74.4(0.1)	0.622	62.3(0.4)	0.791
M	87.8(2.5)	0.308	**88.6(1.2)**	0.296	85.3(0.7)	0.383	78.7(0.3)	0.601	78.2(0.7)	0.562	78.9(0.4)	0.561	72.0(0.3)	0.698	73.6(0.7)	0.668	60.9(0.5)	0.866
Avarage		0.162		0.157		0.198		0.396		0.416		0.516		0.542		0.502		0.612

Table 8: MacroF1 and TPRGap results of the classifiers. Cells in **bold** indicate the highest numerical values for a dataset, and cells in green are statistically equivalent to the highest numerical classification value.

A Comparison among Classifiers - Effectiveness and Bias

Seeking to analyze how recent state-of-the-art Transformer-based algorithms are affected by class imbalance in sentiment analysis tasks and understand if there is room for improvement, we compared them with traditional classifiers using the analyzed skewed datasets. We consider ATC methods based on SLMs **RoBERTa** (Liu et al., 2019), **BART** (Lewis et al., 2020) and **BERT** (Devlin et al., 2018), which are among the best sentiment classifiers (Cunha et al., 2023b).

We use the same methodology discussed in (Cunha et al., 2023b) to adjust the hyperparameters. We set the initial learning rate as 5×10^{-5}, the maximum number of epochs as 20, and the patience as 5 epochs. We performed a grid search on max_len (150 and 256) and batch_size (16, 32 and 64), as these specified values directly impact the efficiency and effectiveness of the model.

We also consider 6 traditional classifiers: **KNN**, Random Forest (Breiman, 2001), Logistic Regression (Wright, 1995), Support Vector Machine (Boser et al., 1992), XGBoost (Chen and Guestrin, 2016) and LightGBM (Ke et al., 2017).

Table 8 presents the MacroF1 and TPRGap results for these classifiers. First, we can highlight the superiority, in terms of *effectiveness*, of classifiers based on Transformers when compared to traditional ones – these were inferior (statistically) to Transformers in all 13 datasets considered, a result consistent with the literature (Cunha et al., 2021).

RoBERTa and BART stand out among the Transformer-based classifiers, with statistically equivalent classification results on all datasets, but one: L, in which RoBERTa was slightly superior. BERT, in turn, is statistically equivalent to RoBERTa and BART in only 5 of the 13 datasets. Finally, when we analyze the absolute numerical values of each classifier, the RoBERTa model produces the highest MacroF1 values in 10 datasets while BART does so in 3 of them, reinforcing the results from the literature reporting (Cunha et al., 2023b) RoBERTa as being a state-of-the-art sentiment classifier. It is, therefore, the classifier of choice used in all analyses in our article.

Analyzing Table 8 again, now focusing on the class imbalance bias (average TPRGap) of the approaches (remind that the smaller the TPRGap, the smaller the bias), we observe that classifiers based on Transformers present lower bias when compared to traditional ATC methods. For traditional classifiers, the method that obtained the best values was SVM with an average TPRGap of 0.396, which is more than double the average TRPGap of methods based on Transformers that achieved 0.162 (RoBERTa), 0.157 (BART) and 0.198 (BERT). This result is very interesting, and as far as we know, it has not been reported in the literature - *the good ability of SLMs to deal with imbalanced data*.

Despite this, SLMs still present high TPRGap results in datasets with a high imbalance, as is the case with datasets K, L, and M, with IR equal to 10.73, 13.84, and 39.71. This evidence suggests that there is still room for improvement, that is, room to explore and devise novel techniques capable of reducing the class imbalance bias of SLMs.

dataset	LLama 3.1		RoBERTa	
	MacroF1	TPRGap	MacroF1	TPRGap
A	**91.9(1.0)**	0.038	88.6(0.7)	0.063
B	**93.2(0.8)**	0.028	89.0(0.7)	0.065
C	**97.6(1.1)**	0.017	93.3(1.1)	0.041
D	**91.2(1.0)**	0.076	89.3(1.2)	0.076
E	**93.9(1.4)**	0.065	89.7(1.9)	0.096
F	**95.2(0.7)**	0.053	94.2(1.0)	0.160
G	**91.7(1.8)**	0.090	90.1(1.5)	0.102
H	**83.9(5.4)**	0.228	83.8(5.0)	0.190
I	**88.8(3.7)**	0.223	83.2(3.4)	0.333
J	76.3(6.0)	0.501	**81.0(4.5)**	0.350
K	**96.7(0.9)**	0.059	93.1(0.6)	0.156
L	84.0(2.5)	0.383	**92.0(0.1)**	0.166
M	**92.1(1.7)**	0.200	87.8(2.5)	0.308
Average		0.151		0.162

Table 9: MacroF1 and TPRGap results of the classifiers (RoBERTa and Llama3.1). Cells in **bold** indicate the highest numerical values for a dataset, and cells in green are statistically equivalent to the highest value.

dataset	MacroF1						TPRGap						Speedup				
	NoUnder	AKCS	ENUB	ENUT	ENUC	ENUR	NoUnder	AKCS	ENUB	ENUT	ENUC	ENUR	AKCS	ENUB	ENUT	ENUC	ENUR
A	88.6(0.7)	89.2(1.3)	89.2(0.9)	88.9(1.2)	88.9(1.1)	88.9(1.1)	0.063	0.008	0.035	0.033	0.032	0.032	0.885	1.188	1.119	1.185	1.186
B	89.0(0.7)	89.0(1.2)	84.1(11.9)	84.1(11.9)	84.1(11.9)	84.1(11.9)	0.065	0.010	0.145	0.145	0.145	0.145	0.665	1.107	1.107	1.106	1.106
C	93.3(1.1)	94.2(1.5)	94.3(1.6)	94.0(1.7)	94.2(1.7)	94.3(1.6)	0.041	0.008	0.028	0.035	0.037	0.028	1.070	1.361	1.441	1.439	1.354
D	89.3(1.2)	83.5(11.3)	88.8(1.7)	88.1(1.9)	88.1(1.9)	88.8(1.7)	0.076	0.085	0.036	0.015	0.015	0.036	1.301	1.139	1.255	1.254	1.139
E	89.7(1.9)	88.5(1.9)	89.4(1.4)	89.9(1.3)	90.1(1.7)	89.4(1.4)	0.096	0.028	0.104	0.083	0.081	0.104	1.202	1.026	1.070	1.075	0.971
F	94.2(1.0)	92.4(1.0)	93.7(0.9)	93.9(0.8)	94.0(0.9)	93.7(0.9)	0.160	0.032	0.060	0.055	0.048	0.060	1.048	1.310	1.164	1.168	1.309
G	90.1(1.5)	87.6(1.8)	89.9(1.0)	90.0(0.8)	90.0(0.8)	89.9(1.0)	0.102	0.029	0.064	0.067	0.067	0.064	0.744	1.275	1.145	1.145	1.271
H	83.8(5.0)	80.3(3.9)	83.7(3.5)	83.7(4.5)	83.7(3.5)	83.7(3.5)	0.190	0.114	0.146	0.123	0.146	0.146	1.288	1.297	1.179	1.296	1.294
I	83.2(3.4)	75.2(3.7)	84.4(3.9)	83.4(5.4)	83.3(3.5)	84.4(3.9)	0.333	0.096	0.303	0.276	0.296	0.303	1.537	1.227	1.041	1.346	1.226
J	81.0(4.5)	61.0(5.1)	77.6(2.9)	77.6(2.9)	77.6(2.9)	77.6(2.9)	0.350	0.162	0.350	0.350	0.350	0.350	2.477	1.790	1.787	1.783	1.783
Average							0.148	0.057	0.127	0.118	0.122	0.127	1.222	1.272	1.231	1.280	1.264

Table 10: MacroF1, TPRGap, and Speedup of the models generated by RoBERTa along with the undersampling approaches. For the MacroF1 results, green cells represent executions that are statistically equivalent to the NoUnder results. For TPRGap, the greener the cell, the greater the bias reduction, while the redder it is, the greater the bias increase compared to NoUnder. For Speedup, the greener the cell, the greater the reduction in total training time compared to the approach without undersampling, whereas the redder it is, the longer the time.

B Comparison between SLMs and LLMs classifiers on imbalanced datasets

Table 9 presents a comparative analysis between the ATC models based on Transformers of 1st (SLMs) and 2nd (LLMs) generations, represented respectively by RoBERTa and Llama3.1. Bold values in MacroF1 represent the best results and green cells represent the statistical ties between methods. Regarding effectiveness, we observe the superiority of Llama3.1, which achieves the best result in 11 of the 13 datasets and is statistically inferior to RoBERTa in only one dataset. We also observe that, in terms of bias, LLM has a lower average value, 0.151 for Llama3.1, against 0.162 for RoBERTa. Even so, significant potential for class imbalance bias reduction is observed, particularly in larger datasets with an Imbalance Ratio (IR) greater than 5. Under these conditions, even advanced LLM models exhibit high TPRGap values, which can reach up to 0.5 (as verified in dataset I).

C Experiments with Recent US methods

Table 10 presents MacroF1, TPRGap, and Speedup results obtained by the most recent undersampling methods utilized with RoBERTa. Regarding effectiveness, as measured by MacroF1, the methods of the ENU family maintain a statistically equivalent performance to NoUnder in all 10 datasets. On its turn, AKCS presents a statistically significant drop in 2 datasets.

For TPRGap, all methods increase bias in relation to NoUnder, with this worsening observed for AKCS (1 dataset), ENUT and ENUC (2 datasets) and ENUB and ENUR (4 datasets).

Finally, in terms of Speedup, ENUB, ENUT, and ENUC considerably reduce training time, although still lower than the best speedups presented in Section 5.3. The other methods do not significantly worsen training time, with impacts observed in only 1 (ENUR) and 3 (AKCS) datasets.

Since none of the methods managed to simultaneously meet the first three criteria analyzed — maintain effectiveness, reduce class imbalance bias, and improve efficiency – we chose not to include their results in the main body of the paper.

D Imbalance Ratio after Undersampling

Table 11 shows the IR of the datasets after applying the undersampling methods proposed in this paper. Our methods can perfectly balance classes in 8 out of 13 datasets. In Datasets I to M, our methods do not balance perfectly because, as mentioned in Section 3, we limited the reduction to 50% of the dataset of only majority class data. These results also reveal opportunities for improvement.

dataset	NoUnder	UBR	E2SC_US
A	1.41	1	1
B	1.44	1	1
C	1.51	1	1
D	1.71	1	1
E	2.17	1	1
F	2.23	1	1
G	2.66	1	1
H	2.72	1	1
I	5.32	2.16	2.16
J	6.60	2.80	2.80
K	10.73	4.87	4.87
L	13.84	6.42	6.42
M	39.71	19.35	19.35

Table 11: Datasets' IR before and after undersampling.

E Hardware Utilized in the Experiments

The experiments related to small datasets (A - J) that used RoBERTa as a classifier were carried out on AWS. The undersampling steps, which require exclusively CPU processing, used an instance of type *c6a.4xlarge*, while for the classification steps, which require specialized hardware (GPU), we used instances of type *g4dn.xlarge*. The experiments that used Llama3.1 as a classifier had the undersampling steps performed on instances of type *g5.4xlarge* and the classification was executed on an *AMD Ryzen 5 5600X 6-Core and 12-Threads, 64 Gb RAM and an NVIDIA GeForce RTX 3090.*

dataset	NoUnder	CNN	NM1	NM2	CC_NN	SBC	NM3	OBU	NCR	IHT	TL	OSS	ENN	RENN	ALLKNN	UBR	E2SC_US
MacroF1																	
K	93.1(0.6)	-	91.2(1.5)	92.2(1.1)	-	56.7(9.9)	41.3(3.5)	86.3(3.8)	92.9(1.5)	83.6(1.4)	**93.5(1.0)**	93.2(0.7)	91.6(1.3)	69.4(1.4)	68.3(3.4)	92.0(0.9)	92.9(0.7)
L	**92.0(0.1)**	-	*	*	-	44.9(2.3)	*	*	-	76.7(1.0)	-	-	-	-	-	91.4(0.7)	91.7(0.2)
M	87.8(2.5)	-	87.1(1.3)	85.1(3.5)	-	30.9(1.7)	51.0(1.7)	77.3(9.1)	80.1(21.4)	60.3(1.5)	80.0(21.4)	72.7(26.5)	79.8(21.2)	69.2(3.5)	67.9(2.6)	86.9(0.6)	**88.7(0.3)**
TPRGap																	
K	0.1559	-	0.075	0.086	-	0.351	0.580	0.092	0.128	0.018	0.137	0.140	0.081	0.165	0.178	0.070	0.091
L	0.166	-	*	*	-	0.485	*	*	-	0.033	-	-	-	-	-	0.104	0.124
M	0.308	-	0.207	0.208	-	0.619	0.198	0.245	0.434	0.067	0.443	0.579	0.403	0.045	0.042	0.185	0.214
Avarage	0.210	-	*	*	-	0.485	*	*	-	0.039	-	-	-	-	-	0.120	0.143
Speedup																	
K	-	-	1.687	1.963	-	2.468	6.443	1.325	1.073	1.723	1.038	0.985	1.537	1.855	2.042	1.880	1.884
L	-	-	*	*	-	2.006	*	*	-	1.878	-	-	-	-	-	2.026	1.924
M	-	-	2.709	2.867	-	12.498	39.486	1.777	1.038	2.103	0.892	1.318	1.230	0.946	1.713	2.067	2.903
Avarage	-	-	*	*	-	5.657	*	*	-	1.901	-	-	-	-	-	1.991	2.237

Table 12: MacroF1, TPRGap, and Speedup of the models generated by RoBERTa along with undersampling approaches. The color scale is the same of previous tables. Cells with “-” represent methods with an execution time with undersampling greater than the classification time without it and were disregarded. Cells with “*” represent methods that could not be executed due to a memory allocation failure during the undersampling step.

F Results of SLMs in Large Datasets

Table 12 presents effectiveness (MacroF1), bias (TPRGap), and efficiency (Speedup) results of the undersampling methods applied in conjunction with RoBERTa on the large datasets. In MacroF1, green cells indicate statistical ties with NoUnder, and bold cells highlight the highest values for a dataset. For TPRGap, darker green cells indicate greater bias reduction than NoUnder, while red cells represent increased bias. For Speedup color scale, greener cells reflect higher speedup, whereas red cells indicate execution times longer than NoUnder. It is observed that, in addition to the proposed methods, only IHT and SBC were able to scale for all large datasets. However, these methods cannot maintain statistically equivalent effectiveness in any dataset.

Regarding effectiveness, efficiency, and bias, the results remain consistent with the analyses presented in Section 5.4, where UBR and E2SC_US are capable of reducing model bias, decreasing the total training time while maintaining statistical equivalence to the model trained without US.

G Best Calibrated Weak Classifier

We conducted a comparative analysis of weak classifiers to identify the best option for use in our undersampling method. Our goal was to find a classifier that is not only effective and efficient but also well-calibrated, meaning that its predicted class probabilities consistently correspond to the observed accuracy of the classifier (Rajaraman et al., 2022). In the experiments, we evaluated the following classifiers: K-Nearest Neighbors (KNN, used in the original E2SC method), Random Forest (RF), Naive Bayes (NB), Nearest Centroid (NC), Decision Trees (DT), Logistic Regression (LR), XGBoost (XGB), LightGBM (LGBM), and Linear SVM (LSVM). We used four datasets described in Table 13 for the analysis.

Dataset	Size
Books	33,594
ACM	24,897
20NG	18,846
Twitter	6,997

Table 13: Datasets.

We assessed the calibration of the weak classifiers using the Brier Score (BS) (Brier, 1950), a scoring rule used to measure the accuracy of probabilistic predictions. Brier (Brier, 1950) defines it as:

$$BS = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\left(P(Y_i = j \mid x_i) - y_{ij}\right)^2,$$

where y_{ij} is the one-hot vector with a value of 1 at the true class index of x_i, and 0 otherwise. The BS ranges from 0 (best) to 2 (worst) – the closer to zero, the better the calibration of the probability estimate.

Weak classifier	Average BS
LR	0.37
KNN	0.44
XGB	0.45
LGBM	0.48
NB	0.49
RF	0.5
LSVM	0.73
NC	0.75
DT	0.77

Table 14: Brier Score Average for each weak classifier.

Table 14 presents the average Brier Score (BS) values for each classification method. It is observed that the Logistic Regression model is the most calibrated, even outperforming K-Nearest Neighbors (KNN). Table 15 presents the MacroF1 values and execution time. We excluded the LSVM, NC, and DT methods for this analysis, as they had already shown significantly high Brier Score (BS) values. In the table, we see that LR also stands out in both effectiveness and efficiency, being 13.8x and 97.8x faster than its competitors while also achieving the

Dataset	LR		KNN		XGB		LGBM		NB		RF	
	MacroF1	**Time (s)**	**MacroF1**	**Time (s)**	**MacroF1**	**Time (s)**	**MacroF1**	**Time (s)**	**MacroF1**	**Time (s)**	**MacroF1**	**Time (s)**
Books	81.2 (0.5)	3.82	76.5 (0.5)	14.42	75.5 (0.5)	41.18	78.8 (0.4)	55.46	73.3 (0.6)	1.86	75.70 (0.6)	154.12
20NG	86.1 (0.7)	5.10	82.8 (0.4)	5.27	77.8 (0.7)	74.83	81.8 (0.7)	94.36	77.4 (0.5)	3.44	81.64 (0.6)	127.07
ACM	59.6 (0.6)	1.69	58.6 (2.0)	3.43	58.6 (0.8)	23.35	62.5 (1.5)	23.96	40.7 (0.8)	3.38	60.08 (0.8)	165.34
Twitter	63.1 (1.1)	0.12	52.9 (2.2)	0.71	52.9 (1.2)	4.05	53.2 (2.0)	3.93	31.4 (0.7)	0.40	43.59 (2.1)	7.97

Table 15: Effectiveness and efficiency of weak classifiers.

highest MacroF1 values. Based on these results, we consider LR the best weak classifier, which is why it was selected to be part of our E2SC_US method.

H Hyperparameters of ATC models

We use the same methodology discussed in (Cunha et al., 2023b) to adjust the hyperparameters. For all models, we set the initial learning rate as 5×10^{-5}, the maximum number of epochs as 20, and the patience as 5 epochs. We performed a grid search on max_len (150 and 256 for SLMs, 150 for LLMs) and batch_size (16, 32 and 64 for SLMs, 2 and 4 for LLMs), as these specified values directly impact the efficiency and effectiveness of the model. For the LLMs, we adopted the methodology provided by (Andrade et al., 2024). Thus, we fine-tuned the LLMs by applying QLoRA, fixing the initial learning rate as 2e-4, the number of epochs and the batch_size as 4, and the max_len as 256.

I Time Complexity - All Methods

Method	Time Complexity
CNN	$O(n^3)$
NM1	$O(n^2)$
NM2	$O(n^2)$
CC_NN	$O(n^2)$
SBC	$O(n^2)$
NM3	$O(n^2)$
OBU	$O(n^2)$
NCR	$O(n^2)$
IHT	$O(nlog(n))$
TL	$O(n^2)$
OSS	$O(n^2)$
ENN	$O(n^2)$
RENN	$O(n^2)$
ALLKNN	$O(n^2)$
AKCS	$O(n^3)$
ENUB	$O(n^2)$
ENUT	$O(n^2)$
ENUC	$O(n^2)$
ENUR	$O(n^2)$
UBR	$O(nlog(n))$
E2SC_US	$O(n)$

Table 16: Time complexity of undersampling methods.

Table 16 presents the time complexity of all baselines. For the time complexity analysis, our two proposals can be divided into two steps: (1) calculation of the probability distribution and (2) sampling weighted by the obtained distribution.

Let n be the total number of instances in the dataset. In the E2SC_US method, the time complexity of the first step is equivalent to that of the logistic regression algorithm used to estimate the confidence of each document, that is, $O(n)$. In the UBR method, the first step involves a search for the nearest neighbors that is done using an approximated version of KNN (Ponomarenko et al., 2014), whose complexity is $O(n \log n)$.

The second step, which consists of weighted random sampling, has constant complexity $O(1)$ for both methods. Thus, the final complexity is $O(n)$ for E2SC_US and $O(n \log n)$ for UBR.

Macro-F1					
Method	50%	60%	70%	80%	Balanced
e2sc_US	92.9(0.7)	92.3(1.7)	91.1(1.1)	89.1(0.9)	87.1(0.9)
UBR	92.0(0.9)	90.5(0.5)	86.6(1.1)	56.0(6.6)	35.2(8.0)
TPRGap					
Method	50%	60%	70%	80%	Balanced
e2sc_US	0.091	0.077	0.060	0.022	0.010
UBR	0.07	0.050	0.004	0.350	0.671

Table 17: MacroF1 and TPRGap of the models generated by RoBERTa along with the undersampling approaches using different maximum removal percentages. Cells in green are statistically equivalent to the highest numerical classification value. Macro-F1 and TPRGap for NoUnder are respectively 93.1(0.6) and 0.156.

J Change in the maximum reduction rate

In these experiments, we gradually increased the maximum removal limit by 10% step until the dataset became balanced. We tested this variation using RoBERTa regarding the two proposed methods, UBR and E2SC_US. Table 17 presents the MacroF1 and TPRGap results of the models, where the “NoUnder” represents the model trained without applying undersampling. The percentage columns indicate the maximum percentage of the dataset that the undersampling techniques could reduce. The maximum limit for the experiments presented in the paper was 50%, a theoretical threshold where performance losses can be avoided, as found in (Cunha et al., 2023a). Finally, the "Balanced" column indicates that the US methods reduced the majority class until it reached the same size as the minority class. Cells in green indicate a statistical tie with the NoUnder model. The experiments were conducted using the luxury beauty data (K), which was chosen due to its high imbalance (10.73).

Results indicate that, in terms of Macro-F1, the average value tends to decrease as the removal percentage increases. This trend continues until

dataset	CNN	NM1	NM2	CC_NN	SBC	NM3	OBU	NCR	IHT	TL	OSS	ENN	RENN	ALLKNN	UBR	E2SC_US
A	24.906	0.027	0.085	11.621	11.443	0.045	0.216	0.131	0.093	0.073	0.109	0.064	0.072	0.074	1.308	0.026
B	110.086	0.064	0.215	27.289	11.365	0.065	0.226	0.313	0.085	0.198	0.276	0.166	0.195	0.167	1.294	0.039
C	4.854	0.009	0.024	3.843	2.241	0.012	0.059	0.040	0.042	0.019	0.036	0.018	0.022	0.022	0.212	0.016
D	23.157	0.020	0.063	6.791	3.134	0.029	0.139	0.106	0.091	0.060	0.112	0.055	0.062	0.068	0.381	0.022
E	40.329	0.025	0.076	13.037	3.915	0.031	0.212	0.131	0.125	0.073	0.113	0.072	0.082	0.117	2.219	0.028
F	90.988	0.062	0.230	29.804	17.660	0.080	0.399	0.361	0.187	0.208	0.295	0.220	0.813	0.472	3.859	0.053
G	58.094	0.042	0.135	15.389	21.646	0.061	0.327	0.264	0.160	0.166	0.229	0.166	0.437	0.313	3.151	0.040
H	2.897	0.005	0.010	1.890	1.713	0.006	0.061	0.028	0.058	0.013	0.027	0.013	0.020	0.023	0.147	0.016
I	2.183	0.004	0.008	2.474	1.250	0.005	0.053	0.029	0.048	0.012	0.026	0.014	0.054	0.042	0.097	0.015
J	2.791	0.005	0.007	2.979	2.401	0.006	0.100	0.032	0.056	0.016	0.031	0.018	0.050	0.042	0.192	0.018

Table 18: Undersampling time for the datasets.

datasets	NoUnder	CNN	NM1	NM2	CC_NN	SBC	NM3	OBU	NCR	IHT	TL	OSS	ENN	RENN	ALLKNN	UBR	E2SC_US
A	366.433	269.810	340.329	322.627	295.390	289.799	334.188	266.310	302.438	309.881	394.366	380.871	258.761	266.044	297.699	318.464	334.785
B	551.192	437.966	473.696	494.413	437.015	392.497	346.657	410.095	557.339	447.777	572.473	556.084	372.960	363.797	379.263	461.863	432.504
C	226.080	161.181	182.717	206.129	184.102	135.699	181.182	186.013	184.686	202.721	258.822	255.384	211.222	217.488	201.768	215.498	161.409
D	616.196	382.343	419.170	424.782	403.984	373.120	412.610	480.193	406.266	439.205	519.633	535.497	313.155	332.073	425.742	395.421	359.800
E	365.993	270.777	278.721	278.792	236.228	202.760	261.127	264.700	227.464	309.298	390.247	385.644	219.812	226.932	222.781	264.189	233.058
F	1,133.576	773.686	739.941	824.100	714.546	766.416	801.155	851.946	1,009.367	872.291	1,197.111	1,138.421	891.936	927.876	977.119	886.730	745.976
G	516.820	287.591	311.189	318.789	261.194	277.108	293.716	320.897	391.141	309.413	489.285	483.619	370.127	291.108	324.982	285.654	264.690
H	148.707	95.459	77.227	92.766	90.233	93.265	97.760	111.547	105.974	90.843	143.731	146.537	96.644	97.960	84.813	85.896	90.850
I	145.117	61.304	83.792	95.200	87.725	49.393	68.100	111.299	165.422	94.398	144.575	181.267	146.428	145.308	148.446	89.381	89.681
J	156.327	62.749	92.242	96.195	105.224	42.911	50.961	110.453	104.887	97.302	140.992	160.648	116.850	127.889	116.296	98.865	86.526

Table 19: Classification time of the RoBERTa model trained alongside the undersampling methods.

datasets	NoUnder	CNN	NM1	NM2	CC_NN	SBC	NM3	OBU	NCR	IHT	TL	OSS	ENN	RENN	ALLKNN	UBR	E2SC_US
A	2060	1407	1708	1708	1708	1576	1708	1520	1445	1708	1954	1949	1303	1303	1416	1708	1708
B	3249	2427	2667	2667	2667	2459	1554	2380	3155	2667	3230	3230	1947	1888	1718	2667	2667
C	1104	708	878	878	878	606	878	818	821	878	1057	1049	788	788	843	878	878
D	1781	1292	1314	1314	1314	1290	1314	1489	1226	1314	1711	1707	1039	1039	1118	1314	1314
E	2188	1531	1380	1380	1380	941	1314	1571	1071	1380	2179	2178	828	828	857	1380	1380
F	3776	2040	2338	2338	2338	2336	2338	2600	3025	2338	3708	3688	2922	2653	2798	2338	2338
G	2754	1499	1506	1506	1506	1505	1506	1813	2139	1506	2619	2614	1883	1512	1731	1506	1506
H	703	395	378	378	378	376	378	471	474	378	672	670	421	334	349	378	378
I	750	266	375	375	375	195	237	539	706	375	749	738	658	606	630	375	375
J	676	268	338	338	338	177	178	391	513	338	653	643	527	506	516	338	338

Table 20: Number of remaining instances in the training set after applying the undersampling methods.

the point at which the US models are no longer statistically equivalent to the NoUnder model. This threshold occurs at 50% for UBR, which supports the ideas presented in (Cunha et al., 2023a), suggesting that 50% is the limit where the dataset can be reduced without a loss in effectiveness. However, we obtained a threshold of 70% for e2sc_US, which is not contrary to the ideas in (Cunha et al., 2023a) since the authors discuss how this rate does not necessarily represent the best reduction rate for all datasets, as for some of the datasets they analyzed, the threshold values were higher than 50%, making it merely an average estimate.

Regarding class imbalance bias, in E2SC_US, TPRGap values decrease as the removal percentage increases. For UBR, the values also show a reduction, except at the 80% threshold and when the data is balanced, where bias increases. The extreme reduction of 80% of the instances may eliminate essential patterns, decreasing data variability and preventing the models from learning certain patterns. Consequently, the models struggle to handle these patterns when encountered in the test set. This limitation is evidenced by the fact that, in addition to exhibiting high bias, these models also showed low effectiveness, indicating that they failed to adapt well to the data patterns.

K Efficiency Gains Analysis

Tables 18, 19, and 20 present, respectively: the time spent performing undersampling, RoBERTa's total classification time (including fine-tuning and prediction time), and number of instances remaining in the training set after applying undersampling. As previously presented, for the methods in which it was possible to define a specific reduction amount, we limited the reduction to up to 50% of the total number of instances in the training set — an empirical limit (given the state-of-the-art) for possible reduction without loss of efficiency (Cunha et al., 2023a) — or until it equals the minority class.

In simpler terms, we observe that performance gains primarily result from the reduction in the training set size, leading to a decrease in the fine-tuning time of LLMs/SLMs models. Methods that remove fewer training documents, such as TL and OSS, exhibit lower performance gains than those that remove more, such as SBC and NM3. For example, in dataset J, the TL method removed only 23 instances, resulting in a classification time nearly identical to that of the model without undersampling. On the other hand, in the same dataset, the NM3 method removed more than 70% of the training data, which, while significantly reducing classification time, negatively impacted its efficiency.

Majority Class													
dataset	metrics	RoBERTa						LLama					
		NoUnder	CNN	NM1	NM2	UBR	E2SC_US	NoUnder	CNN	NM1	NM2	UBR	E2SC_US
A	F1	90.7(0.6)	89.7(1.5)	90.6(0.7)	90.7(0.7)	90.7(1.2)	90.5(0.9)	93.3(0.9)	91.8(1.5)	92.5(1.2)	92.9(1.4)	93.1(1.0)	93.3(1.2)
	Precision	89.9(1.5)	93.3(1.7)	92.0(1.6)	92.4(1.4)	91.6(1.8)	92.1(1.7)	93.0(1.4)	94.6(1.9)	94.0(1.7)	94.9(1.5)	94.6(2.0)	95.0(1.7)
	Recall	91.6(1.8)	86.4(3.0)	89.3(1.7)	89.1(1.6)	89.9(1.8)	89.0(2.0)	93.7(1.9)	89.3(2.6)	91.0(2.0)	91.0(2.2)	91.8(2.3)	91.7(1.7)
B	F1	91.1(0.6)	89.8(2.1)	90.3(1.0)	90.1(0.5)	90.4(0.9)	91.0(0.9)	94.4(0.6)	93.7(1.4)	92.2(4.8)	93.9(0.6)	94.5(0.6)	94.0(0.5)
	Precision	90.2(1.0)	92.7(1.4)	91.0(1.5)	91.3(1.2)	90.9(0.8)	91.2(0.9)	94.3(1.3)	96.1(1.2)	93.6(3.3)	96.1(1.0)	95.9(0.8)	94.8(1.1)
	Recall	92.1(1.1)	87.3(3.8)	89.7(1.6)	89.1(1.0)	89.9(1.7)	90.8(1.5)	94.6(1.1)	91.6(2.5)	90.9(6.1)	91.9(1.5)	93.2(0.9)	93.2(0.8)
C	F1	94.7(0.9)	94.9(0.9)	95.4(1.2)	95.2(0.9)	95.4(0.8)	94.7(1.0)	98.1(0.8)	98.1(0.6)	97.9(0.9)	98.6(0.6)	98.3(0.9)	98.4(0.7)
	Precision	94.3(1.4)	96.0(1.1)	96.2(1.4)	95.5(1.2)	96.1(1.5)	95.8(1.6)	97.9(1.5)	98.8(0.7)	98.5(1.1)	99.2(0.7)	98.6(0.8)	98.5(0.9)
	Recall	95.3(1.5)	93.9(1.4)	94.6(1.5)	95.0(1.8)	94.9(1.5)	93.6(1.6)	98.4(0.9)	97.4(1.3)	97.3(1.4)	98.0(1.3)	98.0(1.2)	98.4(0.9)
D	F1	92.2(0.8)	91.0(0.8)	91.0(1.2)	89.3(1.2)	91.4(1.0)	91.4(1.4)	93.7(0.7)	93.3(0.7)	93.0(1.0)	90.3(1.5)	93.2(1.3)	93.1(1.4)
	Precision	91.3(1.2)	93.0(1.4)	91.8(1.6)	92.5(1.2)	93.5(1.1)	92.9(1.2)	92.7(1.4)	94.0(1.3)	94.0(1.1)	94.3(0.9)	94.1(1.0)	93.8(1.4)
	Recall	93.3(1.2)	89.1(1.7)	90.3(2.0)	86.2(1.7)	89.5(1.5)	90.1(2.4)	94.7(1.2)	92.6(1.2)	91.9(1.4)	86.7(2.5)	92.3(2.3)	92.4(2.0)
E	F1	93.6(1.2)	93.2(1.1)	92.2(1.4)	92.9(1.3)	92.8(1.1)	92.9(1.2)	96.2(0.9)	95.4(1.0)	94.2(1.3)	94.4(1.3)	94.7(1.2)	95.0(1.1)
	Precision	93.0(1.5)	94.6(1.1)	95.4(1.3)	95.1(1.5)	95.5(1.2)	95.4(1.3)	95.6(1.2)	96.8(0.8)	97.3(0.9)	96.3(1.3)	97.2(0.8)	96.7(1.2)
	Recall	94.2(1.9)	91.8(2.2)	89.3(2.4)	90.8(2.1)	90.4(1.7)	90.6(1.9)	96.8(1.3)	94.1(1.9)	91.3(2.5)	92.6(1.9)	92.3(2.0)	93.3(1.7)
F	F1	96.5(0.6)	95.4(0.7)	95.2(0.8)	95.1(0.8)	95.6(0.9)	95.5(0.8)	97.1(0.5)	96.5(0.8)	96.1(0.6)	96.2(0.5)	96.4(0.7)	96.8(0.6)
	Precision	95.8(0.9)	97.5(0.6)	96.9(0.5)	96.9(0.7)	97.2(0.8)	97.4(0.8)	96.6(0.8)	97.9(0.6)	97.9(0.5)	97.7(0.6)	97.7(0.5)	97.7(0.7)
	Recall	97.1(0.6)	93.5(1.2)	93.6(1.3)	93.4(1.1)	94.1(1.4)	93.8(1.2)	97.6(0.8)	95.0(1.4)	94.4(1.1)	94.8(0.8)	95.0(1.4)	95.9(0.9)
G	F1	94.6(0.8)	94.1(1.1)	93.9(1.3)	93.5(1.2)	93.5(0.7)	93.1(1.0)	95.6(0.9)	94.5(0.9)	94.0(1.0)	94.2(1.1)	94.1(0.8)	94.4(0.9)
	Precision	94.3(1.0)	96.2(0.9)	96.6(1.1)	96.9(0.5)	96.7(0.7)	96.3(1.0)	95.2(1.5)	97.0(1.1)	96.6(0.9)	97.1(0.7)	96.8(0.8)	96.7(1.1)
	Recall	95.0(1.0)	92.1(1.6)	91.5(2.0)	90.3(2.0)	90.5(1.4)	90.2(1.4)	96.0(0.7)	92.2(1.1)	91.6(1.7)	91.5(2.0)	91.6(1.6)	92.2(1.7)
H	F1	91.7(2.0)	89.7(3.2)	87.7(3.1)	88.3(2.6)	89.0(2.3)	87.7(3.4)	92.1(2.3)	89.3(1.8)	87.2(1.8)	86.9(1.7)	87.8(1.5)	87.9(1.9)
	Precision	90.8(3.1)	94.2(2.8)	94.1(3.3)	94.7(3.1)	94.6(2.8)	94.4(3.2)	90.2(3.4)	92.9(2.2)	94.7(3.2)	93.4(3.5)	92.6(2.8)	93.3(3.5)
	Recall	92.8(2.3)	85.8(4.7)	82.5(5.0)	83.0(4.1)	84.1(2.9)	82.2(5.3)	94.2(2.2)	86.0(3.1)	80.9(2.8)	81.5(3.2)	83.7(3.3)	83.4(3.2)
I	F1	95.3(0.9)	93.3(1.3)	93.5(1.6)	93.5(1.9)	94.1(1.4)	94.1(1.5)	96.8(0.9)	93.2(2.3)	94.1(1.3)	95.4(1.2)	95.3(1.5)	95.1(1.0)
	Precision	93.3(1.3)	96.0(2.0)	95.1(2.0)	94.8(2.2)	96.0(1.4)	94.6(2.2)	95.6(1.6)	95.8(1.6)	95.8(1.6)	96.1(1.7)	96.4(1.7)	95.6(1.7)
	Recall	97.4(0.9)	90.9(2.2)	92.0(2.2)	92.3(2.3)	92.3(1.7)	93.6(1.5)	98.0(1.4)	90.9(4.1)	92.6(2.3)	94.9(2.6)	94.3(2.4)	94.7(2.3)
J	F1	95.4(1.2)	93.2(1.5)	93.4(1.5)	92.1(1.9)	93.2(1.6)	93.6(1.6)	95.0(1.1)	91.3(2.1)	91.2(2.0)	91.2(1.8)	92.2(1.7)	92.7(1.5)
	Precision	94.3(1.1)	95.6(1.4)	95.4(1.3)	95.1(1.8)	95.8(2.0)	95.4(1.1)	92.5(1.4)	93.0(1.4)	93.7(2.6)	92.8(2.3)	92.9(1.7)	93.4(1.9)
	Recall	96.5(1.9)	91.1(2.8)	91.6(2.5)	89.6(3.4)	91.0(3.3)	92.0(2.8)	97.7(1.5)	89.7(3.1)	89.0(3.0)	89.7(2.7)	91.6(3.2)	92.0(2.7)
K	F1							99.4(0.1)	-	99.3(0.2)	99.3(0.2)	99.3(0.1)	99.4(0.1)
	Precision							99.4(0.2)	-	99.6(0.2)	99.6(0.2)	99.6(0.1)	99.6(0.1)
	Recall							99.5(0.2)	-	99.0(0.2)	98.9(0.4)	98.9(0.2)	99.2(0.2)
L	F1							98.1(0.3)	-	*	*	98.4(0.8)	99.1(0.1)
	Precision							97.2(0.3)	-	*	*	98.6(1.2)	99.2(0.0)
	Recall							99.1(0.2)	-	*	*	98.2(0.4)	99.0(0.2)
M	F1							99.6(0.1)	-	99.6(0.0)	99.3(0.3)	99.5(0.0)	99.6(0.0)
	Precision							99.5(0.1)	-	99.6(0.0)	99.7(0.1)	99.7(0.0)	99.6(0.0)
	Recall							99.8(0.0)	-	99.5(0.1)	99.0(0.7)	99.4(0.1)	99.6(0.1)

Table 21: F1, Precision, and Recall of the majority class generated by the RoBERTa and LLama classifiers, along with the best undersampling methods. The number in parentheses represents the 95% confidence interval. Cells with "-" represent methods with an execution time with undersampling greater than the classification time without it and were disregarded. Cells with "*" represent methods that could not be executed due to a memory allocation failure during the undersampling step.

Another factor that influences performance is the complexity of the undersampling algorithms themselves. For instance, the CNN algorithm has a complexity of $O(n^3)$ and, consequently, incurs a very high execution time in large datasets, affecting overall performance. It is worth noting that the proposed methods achieve a balance between selection speed and the removal of a significant amount of data, ensuring a reduction in total classification time without compromising model effectiveness.

Another point that can be observed when analyzing Table 20 is that among the methods with the same removal rate –NM1, NM2, CC_NN, IHT, UBR, and E2SC_US– our proposed methods are the ones that show the best performance in terms of effectiveness and bias reduction. This suggests that the choice of which instances to remove is a key factor in achieving better results, highlighting that not only the quantity but also the quality of the selection makes a difference.

L Classes F1 Behavior

Tables 21 and 22 present F1, precision, and recall values for the best undersampling methods found in section 5 (UBR, E2SC_US, CNN, NM1, NM2) and for the models without undersampling (NoUnder) across all datasets, considering the results obtained using RoBERTa and LLama as classifiers. Average values are presented with confidence intervals in parentheses, based on results from 10 (for datasets A to J) or 5 folds (for datasets K to M) for the majority and minority classes, respectively. As previously mentioned, we did not consider RoBERTa for the large datasets.

By analyzing Tables 21 and 22, we observe that by applying undersampling approaches, the classification model tends to increase the number of instances attributed to the minority class, which increases this class's recall, but with a corresponding reduction in precision. On the other hand, as the

Minority Class													
dataset	metrics	RoBERTa						LLama					
		NoUnder	CNN	NM1	NM2	UBR	E2SC_US	NoUnder	CNN	NM1	NM2	UBR	E2SC_US
A	F1	86.6(0.9)	86.7(1.6)	87.2(1.0)	87.4(0.9)	87.1(1.8)	87.1(1.3)	90.5(1.2)	89.3(1.8)	89.7(1.5)	90.4(1.7)	90.6(1.4)	90.9(1.6)
	Precision	88.0(2.1)	82.9(3.0)	85.6(1.7)	85.4(1.8)	86.1(2.1)	85.3(2.1)	91.2(2.3)	86.2(2.9)	88.0(2.4)	88.1(2.4)	89.1(2.5)	88.9(2.0)
	Recall	85.4(2.5)	91.2(2.5)	88.9(2.6)	89.6(2.2)	88.3(2.7)	89.1(2.8)	89.9(2.3)	92.7(2.7)	91.7(2.6)	93.0(2.3)	92.4(3.2)	93.0(2.5)
B	F1	86.9(0.9)	86.5(2.0)	86.3(1.4)	86.2(0.9)	86.4(1.1)	87.2(1.2)	92.0(0.9)	91.6(1.6)	89.6(5.4)	91.7(0.7)	92.4(0.8)	91.6(0.8)
	Precision	88.4(1.4)	83.5(3.5)	85.5(2.0)	84.8(1.1)	85.8(2.0)	87.0(1.9)	92.3(1.4)	88.9(2.8)	88.1(6.7)	89.1(1.7)	90.7(1.2)	90.5(1.0)
	Recall	85.6(1.7)	90.0(2.2)	87.1(2.3)	87.7(1.9)	87.1(1.2)	87.4(1.4)	91.8(2.0)	94.6(1.8)	91.4(3.8)	94.6(1.5)	94.3(1.1)	92.6(1.7)
C	F1	91.9(1.4)	92.5(1.3)	93.1(1.8)	92.9(1.4)	93.2(1.2)	92.1(1.5)	97.1(1.3)	97.2(0.8)	96.9(1.4)	97.9(0.8)	97.5(1.3)	97.6(1.1)
	Precision	92.8(2.1)	91.1(1.9)	92.0(2.2)	92.6(2.5)	92.5(2.0)	90.8(2.1)	97.5(1.3)	96.3(1.8)	96.0(1.9)	97.1(1.8)	97.0(1.8)	97.6(1.3)
	Recall	91.2(2.3)	94.1(1.7)	94.3(2.2)	93.2(2.0)	94.1(2.4)	93.6(2.6)	96.7(2.3)	98.2(1.1)	97.7(1.6)	98.8(1.0)	98.0(1.2)	97.8(1.5)
D	F1	86.3(1.6)	85.4(1.2)	84.9(1.9)	83.2(1.8)	86.2(1.5)	86.0(2.1)	88.8(1.3)	88.8(1.3)	88.3(1.6)	85.3(2.0)	88.7(1.9)	88.5(2.1)
	Precision	88.1(2.0)	82.8(2.2)	84.0(2.6)	79.0(2.2)	83.3(2.0)	84.1(3.2)	90.7(1.9)	87.8(1.7)	86.7(2.0)	80.2(3.2)	87.5(3.4)	87.4(2.8)
	Recall	84.7(2.3)	88.5(2.5)	86.0(3.0)	88.1(2.1)	89.3(1.9)	88.2(2.2)	87.1(2.7)	89.9(2.4)	90.0(1.9)	91.1(1.4)	90.1(1.8)	89.6(2.4)
E	F1	85.9(2.6)	86.0(2.0)	84.7(2.3)	85.7(2.5)	85.7(2.0)	85.8(2.2)	91.6(1.9)	90.5(1.9)	88.7(2.3)	88.6(2.4)	89.4(2.1)	89.7(2.1)
	Precision	87.4(4.0)	83.7(3.8)	79.8(3.6)	82.1(3.6)	81.4(2.7)	81.7(3.1)	93.0(2.7)	88.1(3.4)	83.7(4.0)	85.4(3.2)	85.2(3.4)	86.7(2.9)
	Recall	84.6(3.5)	88.7(2.4)	90.5(2.7)	89.8(3.2)	90.6(2.5)	90.5(2.7)	90.4(2.8)	93.2(1.9)	94.5(1.9)	92.3(2.7)	94.1(1.8)	93.1(2.7)
F	F1	91.9(1.4)	90.5(1.4)	89.9(1.4)	89.8(1.6)	90.7(1.8)	90.6(1.7)	93.4(1.0)	92.5(1.7)	91.8(1.3)	91.9(1.1)	92.3(1.3)	93.0(1.3)
	Precision	93.4(1.4)	86.8(2.1)	86.9(2.2)	86.5(2.1)	87.7(2.5)	87.2(2.3)	94.6(1.8)	89.7(2.6)	88.5(2.1)	89.1(1.5)	89.7(2.6)	91.2(1.8)
	Recall	90.5(2.1)	94.7(1.3)	93.3(1.0)	93.4(1.5)	93.8(1.8)	94.4(1.6)	92.3(1.9)	95.5(1.3)	95.5(1.1)	94.9(1.4)	95.1(1.2)	94.8(1.6)
G	F1	85.5(2.2)	85.6(2.5)	85.4(2.9)	84.9(2.3)	84.6(1.3)	83.8(2.1)	87.9(2.7)	86.7(2.2)	85.6(2.1)	86.2(2.3)	85.9(1.7)	86.3(1.9)
	Precision	86.4(2.4)	81.3(3.3)	80.3(3.7)	78.5(3.3)	78.5(2.5)	77.8(2.5)	89.0(1.8)	81.8(2.3)	80.7(3.1)	80.7(3.7)	80.7(2.8)	81.9(3.3)
	Recall	84.7(2.8)	90.4(2.4)	91.4(2.7)	92.5(1.1)	91.8(1.9)	90.8(2.4)	87.0(4.0)	92.4(2.7)	91.4(2.4)	92.8(1.8)	92.0(2.1)	91.5(2.8)
H	F1	75.8(8.0)	76.4(6.9)	73.2(5.8)	74.4(5.8)	75.3(5.1)	73.6(6.2)	75.6(8.7)	74.5(3.7)	72.7(3.9)	70.9(5.6)	71.7(3.5)	72.4(5.2)
	Precision	79.0(5.9)	69.8(8.1)	64.8(5.7)	65.8(5.7)	66.9(4.8)	65.0(7.2)	81.9(6.7)	68.8(4.7)	62.9(3.3)	62.4(3.8)	65.1(3.5)	65.0(4.6)
	Recall	73.8(10.4)	85.2(7.8)	85.2(8.9)	86.7(8.6)	86.7(7.1)	86.2(8.3)	71.4(11.2)	81.9(5.7)	87.1(8.0)	83.3(10.2)	81.0(7.9)	82.9(9.5)
I	F1	71.0(5.9)	69.6(6.0)	68.5(7.8)	68.1(9.1)	72.3(6.1)	69.0(8.8)	80.8(6.5)	70.1(7.5)	71.7(5.5)	76.4(5.6)	76.8(6.2)	74.6(4.7)
	Precision	82.4(5.1)	62.8(5.4)	64.2(7.6)	64.6(8.8)	66.5(6.3)	67.5(7.3)	89.3(7.3)	65.1(10.8)	67.6(6.3)	77.1(9.4)	74.1(7.2)	75.8(8.9)
	Recall	63.0(7.6)	79.7(10.8)	74.5(10.6)	73.0(11.4)	79.7(7.0)	71.4(11.8)	75.7(9.6)	78.8(8.3)	78.0(9.1)	78.8(9.9)	81.2(9.2)	76.5(9.7)
J	F1	66.7(8.0)	62.3(7.4)	62.6(6.7)	57.8(8.3)	62.5(7.9)	63.7(6.9)	57.5(11.0)	50.6(9.9)	50.9(11.2)	47.9(11.2)	51.2(9.6)	53.7(10.2)
	Precision	74.9(12.7)	56.4(8.1)	57.3(8.0)	51.4(8.3)	57.4(10.7)	59.8(10.4)	77.3(13.7)	47.4(12.5)	45.6(10.5)	44.2(10.0)	52.1(13.6)	53.2(11.8)
	Recall	61.4(8.0)	71.8(9.9)	70.7(8.6)	68.6(12.1)	72.6(12.9)	70.6(7.6)	47.6(10.9)	55.6(8.3)	60.0(16.5)	53.9(15.4)	53.8(12.0)	56.8(12.9)
K	F1							94.0(1.6)	-	92.8(1.6)	92.3(2.1)	92.4(1.3)	93.6(1.4)
	Precision							94.5(1.7)	-	89.9(2.0)	89.2(3.3)	89.1(1.7)	91.4(2.4)
	Recall							93.6(2.2)	-	96.0(1.7)	95.8(2.1)	96.0(1.3)	96.0(0.8)
L	F1							70.0(4.7)	-	*	*	78.6(12.1)	87.7(1.2)
	Precision							82.5(4.6)	-	*	*	76.4(7.5)	86.3(2.2)
	Recall							60.7(4.6)	-	*	*	81.1(17.2)	89.2(0.3)
M	F1							84.5(3.3)	-	83.8(1.6)	77.1(8.2)	82.9(1.1)	85.2(1.0)
	Precision							89.9(2.1)	-	82.0(2.6)	69.8(14.3)	79.4(1.7)	84.7(2.5)
	Recall							79.8(4.3)	-	85.6(1.0)	87.2(2.9)	86.8(1.6)	85.8(1.2)

Table 22: F1, Precision, and Recall of the minority class generated by the RoBERTa and LLama classifiers, along with the best undersampling methods. The number in parentheses represents the 95% confidence interval. Cells with "-" represent methods with an execution time with undersampling greater than the classification time without it and were disregarded. Cells with "*" represent methods that could not be executed due to a memory allocation failure during the undersampling step.

datasets	CNN	NM1	NM2	CC_NN	SBC	NM3	OBU	NCR	IHT	TL	OSS	ENN	RENN	ALLKNN	UBR	E2SC_US
A	1.3(0.2)	1.1(0.1)	1.2(0.2)	1.2(0.1)	1.2(0.1)	1.1(0.1)	1.4(0.1)	1.3(0.2)	1.2(0.2)	0.9(0.1)	1.0(0.1)	1.4(0.2)	1.4(0.2)	1.3(0.2)	1.2(0.1)	1.1(0.1)
B	1.0(0.1)	1.2(0.1)	1.1(0.1)	1.2(0.1)	1.4(0.2)	1.6(0.3)	1.4(0.2)	1.0(0.1)	1.2(0.2)	1.0(0.1)	1.0(0.1)	1.5(0.2)	1.5(0.2)	1.5(0.2)	1.2(0.1)	1.3(0.1)
C	1.5(0.3)	1.3(0.3)	1.1(0.3)	1.3(0.4)	1.7(0.3)	1.3(0.3)	1.3(0.2)	1.3(0.2)	1.2(0.4)	0.9(0.2)	0.9(0.2)	1.2(0.3)	1.1(0.3)	1.2(0.3)	1.1(0.3)	1.4(0.3)
D	1.5(0.2)	1.5(0.2)	1.5(0.2)	1.6(0.3)	1.6(0.2)	1.5(0.3)	1.3(0.3)	1.6(0.3)	1.4(0.3)	1.2(0.2)	1.2(0.2)	2.0(0.4)	1.9(0.3)	1.6(0.4)	1.6(0.3)	1.7(0.3)
E	1.2(0.1)	1.3(0.2)	1.3(0.1)	1.5(0.1)	1.9(0.4)	1.5(0.2)	1.4(0.1)	1.7(0.3)	1.2(0.2)	0.9(0.0)	1.0(0.0)	1.8(0.3)	1.8(0.3)	1.8(0.3)	1.4(0.2)	1.6(0.2)
F	1.4(0.4)	1.6(0.4)	1.4(0.4)	1.5(0.4)	1.5(0.4)	1.5(0.4)	1.3(0.4)	1.2(0.3)	1.3(0.4)	1.0(0.3)	1.0(0.2)	3.0(4.2)	1.3(0.4)	2.8(3.8)	1.4(0.4)	1.5(0.4)
G	1.5(0.1)	1.7(0.2)	1.6(0.2)	1.9(0.2)	1.8(0.3)	1.8(0.3)	1.6(0.2)	1.3(0.2)	1.7(0.2)	1.1(0.2)	1.1(0.2)	1.4(0.3)	1.8(0.3)	1.6(0.2)	1.8(0.2)	2.0(0.3)
H	1.5(0.2)	1.9(0.3)	1.6(0.3)	1.6(0.2)	1.6(0.3)	1.6(0.3)	1.4(0.3)	1.4(0.3)	1.7(0.2)	1.0(0.2)	1.1(0.2)	1.6(0.3)	1.6(0.4)	1.8(0.3)	1.8(0.2)	1.7(0.2)
I	2.3(0.4)	1.8(0.4)	1.6(0.3)	1.7(0.4)	3.0(0.7)	2.1(0.2)	1.4(0.3)	0.9(0.2)	1.7(0.5)	1.0(0.1)	0.8(0.2)	1.1(0.3)	1.1(0.2)	1.0(0.2)	1.7(0.3)	1.7(0.4)
J	2.4(0.2)	1.7(0.2)	1.7(0.3)	1.5(0.2)	3.5(0.4)	3.1(0.3)	1.5(0.2)	1.5(0.1)	1.7(0.2)	1.1(0.1)	1.0(0.2)	1.4(0.2)	1.2(0.2)	1.4(0.1)	1.6(0.2)	1.8(0.1)

Table 23: Speedup results in the total cost (time) for generating the models using the RoBERTa classifier in conjunction with undersampling approaches. The greener the cell, the greater the reduction in total training time compared to the approach without undersampling. The redder, the longer the time. Numbers in parentheses represent 95% confidence intervals

model reduces the number of documents attributed to the majority class, the opposite effect occurs: recall decreases while precision increases. These changes occur in a balanced way so that recall compensates for precision, keeping the F1-score of the classes practically unchanged before and after undersampling, as evidenced in the tables below and in the results presented in the main article. Despite maintaining overall classification effectiveness, the decrease in documents assigned to the majority class is responsible for a significant improvement (decrease) in the model's bias. These results are consistent across all datasets and classifiers.

M Speedup with confidence interval

Table 23 is identical to Table 4, except for the confidence interval values, which were omitted from Table 4 due to space limitations and to improve readability. As previously mentioned, we did not consider RoBERTa for the large datasets.