Practical Assignment
Machine Learning I

Carolina Proença – 202306055
Eduarda Neves – 202307178
Maria Morais – 202304201

Goals

- Gain a theoretical and empirical understanding of the XGBoost algorithm
- Learn how to improve its effectiveness in the face of data-related challenges(class imbalance) that affect its performance

Outline of the approach

- Choose and organize the algorithm
- Choose the data characteristic to tackle
- Reflect about the changes we can implement
- Implement and evaluate the changes of the proposed variant
- Compare the results

Summary of results

- Originally, the model mostly predicted the majority class due to imbalance and absence of class-aware mechanisms.
- The proposed variant moderately improves performance on the minority class while maintaining overall model quality.

Selected Algorithm:
Gradient Boosting Trees – XGBoost

We implemented a classification algorithm based on Gradient Boosting Trees, inspired by XGBoost.

Gradient Boosting is an ensemble learning technique that combines multiple weak learners — typically shallow decision trees — in a sequential manner. Each new tree is trained to correct the errors made by the previous one, gradually improving overall model performance.

Instead of fitting a single complex model, Gradient Boosting builds a series of simple models that focus on the residuals of the combined previous models. The final prediction is a weighted sum of the outputs from all trees.

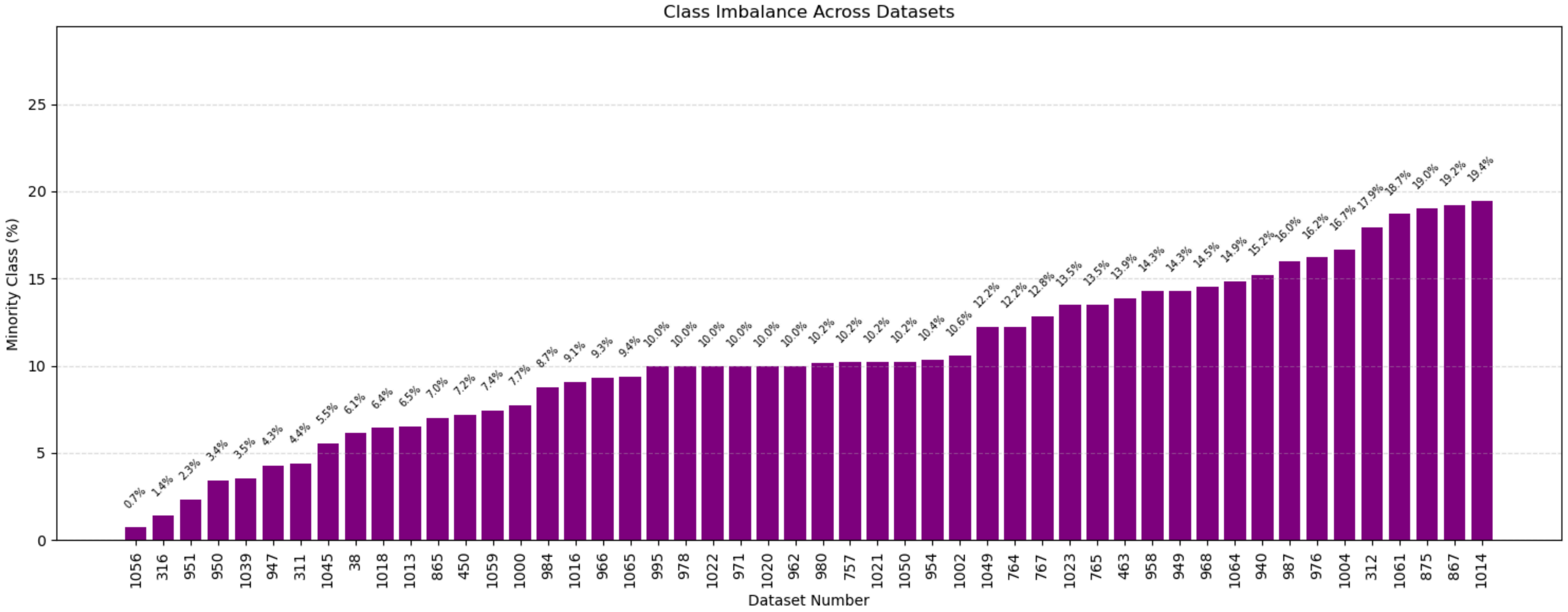
This approach allows the model to capture complex patterns with high accuracy while maintaining flexibility and interpretability. XGBoost extends this by optimizing efficiency, regularization, and scalability.

Dataset Overview

- 50 binary classification datasets with class imbalance
- Wide variety in:
  - Number of samples: from 100 to 9961
  - Number of features: from 3 to 1618
  - Feature types: numerical, categorical, binary

Preprocessing

- Removed empty columns
- Filled missing values using:
  - Mean (numerical)
  - Mode (categorical)
- Applied Label Encoding for categorical variables
- Ensured binary target labels: converted to 0/1



- Most of the datasets show a strong imbalance, with the minority class representing less than 20% of the examples.
- This phenomenon significantly influences the performance of the classifiers and was the main focus of our analysis.

Performance estimation methodology

Stratified K-Fold Cross-Validation

- The performance of the model is evaluated using Stratified K-Fold Cross-Validation.
- The number of folds is dynamically set to 3 or less, depending on the number of samples in the minority class (to avoid folds with too few positive examples).
- This method ensures that each fold maintains the original class distribution, which is essential for imbalanced classification problems.

In each iteration:

- The model is trained on 66.7% of the data and tested on the remaining 33.3%.
- Predictions are evaluated with the minority class treated as the positive class.

Evaluation Metrics

- F1 Score(minor class): how well the model correctly identifies the minority class
- ROC AUC: model's overall ability to distinguish between the two classes
- PR AUC: how the model balances precision and recall

Discussion of the behavior of the algorithm concerning class imbalance

Before introducing any modifications to handle class imbalance, the original version of the algorithm struggled significantly on datasets with strong imbalance. In many cases, the F1 Score for the minority class was 0, meaning the model failed to identify any instances of the minority class.

Interestingly, ROC AUC and PR AUC scores were sometimes deceptively high, especially on datasets with extreme imbalance. This occurred because the model tended to always predict the majority class, and due to the overwhelming number of majority examples, it still achieved good scores in metrics that are influenced by overall ranking or class distribution.

This gave the illusion of strong performance, when in reality, the classifier was not effectively learning the decision boundary for the minority class. The model prioritized accuracy on the majority class, ignoring the minority class due to the lack of corrective mechanisms like class weights or modified split criteria.

Why modify the algorithm?

- The original model performed poorly on some imbalanced datasets, often failing to predict any minority class instances (F1 score= 0).
- Although ROC AUC and PR AUC appeared high, this was misleading due to constant majority class predictions.
- There was a need to enhance sensitivity to the minority class without sacrificing general performance.

Dataset	F1 Score (minor class)	ROC AUC	PR AUC
311_oil_spill_cleaned.csv	0.2620439988	0.900935652	0.513797
1059_ar1_cleaned.csv	0.0	0.722498814	0.311354
984_analcatdata_draft_cleaned.csv	0.0512820512	0.529334123	0.127464
865_analcatdata_neavote_cleaned.csv	0.0	0.654121863	0.125016
1002_ipums_la_98-small_cleaned.csv	0.0	0.152757063	0.061058
1014_analcatdata_dmft_cleaned.csv	0.0	0.547576507	0.224320

Goals of the proposal

- Improve performance on the minority class, especially in F1 Score and PR AUC.
- Ensure the algorithm makes more balanced predictions.
- Test whether simple adjustments (e.g., class weights, alternative gain functions) can provide meaningful improvements.

Modifications implemented

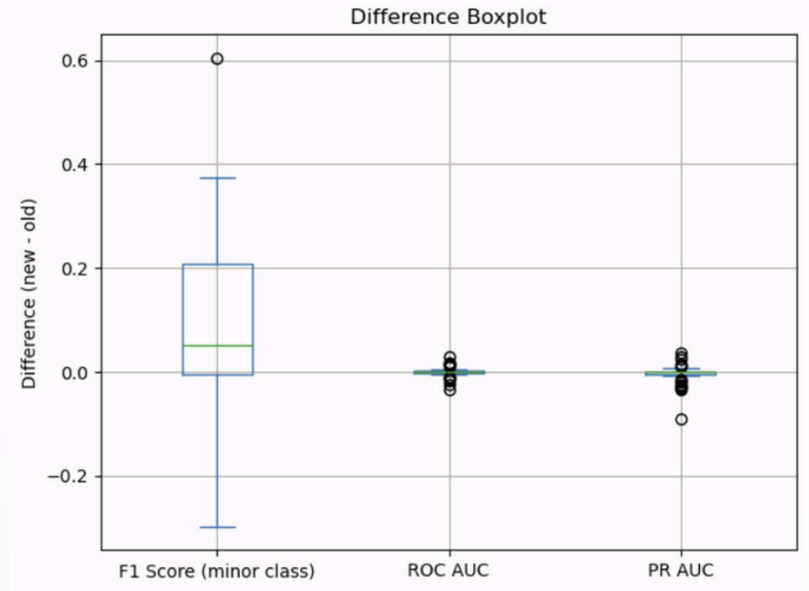
1. Class Weighting
    - Computed weights inversely proportional to class frequencies.
    - Used to modify the loss function, giving more importance to minority class errors during training.

2. Hellinger Distance Gain
    - Replaces the default split criterion only in the first tree with Hellinger Gain.
    - Focuses on maximizing class separability, especially benefiting the minority class, even when it has very few examples.
    - Encourages early splits that highlight minority patterns, improving how the model initiates learning on imbalanced data.

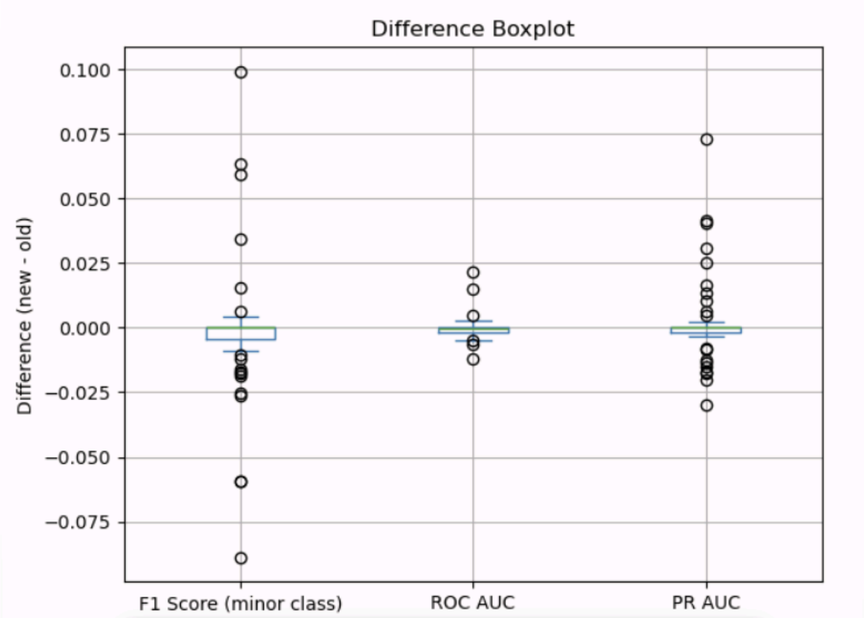
3. Combined Use
    - Evaluated both methods separately and together to understand their individual and joint effects on performance.

Hyperparameters of the algorithm	Description
n_estimators = 50	Number of trees (boosting iterations)
learning_rate = 0.1	Contribution of each new tree to the model
max_depth = 2	Maximum depth of each tree
min_samples_split = 5	Minimum number of samples to split a node
max_features = 10	Maximum number of features considered per tree
Added after changes:	
use_weights = True	Applies weights to instances
use_hellinger = True	Uses the Hellinger distance as a criterion for the split

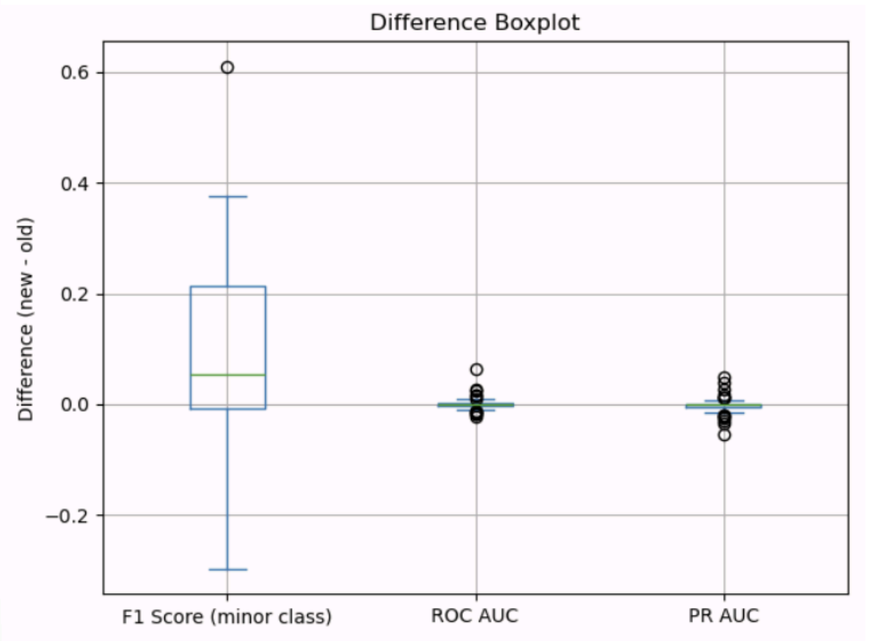
Analysis of results – Boxplots



Class Weights



Hellinger Gain



Class Weights + Hellinger Gain

Analysis of results

The analysis of the three scenarios shows that the class weights technique is mainly responsible for improving the model's performance in problems with class imbalance. Applied alone, it generated a statistically significant increase in the F1 Score of the minority class, without negatively impacting other metrics.

On the other hand, the technique based only on splitting adjustments showed no statistical benefit in any metric, even having a slight negative impact on the F1 Score. When combined, the two techniques maintain the gains from weighting, but do not bring significant additional improvements.

Therefore, the recommended strategy is to focus on using weights to deal with the imbalance, considering that additional splits do not add measurable value to the model's performance.

Discussion of hypothesized effect

The initial hypothesis was that class imbalance would negatively affect the model's ability to correctly classify minority class samples, and that specific techniques (e.g., class weighting or modified splits) would mitigate this issue.

The experimental results confirm this hypothesis partially: applying class weighting clearly improved the F1 Score for the minority class, supporting the effectiveness of this technique. In contrast, splitting adjustments alone failed to bring measurable benefits, suggesting that not all imbalance-handling methods are equally effective in practice.

Conclusion

This project focused on understanding and adapting the XGBoost algorithm to tackle class imbalance.

We explored modifications to the algorithm and evaluation methods specifically designed for imbalanced datasets.

The assignment deepened our understanding of how subtle algorithmic changes can impact learning dynamics and highlighted the trade-offs in model behavior.

Anexes

Datasets

- class_imbalance
- cleaned_datasets

Interpretation

- interpretation_of_results.pdf: detailed analysis of the results with the modifications implemented

Results

- resultados_gbm_antes.csv: results of the unmodified algorithm
- resultados_gbm_weights.csv: results of the algorithm using class weights
- resultados_gbm_splits.csv: results of the algorithm using hellinger gain
- resultados_gbm_weights_splits.csv: results of the algorithm using both methods
- sklearn_comparison_results.csv:results of the scikit-learn’s algorithm