When Balancing Harms: Structural Conditions That Degrade Android Malware Detection After Class Imbalance Correction

Lucas Ferreira Areias de Oliveira[1], Anna Luiza Gomes da Silva[1], Angelo Diniz[1], Diego Kreutz[1], Dionatan R. Schmidt[1], Rodrigo Mansilha[1], Kayuã Oleques Paim[2]

[1]AI Horizon Labs
Programa de Pós-Graduação em Engenharia de Software (PPGES)
Universidade Federal do Pampa (UNIPAMPA)

[2]Universidade Federal do Rio Grande do Sul (UFRGS)

{lucasareias, annaluiza, angelonogueira}.aluno@unipampa.edu.br
{diegokreutz, dionatanschmidt, rodrigomansilha}@unipampa.edu.br
kayua.paim@inf.ufrgs.br

Abstract. *Class imbalance is a recurrent challenge in Android malware detection, and class balancing is often adopted to improve minority-class detection. This paper compares original, dimensionality-reduced, and class-balanced versions of 11 Android malware datasets, using Recall, F1-score, and execution time. Dimensionality reduction followed by class balancing improved performance in 7 of the 11 datasets; in MH100k, this combination increased Recall from 0.63 to 0.99 and reduced execution time by approximately 99%. However, class balancing reduced performance in 4 datasets, especially in cases characterized by high sparsity, reduced diversity, or near-balanced class distributions, indicating that it should be empirically validated rather than applied by default.*

1. Introduction

Machine learning-based Android malware detection depends on the quality of the data used for training and evaluating models [Sun et al. 2018]. In tabular approaches, Android applications are represented by features extracted from APKs, such as permissions, API calls, declared components, and structural metadata. From this information, supervised classifiers learn to distinguish benign and malicious applications. In this context, the effects of imbalance on learning are not uniform: they depend on the class structure and the adopted classifier [Das et al. 2022].

However, datasets in this domain often exhibit characteristics that hinder supervised learning, including class imbalance, high sparsity, sample duplication, low diversity, and redundant attributes [Ge et al. 2021]. In particular, class imbalance is a recurring problem in supervised classification tasks, as it can favor the majority class and reduce the detection capability for the class of interest [He and Garcia 2009, Chen et al. 2024].

This favoring of the majority class is particularly critical in security, since false negatives correspond to undetected threats and can compromise the practical utility of the model . And while true and false positives, for instance, provide a more detailed picture of a system's performance, they can also disguise the actual precision when the prevalence of attacks is low [Arp et al. 2022]. For this reason, this study adopts Recall of the malicious class as the primary metric. F1-score is used as a complementary metric,

allowing us to assess whether Recall gains occur without sharp degradation of the balance between precision and recall.

The heterogeneity of Android malware datasets reinforces the need for empirical analysis. The MH100k [Bragança et al. 2023], for example, contains 101,934 samples and 24,833 features, with approximately nine benign applications for each malicious application. AndroCrawl [Sisto 2013] also exhibits strong class asymmetry. On the other hand, variants of DefenseDroid [Colaco et al. 2021] have distributions close to balance, but still present challenges associated with dimensionality and sparsity. Thus, the effect of balancing should not be assumed to be uniform across different databases.

Given this heterogeneity, class balancing techniques are often used to mitigate this bias [Wongvorachan et al. 2023]. Undersampling methods reduce or filter samples from the majority class, and hybrid strategies may combine this reduction with cleaning steps, removing noisy, redundant, or ambiguous examples before resampling. Oversampling methods, in turn, increase the representation of the minority class through replication or synthetic generation. None of these techniques guarantees automatic performance improvement: in datasets with low diversity, high sparsity, or sensitive decision boundaries, these techniques may replicate uninformative patterns, remove relevant samples, or generate synthetic examples with low discriminatory contribution.

We demonstrate that class balancing is not a neutral preprocessing step: across 11 tabular Android malware datasets, it produces Recall gains in 7 and performance degradation in 4, with sparsity and diversity as the structural factors that predict which outcome occurs. Hybrid cleaning strategies (ENN + RandomUnder) consistently outperform pure undersampling or oversampling when imbalance is the dominant factor, while also reducing execution time by up to 99%.

The main contributions of this paper are:

- we demonstrate that class balancing improves detection in 7 of 11 tabular Android malware datasets and degrades it in 4, establishing, for the first time in this domain, structural conditions that reliably predict each outcome;
- we establish that hybrid strategies (ENN + RandomUnder) outperform pure undersampling in the majority of evaluated datasets while preserving an F1-score ≥ 0.70;
- we quantify the computational impact of balancing, showing that when it is combined with dimensionality reduction, execution time falls by up to 99% (from 28,583h to $\sim$130h in MH100k);
- we introduce sparsity and diversity as diagnostic indicators, representing the first structural predictors of balancing-induced degradation proposed in the Android malware literature.

2. Related Work

Table 1 positions this study in relation to recent works, highlighting the domain explored, the balancing strategies evaluated, the number of datasets, the evaluation metrics, and the use of structural indicators (sparsity and diversity) to interpret the effects of balancing.

The related work can be organized into three groups, according to the domain addressed and the experimental scope. The first group consists of **studies focused on**

Table 1. Positioning of this study in relation to recent works on class balancing and malware detection.

Work	Domain	Strategies Evaluated	# datasets	Classification Metrics	Structural Charact.
[Guan et al. 2021]	Android malware	Oversampling and undersampling	7	AUC, Precision, Recall, F1	No
[Akintola et al. 2022]	Android malware	Oversampling and undersampling	2	Accuracy, F-measure, AUC	No
[Haluška et al. 2022]	Cybersecurity and general	Oversampling, undersampling and hybrid strategies	23	PR AUC, ROC AUC, P-ROC AUC	No
[Wongvorachan et al. 2023]	Educational mining	Oversampling, undersampling and hybrid strategies	1	Accuracy, Precision, Recall, ROC-AUC	No
[Vairetti et al. 2024]	Big data	Oversampling, undersampling and hybrid strategies	35	G-mean, statistical tests and time	No
[John and Di Troia 2025]	Malware	Oversampling, undersampling and hybrid strategies	1	Accuracy, F1, AUC	No
This work	**Android malware**	**Oversampling, undersampling and hybrid strategies**	**11**	**Recall, F1-score and time**	**Yes**

Android malware. [Guan et al. 2021] and [Akintola et al. 2022] investigate the impact of imbalance specifically on tabular Android malware data, evaluating oversampling and undersampling strategies. Although they share the domain of this work, both consider a reduced number of datasets (7 and 2, respectively), which limits the generalization of conclusions to other detection contexts.

Comprehensive benchmarks in general domains. [Haluška et al. 2022] and [Vairetti et al. 2024] conduct large-scale evaluations, covering 23 and 35 datasets, respectively, with diverse strategies that include hybrid approaches. These works offer broad perspectives on the behavior of balancing techniques, but address general classification or big data problems, without considering the particularities of the Android malware domain.

Single-scope evaluations. [Wongvorachan et al. 2023] and [John and Di Troia 2025] compare multiple balancing strategies, respectively, in the domain of educational mining and in malware with opcode sequences, but each is restricted to a single dataset. Although they present comprehensive comparisons of the evaluated techniques, this limitation reduces the generalization of conclusions to other datasets in their respective domains.

A cross-cutting aspect across all groups is the absence of structural indicators of the datasets, such as sparsity and diversity, to aid in interpreting the effects of balancing. As summarized in Table 1, prior studies do not jointly consider the four dimensions analyzed in this work. This combination uniquely enables us to identify, for the first time, the dataset-level structural conditions under which class balancing systematically degrades rather than improves Android malware classifiers. This work fills this gap by combining four aspects rarely explored together: (i) the specific domain of Android malware with tabular data; (ii) multiple balancing strategies, including undersampling, hybrid strategies, and oversampling; (iii) a systematic evaluation across 11 datasets, using Recall,

F1-score, and execution time as analysis criteria; and (iv) the structural characterization of datasets using sparsity and diversity indicators, enabling the identification of conditions under which balancing tends to benefit or degrade learning.

3. Methodology

This section describes the experimental protocol adopted to evaluate the impact of balancing techniques on tabular Android malware datasets. The evaluation was structured to compare different resampling strategies under the same experimental pipeline, considering the same datasets, classifiers, metrics, and analysis criteria.

3.1. Research Questions

The evaluation was conducted based on five research questions:

RQ1: Does balancing improve Recall on Android malware datasets?
RQ2: Do different balancing strategies produce distinct effects on the evaluated datasets?
RQ3: In which scenarios does balancing degrade performance?
RQ4: Do diversity and sparsity help explain the effects of balancing?
RQ5: What is the impact on execution time?

These questions guide the comparison among the evaluated techniques and allow analysis not only of the final classifier performance but also of the conditions under which balancing is beneficial, harmful, or computationally advantageous.

3.2. Pipeline Overview

The experimental pipeline consisted of five main steps, as illustrated in Figure 1. Initially, the original datasets were organized and characterized regarding the number of samples, number of features, and class distribution. Next, a complementary dimensionality reduction step was applied to reduce computational cost and enable systematic execution of the experiments. Subsequently, balancing techniques were applied. Finally, the resulting datasets were evaluated with supervised classifiers, and the metrics were consolidated for comparison.

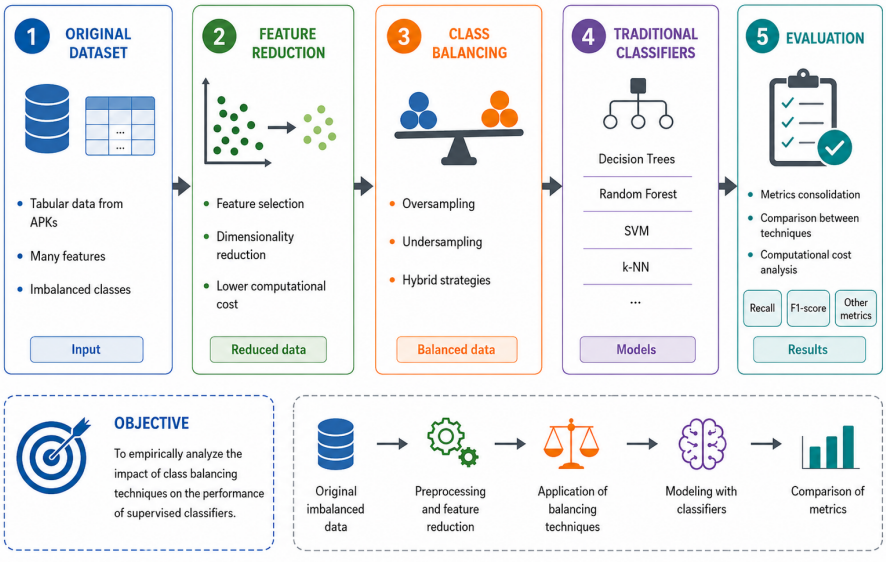


Figure 1. Overview of the experimental pipeline for empirical analysis of the impact of class balancing on tabular Android malware datasets.

3.3. Dataset Characterization and Complementary Feature Reduction

The experimental evaluation considered 11 tabular datasets related to Android malware detection. Table 2 presents their main characteristics before applying the balancing techniques, including the total number of samples, the original number of features, the number of features after reduction, the distribution between benign and malicious applications, and the reduction method adopted for each dataset.

Table 2. Datasets used in the experimental evaluation and feature reduction methods adopted.

Dataset	Year	Samples	Orig. features	Red. features	Reduction (%)	Benign	Malicious	Reduction method
MH100k	2023	101,934	24,833	93	99.63	92,134	9,800	Statistical Ranking
KronoDroid Real	2021	78,137	286	29	89.86	36,755	41,382	Statistical Ranking
KronoDroid Emulator	2021	63,991	276	25	90.94	35,246	28,745	Statistical Ranking
DREBIN-215	2018	15,031	215	64	70.23	9,476	5,555	RFE
DefenseDroid PRS	2021	11,975	2,877	144	94.99	5,975	6,000	Semidroid
DD API Katz	2021	10,476	6,002	300	95.00	5,222	5,254	RFE
DD API Degree	2021	10,476	6,002	300	95.00	5,222	5,254	RFE
DD API Closeness	2021	10,476	4,274	213	95.02	5,222	5,254	RFE
Android Permissions	2020	26,864	151	46	69.54	9,077	17,787	Semidroid
AndroCrawl	2012	96,744	141	12	91.49	86,574	10,170	Statistical Ranking
Adroit	2016	11,476	166	66	60.24	8,058	3,418	RFE

Although the main focus of this paper is class balancing, a complementary feature reduction step was adopted as experimental preprocessing. This step was motivated primarily by the computational cost associated with high-dimensionality datasets. MH100k, for example, contains 101,934 samples and 24,833 features, which corresponds to approximately 2.53 billion values to be processed. In this scenario, applying multiple balancing techniques directly to all features could compromise the experimental feasibility of the study.

Feature selection was based on the study MH-FSF, which proposes a modular framework for reproduction, experimentation, and evaluation of feature selection methods in the context of Android malware detection [Rocha et al. 2024]. The best previously evaluated methods were considered, including RFE, Semidroid, and Lasso. Additionally, a statistical ranking strategy investigated in a parallel study [Gomes et al. 2026] was included, based on the combination of criteria such as variance, correlation, Chi-Square, Mutual Information, and Random Forest importance.

For each dataset, the reduced version that maintained or improved predictive performance relative to the original version was adopted, as validated in [Gomes et al. 2026]. In that study, selection was integrated within each fold of stratified cross-validation, eliminating the risk of data leakage between training and testing. The comparison between the original and reduced sets, performed using the Wilcoxon test, indicated no statistically significant differences ($p > 0.05$ in all evaluated datasets). This result suggests that feature reduction did not produce statistically detectable degradation in predictive performance. Thus, reduction was used as an experimental enabler, not as a central contribution of this paper.

3.4. Balancing Techniques

On the reduced datasets described in the previous section, undersampling techniques, hybrid strategies, and oversampling techniques were evaluated. Undersampling techniques

reduce the number of samples from the majority class. Hybrid strategies combine this reduction with data cleaning steps, removing potentially problematic samples before adjusting the class distribution. Oversampling techniques, in turn, increase the representation of the minority class through replication or synthetic generation.

The evaluated techniques included RandomUnder, NearMiss, ENN, and Tomek Links as undersampling techniques; ENN + RandomUnder and Tomek Links + RandomUnder as hybrid strategies; and RandomOver, SMOTE, Borderline-SMOTE, and ADASYN as oversampling techniques. Hybrid strategies were considered because ENN and Tomek Links can remove problematic samples without necessarily correcting class imbalance. Combining them with RandomUnder allows a cleaning step before majority class reduction, assessing whether this sequence produces more suitable sets for classification. For this reason, these strategies were analyzed with particular attention, given their tendency to produce informative sets without relying on synthetic generation.

3.5. Experimental Procedure

We organized the evaluation into two stages, comparing original, reduced, and balanced versions of each dataset under identical pipeline conditions. We computed all metrics via stratified 5-fold cross-validation using the MalDataGen tool[1], ensuring consistent comparisons across dataset versions. All experiments were executed on an Intel Xeon Gold 6138 CPU with 96 GB of RAM, running Ubuntu 24.04 under WSL 2, using CPU only. We note that these results should not be interpreted as definitive operational performance estimates: in deployment, balancing would be applied exclusively to training folds. For this reason, conclusions are formulated conservatively, with greater emphasis on undersampling and hybrid strategies, while oversampling results are treated as exploratory.

This distinction is necessary because the evaluated oversampling techniques (SMOTE, ADASYN, Borderline-SMOTE, and RandomOver) create synthetic samples from real minority class points. When resampling is performed before cross-validation, artificial correlation between training and testing may arise, which tends to inflate reported metrics [Demircioğlu 2024, Blagus and Lusa 2013]. Undersampling techniques (RandomUnder, NearMiss, ENN, and Tomek Links) and hybrid strategies based on undersampling, on the other hand, only remove original samples, without generating new synthetic examples. This reduces the risk of artificial correlation compared to oversampling.

The first stage was exploratory and was used to compare the behavior of balancing techniques on each dataset. This stage considered the classifiers Random Forest, Support Vector Machine, K-Nearest Neighbors, Decision Tree, Naive Bayes, Gradient Boosting, and Stochastic Gradient Descent, in addition to generative models (Autoencoder and Variational Autoencoder) used as alternative synthetic sample generation strategies. Table 3 presents the main hyperparameters of the generative models, whose configurations differ from traditional classifiers, which were adopted with default parameters.

The second stage focused the analysis on Random Forest, adopted as the reference classifier for being widely used in comparative studies of tabular Android malware detection and for presenting a good balance between predictive performance and computational

[1]`https://github.com/MalwareDataLab/Maldatagen_additional_metrics.git`

Table 3. Main hyperparameters used in the experimental campaigns.

Model	Epochs	Batch size	Latent dimension	Main layers	Function / optimizer
Autoencoder	300	256	128	Encoder: 128–128; Decoder: 128–128	ELU / N/A
Variational Autoencoder	300	32	68	Encoder: 128–64; Decoder: 64–128	ReLU / Adam

cost. Thus, the classification model was kept fixed, allowing a more direct comparison of the effect of balancing techniques on the datasets. In this stage, the configurations retained from the exploratory stage, the hybrid strategies (ENN + RandomUnder and Tomek Links + RandomUnder), and two additional methods included after review of the experimental coverage (NearMiss and Borderline-SMOTE) were analyzed. Furthermore, the techniques retained in Stages 1 and 2 are presented in Section 4, as they represent intermediate results of the experimental process. The datasets, derived versions, and experimental results were organized in a GitHub repository[2] to support the traceability and reproducibility of the analyses presented in this study.

3.6. Evaluation Metrics

The evaluation considered predictive metrics and computational cost. For classifier performance, Recall and F1-score were used. Recall was adopted as the primary metric because, in the malware detection context, false negatives correspond to malicious applications not identified by the classifier. F1-score was used as a complementary metric, allowing verification whether Recall gains occurred with sharp degradation of the balance between precision and recall. In this study, an F1-score greater than or equal to 0.70 was adopted as the minimum operational criterion to consider a configuration acceptable. This threshold was defined based on preliminary results observed in the evaluated datasets and is not proposed as a universal reference. Its use served to identify cases where increased Recall is accompanied by relevant loss of overall performance. To avoid conclusions being conditioned to this specific value, degradation cases were additionally analyzed in terms of absolute percentage variation of F1-score relative to the reduced version without balancing.

In addition to predictive metrics, execution time was used as an indicator of computational cost. For each dataset, the execution time of the original version was adopted as a reference. This time was compared separately with the time obtained after dimensionality reduction and with the time obtained after applying balancing techniques. The percentage reduction in execution time was calculated according to Equation 1.

$$R_T = \left(1 - \frac{T_{\text{evaluated}}}{T_{\text{original}}}\right) \times 100 \tag{1}$$

where R_T represents the percentage reduction in execution time, T_{original} corresponds to the execution time of the original dataset, and $T_{\text{evaluated}}$ represents the execution time of the reduced or balanced version being evaluated. In addition to Recall and F1-score, two complementary analytical criteria were adopted:

- **Statistical significance:** the paired Wilcoxon test ($\alpha = 0.05$) was applied to verify

[2]`https://github.com/AILabs4All/datasets-mh30plus`

whether differences in Recall between original and balanced versions are statistically significant.

- **Overfitting diagnosis:** for each dataset, the gap between average Recall on the training set and on the test set was calculated using stratified 5-fold cross-validation. A significant increase in this gap after balancing indicates possible overfitting to synthetic or reduced samples.

4. Results and Discussion

4.1. Results of Balancing Technique Selection

The experiments were conducted considering 130 dataset variations (original, reduced, and balanced versions) and approximately 230 runs over about one month of continuous execution. Table 4 presents the best-performing techniques for each dataset in the two stages, organized side by side. It can be observed that no technique was dominant across all datasets, indicating that the effect of balancing varies according to the characteristics of each dataset. In the transition from Stage 1 to Stage 2, after the inclusion of hybrid strategies (ENN + RandomUnder, Tomek Links + RandomUnder) and additional methods NearMiss and Borderline-SMOTE, a consistent migration from pure undersampling techniques to hybrid strategies is observed, indicating that the prior cleaning step of samples tends to benefit the reference classifier.

Table 4. Techniques selected for each dataset in Stages 1 (multi-classifier screening) and 2 (Random Forest as reference).

	Stage 1			Stage 2		
Dataset	**Reduction**	**Undersampling**	**Oversampling**	**Reduction**	**Undersampling**	**Oversampling**
MH100k	Statistical	ENN	RandomOver	Statistical	ENN + RandomUnder	RandomOver
KronoDroid Real Device	Statistical	ENN	RandomOver	Statistical	ENN + RandomUnder	ADASYN
KronoDroid Emulator	Statistical	RandomUnder	SMOTE	Statistical	ENN + RandomUnder	SMOTE
DREBIN-215	RFE	RandomUnder	SMOTE	RFE	ENN + RandomUnder	SMOTE
DefenseDroid PRS	Semidroid	ENN	SMOTE	Semidroid	ENN + RandomUnder	RandomOver
DefenseDroid API Katz	RFE	RandomUnder	RandomOver	RFE	ENN + RandomUnder	RandomOver
DefenseDroid API Degree	RFE	RandomUnder	RandomOver	RFE	ENN + RandomUnder	RandomOver
DefenseDroid API Closeness	RFE	ENN	RandomOver	RFE	ENN + RandomUnder	RandomOver
Android Permissions	Semidroid	RandomUnder	ADASYN	Semidroid	ENN + RandomUnder	SMOTE
AndroCrawl	Statistical	RandomUnder	ADASYN	Statistical	NearMiss	SMOTE
Adroit	RFE	RandomUnder	SMOTE	RFE	NearMiss	RandomOver

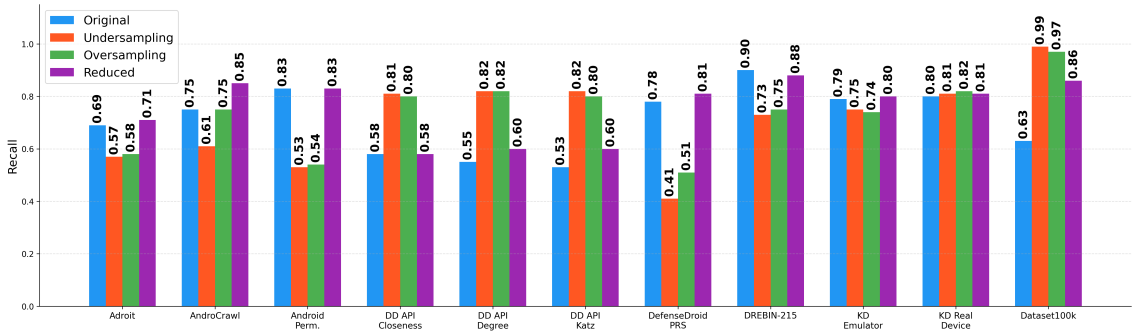
Table 5 presents the class distribution before and after applying balancing techniques for each of the 11 evaluated datasets.

Table 5. Examples of distribution before and after balancing.

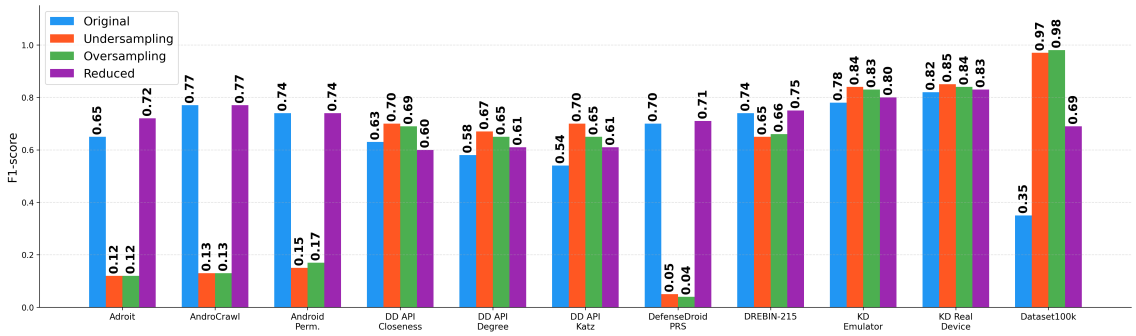
Dataset	**Original (B/M)**	**Undersampling (B/M + technique)**	**Oversampling (B/M + technique)**
MH100k	92,134 / 9,800 (9.4:1)	9,800 / 9,800 (1:1, ENN + RandomUnder)	92,134 / 92,134 (1:1, RandomOver)
KronoDroid Real	36,755 / 41,382 (1:1.1)	36,755 / 36,755 (1:1, ENN + RandomUnder)	40,873 / 41,382 (∼1:1, ADASYN)
KronoDroid Emulator	35,246 / 28,745 (1.2:1)	28,745 / 28,745 (1:1, ENN + RandomUnder)	35,246 / 35,246 (1:1, SMOTE)
DREBIN-215	9,476 / 5,555 (1.7:1)	5,555 / 5,555 (1:1, ENN + RandomUnder)	9,476 / 9,476 (1:1, SMOTE)
DefenseDroid PRS	5,975 / 6,000 (∼1:1)	4,827 / 4,827 (1:1, ENN + RandomUnder)	6,000 / 6,000 (1:1, RandomOver)
DD API Katz	5,222 / 5,254 (∼1:1)	5,200 / 5,200 (1:1, ENN + RandomUnder)	5,254 / 5,254 (1:1, RandomOver)
DD API Degree	5,222 / 5,254 (∼1:1)	5,200 / 5,200 (1:1, ENN + RandomUnder)	5,254 / 5,254 (1:1, RandomOver)
DD API Closeness	5,222 / 5,254 (∼1:1)	5,200 / 5,200 (1:1, ENN + RandomUnder)	5,254 / 5,254 (1:1, RandomOver)
Android Permissions	9,077 / 17,787 (1:2)	7,379 / 7,379 (1:1, ENN + RandomUnder)	17,787 / 17,787 (1:1, SMOTE)
AndroCrawl	86,574 / 10,170 (8.5:1)	10,170 / 10,170 (1:1, NearMiss)	86,574 / 86,574 (1:1, SMOTE)
Adroit	8,058 / 3,418 (2.4:1)	3,418 / 3,418 (1:1, NearMiss)	8,058 / 8,058 (1:1, RandomOver)

4.2. RQ1: Does balancing improve Recall on Android malware datasets?

Overall, the results show that balancing can improve Recall in datasets with greater class imbalance. However, this effect is not uniform, as some datasets showed performance degradation after balancing, possibly due to their structural characteristics. Figures 2a and 2b present, respectively, the Recall and F1-score results obtained for the original, reduced, and balanced versions of each dataset.



(a) Recall per dataset for original, reduced, and balanced versions.



(b) F1-score per dataset for original, reduced, and balanced versions.

Figure 2. Comparison of Recall and F1-score among original, reduced, and balanced versions of datasets.

The largest gain was observed in MH100k, a dataset with an approximate imbalance of 9.4:1 between benign and malicious classes. In the original version, the classifier performed poorly, with Recall of 0.63 and F1-score of 0.35. After dimensionality reduction, performance increased to Recall of 0.86 and F1-score of 0.69. The best result was obtained by applying the hybrid strategy ENN + RandomUnder on the reduced version, raising Recall to 0.99 and F1-score to 0.97. A similar result was also achieved by the RandomOver technique, although the latter should be interpreted with the methodological caveat discussed in Section 4.7. This result indicates that, in MH100k, correcting class asymmetry was associated with a significant improvement in malicious class detection. Gains were also observed in the DefenseDroid API Closeness, Degree, and Katz datasets. In these bases, the original Recall was below 0.60 and increased to approximately 0.80, accompanied by improvement in F1-score.

On the other hand, four datasets did not show simultaneous gains in Recall and F1-score. In three of them, degradation is direct: In Android Permissions, the original performance was superior to that obtained with the balanced versions, and none of the

strategies reached the minimum operational criterion adopted in this study (F1-score $\geq$ 0.70); in Adroit, both oversampling and undersampling reduced Recall relative to the original version and produced an F1-score close to 0.1; in DefenseDroid PRS, the most critical case, Recall dropped after applying balancing, with F1-score below 0.05 in both configurations. AndroCrawl constitutes a distinct case: despite the severe original imbalance (8.5:1), balancing did not produce simultaneous improvement in Recall and F1-score, due to a specific structural limitation detailed in RQ3. The four cases are analyzed in greater depth in RQ3.

The joint analysis of Recall and F1-score, illustrated in Figure 2, shows that the effect of balancing is not uniform. In 7 of 11 datasets, at least one balancing technique maintained or improved Recall without reducing F1-score below the adopted operational criterion. In the remaining four datasets (Android Permissions, Adroit, DefenseDroid PRS, and AndroCrawl), no balanced configuration simultaneously surpassed the Recall and F1-score of the reduced version. The structural and statistical causes of these four cases are detailed in RQ3.

4.3. RQ2: Do different balancing strategies produce distinct effects on the evaluated datasets?

The results indicate distinct behaviors among undersampling, hybrid strategies, and oversampling. In the two experimental stages, no technique was superior for all datasets, reinforcing that the effect of balancing depends on the profile of each dataset. Still, some patterns can be observed from the techniques selected in Table 4.

In the case of oversampling, RandomOver, SMOTE, and ADASYN were selected in different datasets. This result indicates that expanding the minority class can be useful in different scenarios, but does not point to consistent superiority of a single method. In the second stage, RandomOver was selected in several datasets, while SMOTE remained competitive in datasets such as KronoDroid Emulator, DREBIN-215, and AndroCrawl. Thus, simple replication of the minority class may be sufficient in some cases, while synthetic sample generation may be more appropriate in others.

Among the undersampling techniques, an important change is observed between the stages. In the first stage, RandomUnder and ENN appeared as frequent alternatives. In the second stage, after the inclusion of hybrid strategies, ENN + RandomUnder was selected for most datasets. This result suggests that applying a cleaning step before majority class reduction may produce more suitable sets for the reference classifier. However, this behavior was not universal: NearMiss was selected in AndroCrawl and Adroit, indicating that some datasets respond better to a more targeted selection of majority samples.

Differences also appear in computational cost. Undersampling techniques tend to reduce the number of samples and therefore can decrease execution time. Oversampling, on the other hand, increases the training set and can increase cost, although it may still improve malicious class identification in imbalanced datasets. Thus, the choice of technique should consider not only Recall or F1-score in isolation, but also the final dataset size and processing cost.

4.4. RQ3: In which scenarios does balancing degrade performance?

Figure 3 presents the structural indicators of diversity and sparsity for the four evaluated versions.In Adroit, high sparsity (95.31%) persisted after balancing. Undersampling concentrated the benign class on very similar samples, while oversampling generated uninformative synthetic samples in an already homogeneous space. The result was a drop in Recall and F1-score to values close to 0.57 and 0.12, respectively.

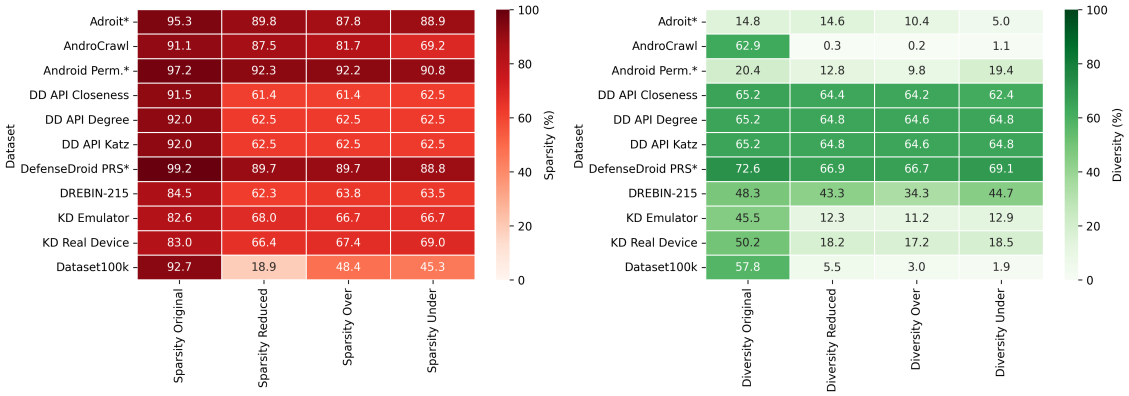


Figure 3. Structural characterization of datasets by version. High sparsity (intense red) and low diversity (light green) indicate higher risk of degradation after balancing. Datasets marked with * showed performance degradation.

As for the Android Permissions dataset, sparsity remained above 90% in all versions. Oversampling further compressed data representation, while undersampling, although preserving global diversity, also did not improve performance. In both cases, F1-score dropped from 0.74 to approximately 0.16. In comparison, DefenseDroid PRS presents a distinct profile: high diversity, but sparsity between 88% and 89% even after balancing. Recall dropped from 0.81 to approximately 0.46 and F1-score fell below 0.05.

AndroCrawl presents a distinct case: although the original imbalance is severe (8.5:1), feature reduction collapsed global diversity to less than 1%, causing both oversampling and undersampling to operate on an extremely homogeneous feature space, which may have limited the classifier's ability to establish appropriate decision boundaries. In contrast, the datasets that benefited from balancing have sparsity below 70% after feature reduction, such as the DefenseDroid API variants and MH100k. In the latter, the dominant factor was the severe class proportion imbalance, not the internal data structure.

To verify whether the degradation observed in the three direct cases (Adroit, Android Permissions, and DefenseDroid PRS) stems from poor learning in particular, or from overfitting to synthetic samples or to the reduced majority class subset, a complementary analysis was conducted with Random Forest and stratified 5-fold cross-validation. The paired Wilcoxon test ($\alpha = 0.05$) indicated that, in these three datasets, the numerical degradation of Recall did not reach statistical significance ($p > 0.05$ in all cases), suggesting that the variability across folds is high enough to prevent stating, with 95% confidence, that balancing systematically worsened performance. Table 6 presents the gap between average training and test Recall before and after balancing, along with the standard deviation of test Recall (σ).

Table 6. Overfitting diagnosis: train-test gap and stability in the three datasets with direct degradation.

Dataset	Gap Original	Gap Balanced	σ Recall Original	σ Recall Balanced
Adroit (Under)	0.0444	0.0110	0.0239	0.0132
Adroit (Over)	0.0444	0.0127	0.0239	0.0116
DD PRS (Under)	0.0697	0.0142	0.0102	0.0028
DD PRS (Over)	0.0697	0.0598	0.0102	0.0045
Android Perm. (Under)	0.0512	0.0360	0.0082	0.0101
Android Perm. (Over)	0.0512	0.0412	0.0082	0.0070

In five of the six configurations, balancing simultaneously reduced the gap and the standard deviation, indicating more stable models without classic overfitting. The exception was Android Permissions with undersampling, where the drastic reduction in gap came with increased variability across folds ($\sigma \uparrow 23\%$), suggesting that removing samples from the majority class created structurally heterogeneous training sets. This pattern does not support the hypothesis of overfitting as the main cause of degradation. If the model were memorizing synthetic samples or being mainly affected by the loss of informative examples, an increase in the train-test gap would be expected, not the reduction observed in most configurations. The greater stability (σ smaller in 5 of 6 configurations) does not necessarily translate into better performance. One possible interpretation is that the model learns more consistently on an informationally limited representation: it exhibits low gap and lower variability across folds, but remains limited by the quality of the available data.

Overall, persistently high sparsity after balancing is an indicator of degradation risk. Combined with the absence of evidence of overfitting and non-significant p-values, this analysis supports the interpretation that, in these cases, balancing does not introduce learning artifacts but amplifies pre-existing structural limitations. These indicators should, however, be treated as contextual signals rather than deterministic rules; the decision to balance should always be empirically validated for each dataset.

4.5. RQ4: Do diversity and sparsity help explain the effect of balancing?

The analysis of RQ3 indicates that persistent sparsity and collapsed diversity are useful indicators for interpreting the observed degradation cases, but they are not sufficient in isolation. MH100k combines low diversity after feature reduction with a strong Recall gain, because the dominant factor in that case is the class proportion (9.4:1), not the internal data structure. Conversely, DefenseDroid PRS was already nearly balanced and still showed degradation, indicating that numerical balancing can alter the decision boundary without correcting any real bias.

Diversity and sparsity can be used as structural indicators complementary to the class proportion, helping to identify scenarios where balancing tends to benefit or degrade learning. These results indicate that the application of balancing techniques should be evaluated individually for each dataset, rather than adopted automatically in the pipeline.

4.6. RQ5: What is the impact on execution time?

Figure 4 presents the execution time of datasets in their original, reduced, undersampled, and oversampled versions. The logarithmic scale was used due to the large difference between observed times, especially in MH100k.

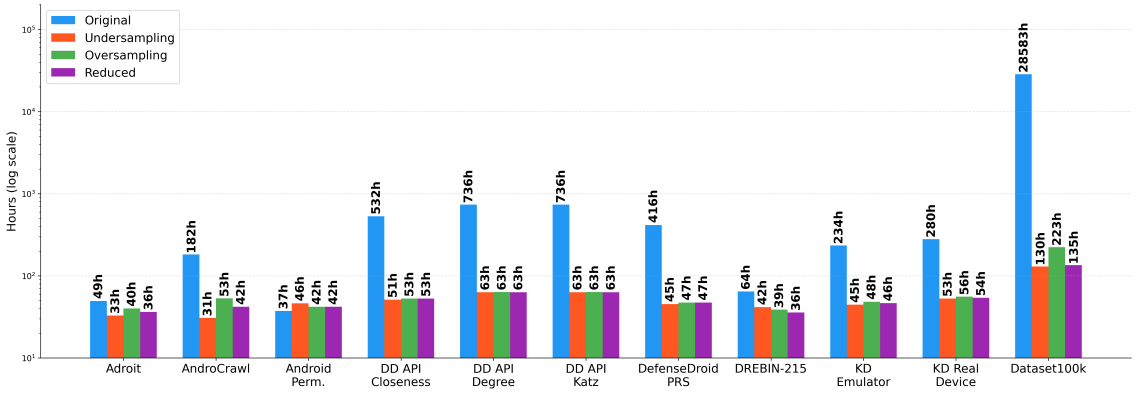


Figure 4. Execution time of datasets in original, reduced, and balanced versions. The vertical axis is on a logarithmic scale.

Considering the datasets in which the balanced versions preserved or improved Recall and F1-score relative to the reduced version, an average reduction of approximately 70% in execution time was observed compared to the original versions. The reduction was particularly pronounced in MH100k, due to the combination of high original dimensionality and strong class imbalance. In this dataset, the reference time of the original version was 28,583 hours, while the reduced version subjected to undersampling presented a time of approximately 130 hours, corresponding to a reduction of over 99%. This result indicates that, in large-scale datasets, dimensionality reduction combined with sample removal techniques can make feasible processing that, in the original configuration, would have impractical computational cost.

In AndroCrawl, the time of the original version was over 180 hours, while the reduced and balanced versions were on the order of tens of hours. A similar trend occurs in the DefenseDroid variants, where original times exceeded hundreds of hours and were reduced to approximately 50–65 hours after transformations.

The comparison between undersampling and oversampling reveals distinct behaviors. In general, undersampling tends to reduce execution time more, as it decreases the number of samples used in training. Oversampling, on the other hand, can increase the number of samples and therefore have a higher cost than undersampling. Still, when applied to previously reduced datasets, oversampling remained substantially more efficient than processing the original versions in higher-dimensionality datasets. These results reinforce that balancing should also be analyzed from a computational cost perspective. In scenarios where different techniques present similar predictive performance, execution time can become a decisive criterion for choosing the most appropriate strategy.

4.7. Threats to Validity

This section discusses the main threats to the validity of the presented results.

Internal validity. Balancing techniques were applied before the fold partition, which affects oversampling and undersampling differently, as described in Section 3.5. Therefore, this study bases its conclusions on undersampling and hybrid strategies, and treats oversampling only as an exploratory comparison reference. Replicating the oversampling experiments under a strictly intra-fold protocol to confirm the exploratory findings reported is left as future work.

External validity. The 11 evaluated datasets cover a good variety of size, dimensionality, and imbalance level, but all are Android malware. Thus, the effect of temporal concept drift was not evaluated, and the generalization of the findings to other operating systems or other tabular classification domains remains open.

Conclusion validity. The paired Wilcoxon test was applied with $n = 5$ folds, which limits the statistical power of the analysis. The significance results, therefore, should be read as indicative rather than definitive.

Construct validity. Recall and F1-score capture specific aspects of performance. An analysis based on curve (AUC-PR) could complement this view and is left as future work.

5. Conclusion

Class balancing is not a neutral preprocessing step. Across 11 tabular Android malware datasets, it improved detection in 7 and degraded it in 4, a discrepancy resolved not by technique selection, but by understanding the structural profile of each dataset: sparsity and diversity indicate the outcome. Undersampling strategies, hybrid strategies, and oversampling were compared, adopting Recall as the primary metric, F1-score as a complementary metric, and execution time as an indicator of computational cost.

The results show that balancing can promote gains in heavily imbalanced datasets, in addition to reducing computational cost. In MH100k, dimensionality reduction improved performance relative to the original version, and applying the hybrid strategy ENN + RandomUnder on the reduced version resulted in the best values observed for this dataset, with Recall of 0.99 and F1-score of 0.97. This result suggests that, in this scenario, balancing was more effective when combined with a lower-dimensionality representation of the data.

On the other hand, balancing degraded performance in some datasets (Adroit, Android Permissions, DefenseDroid PRS, AndroCrawl), with no simultaneous gains in Recall and F1-score. The structural analysis points to low diversity, high sparsity, and sensitive distributions as limiting factors. The main conclusion is that balancing should not be automatic, but rather an experimental decision guided by the data. Among the techniques, hybrid undersampling (especially ENN + RandomUnder) was the most consistent when imbalance was the main limiting factor.

As future work, we highlight the development of automatic screening criteria based on structural dataset indicators, complementary evaluation with curve-based metrics such as AUC-PR, direct comparison with cost-sensitive approaches, and the replication of oversampling experiments under a strictly intra-fold balancing protocol to confirm the reported exploratory findings.

Overall, these findings reinforce that class balancing in Android malware detection should be treated as a dataset-dependent experimental decision, evaluated jointly in terms of predictive performance, computational cost, class distribution, and structural characteristics, rather than as a default preprocessing step.

Acknowledgments

This work was partially supported by Capes, CNPq (Grant No. 409743/2025-9), FAPERGS (Grant Nos. 24/2551-0001368-7, 24/2551-0000726-1, and 25/2551-0002572-9), and FAPESP (Grant Nos. 2020/05183-0 and 2023/00816-2). The authors also acknowledge the use of Generative AI as a supporting tool for reviewing and improving the code and manuscript.

References

Akintola, A. G., Balogun, A., Mojeed, H. A., Usman-Hamza, F., Salihu, S. A., Adewole, K. S., Balogun, G. B., and Sadiku, P. O. (2022). Performance analysis of machine learning methods with class imbalance problem in android malware detection. *IJIM*, 16(10).

Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., and Rieck, K. (2022). Dos and don'ts of machine learning in computer security. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3971–3988.

Blagus, R. and Lusa, L. (2013). SMOTE for high-dimensional class-imbalanced data. *BMC Bioinformatics*, 14(106).

Bragança, H., Rocha, V., Barcellos, L. V., Souto, E., Kreutz, D., and Feitosa, E. (2023). Android malware detection with mh-100k: An innovative dataset for advanced research. *Data in Brief*, 51:109750.

Chen, W., Yang, K., Yu, Z., Shi, Y., and Chen, C. L. P. (2024). A survey on imbalanced learning: Latest research, applications and future directions. *Artificial Intelligence Review*, 57(137).

Colaco, C., Bagwe, M., Bose, S., and Jain, K. (2021). Defensedroid: A modern approach to android malware detection. *Strad Research*, 8(5).

Das, S., Mullick, S. S., and Zelinka, I. (2022). On supervised class-imbalanced learning: An updated perspective and some key challenges. *IEEE Transactions on Artificial Intelligence*, 3(6).

Demircioğlu, A. (2024). Applying oversampling before cross-validation will lead to high bias in radiomics. *Scientific Reports*, 14(1).

Ge, X., Huang, Y., Hui, Z., Wang, X., and Cao, X. (2021). Impact of datasets on machine learning based methods in android malware detection: an empirical study. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. IEEE.

Gomes, A. L., Ferreira, L., Nogueira, A. G. D., Kreutz, D., Schmidt, D., Mansilha, R. B., and Paim, K. O. (2026). Statistical ranking: A voting-based ensemble approach to feature selection in android malware detection. Manuscript submitted for review.

Guan, J., Jiang, X., and Mao, B. (2021). A method for class-imbalance learning in android malware detection. *Electronics*, 10(24).

Haluška, R., Brabec, J., and Komárek, T. (2022). Benchmark of data preprocessing methods for imbalanced classification. In *2022 IEEE International Conference on Big Data*.

He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

John, R. and Di Troia, F. (2025). Comparing balancing techniques for malware classification. In *Machine Learning, Deep Learning and AI for Cybersecurity*. Springer.

Rocha, V., Bragança, H., Kreutz, D., and Feitosa, E. (2024). Mh-fsf: um framework para reprodução, experimentação e avaliação de métodos de seleção de características. In *Anais Estendidos do XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. SBC.

Sisto, A. (2013). Androcrawl: Studying alternative android marketplaces. Master's thesis, Politecnico di Milano.

Sun, N., Zhang, J., Rimba, P., Gao, S., Zhang, L. Y., and Xiang, Y. (2018). Data-driven cybersecurity incident prediction: A survey. *IEEE communications surveys & tutorials*, 21(2).

Vairetti, C., Assadi, J. L., and Maldonado, S. (2024). Efficient hybrid oversampling and intelligent undersampling for imbalanced big data classification. *Expert Systems with Applications*, 246:123149.

Wongvorachan, T., He, S., and Bulut, O. (2023). A comparison of undersampling, oversampling, and smote methods for dealing with imbalanced classification in educational data mining. *Information*, 14(1).