Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Few Informative Data Samples are Good Enough: Introducing Intelligent Data Pruning

RISHI SINGHAL[1], ADITYA PAI BRAHMAVAR[2], AND DEEPAK RAJENDRAN.[3]
[1]Department of Computer Science, North Carolina State University, (e-mail: rsingha4@ncsu.edu)
[2]Department of Computer Science, North Carolina State University, (e-mail: abrahma@ncsu.edu)
[3]Department of Computer Science, North Carolina State University, (e-mail: drajend2@ncsu.edu)

Corresponding author: Rishi Singhal (e-mail: rsingha4@ncsu.edu).

ABSTRACT Data quality is a critical metric in ensuring that machine learning models perform well on real-world tasks. More specifically, in the case of class-imbalanced datasets, sampling strategies play an important role in deciding the quality of data fed to machine learning models. Our study introduces a novel approach to data under-sampling called 'intelligent data pruning,' which employs advanced clustering techniques to identify and eliminate redundant or irrelevant data points. By significantly reducing dataset size without loss of 'informative' data samples, our method optimizes the model training process for low computational resource utilization while enhancing model generalization. We extensively compare our method with existing under-sampling and over-sampling techniques on various software engineering datasets using three different learners: Logistic Regression, Decision Tree, and Support Vector Machine. Traditional data sampling methods like random sampling, SMOTE as well as sophisticated techniques such as Gaussian Copula and Recursive Random Projection based oversampling techniques are included in our evaluation as benchmarks. Through rigorous experimentation and performance analysis, we demonstrate the superior accuracy of our proposed data pruning method across diverse datasets and learners. Additionally, we provide insights into the underlying mechanisms driving its effectiveness and its potential applications in real-world scenarios with large-scale datasets and computational constraints.

INDEX TERMS Class Imbalance, Oversampling, Undersampling, Intelligent Data Pruning, Redundant Samples, Informative Samples, Classification, Scalability, Statistical Analysis

I. INTRODUCTION

IN current times where 'data-hungry' machine learning/artificial intelligence (ML/AI) algorithms are ubiquitously used to solve problems across diverse fields, there is always a requirement of not only large amounts of data but also good quality data [1]. This comes with challenges like inconsistencies, incompleteness and ambiguity in the data captured from the real world, due to which data preprocessing is a critical task in all ML/AI pipelines to ensure good performance of downstream models. Moreover, data preprocessing is a context-specific task, i.e., treatment of raw data varies with nature (and quality) of data, domain knowledge, availability of computational resources, stakeholder requirements etc. [2] In our research, we explore the data preprocessing task in a scenario where there is 1) class-imbalanced dataset [3] 2) a constraint in resources (storage and compute) [4]. When there is class imbalance during training, the model can become biased towards the majority class, hence hurting the performance of the minority classes. To tackle this issue, various over-sampling/under-sampling techniques have been recommended [5]. On the other hand, machine learning & deep learning algorithms require an ample amount of data to achieve good performance. Hence, to handle this issue, various few-shot learning [11]. [12] and semi-supervised learning algorithms [13], [14] have been developed.

In our study, we support an intuitive statement - "A small amount of informative data is as good as large amount of data with redundant information, if not better". To this end, we propose and evaluate a data under-sampling method called 'intelligent data pruning' which removes redundant samples within the majority class (of an imbalanced dataset) in an

’intelligent’ way. As described in the Algorithm 3, intelligent pruning first clusters the majority class into multiple sub-clusters. It then removes more data samples from the bigger sub-clusters as they have more redundant samples in comparison to smaller sub-clusters, which are more unique. This is then compared to existing over-sampling/under-sampling techniques such as (a) Synthetic Data Vault [6], (b) Recursive Random Projection Oversampling [7], (c) SMOTE & its variants [8]. [9], (d) Random undersampling & oversampling [10].

To structure this inquiry, we ask these questions:

- RQ1: When considering the data imbalance problem, which under-sampling/over-sampling technique performs the best by considering performance metrics of precision, recall, f1 score & accuracy?
- RQ2: Which sampling technique is statistically more significant while considering model performance, using average wins (Avg Wins) metric?
- RQ3: Which under-sampling(pruning) strategy is better - randomly pruning or intelligently pruning?
- RQ4: Which sampling technique is more scalable in cases of resource-constrained environment(insufficient data storage) and limited compute time?
- RQ5: What suggestions can we provide from analyzing the conclusions of RQ1 to RQ4?

The contributions of this paper are as follows:

- We proposed Intelligent Data Pruning, an under-sampling technique for balancing datasets.
- We perform extensive empirical/experimental analysis by comparing our proposed pruning technique against the rest of the existing sampling techniques in different aspects such as (a) model performance metrics - accuracy, precision, recall, , F1-Score, (b) Average Wins based on the Skott-Knott test, (b) performance comparison in resource-constrained environments.
- We find that Intelligent Data Pruning performs at par and sometimes better than existing techniques, while achieving fairly significant average wins.
- The Intelligent Data Pruning technique, in the case of resource-constrained environments (limited data storage space and compute time), is efficient in comparison to the rest of the techniques included in this study.

The rest of this paper is constructed as follows: Section II illustrates the background of sampling data (over-sampling & under-sampling) and an extensive literature review on existing data sampling techniques. Section III presents different sampling techniques considered in this study including our proposed Intelligent Data Pruning. It also consists of different downstream learners/machine learning models that we use to evaluate the performance of the sampling techniques. Section IV illustrates various benchmark datasets, hyperparameters tuned, evaluation metrics, and the statistical analysis test used in this study. Section V presents our experimental findings and a thorough discussion of the same. We also discuss the threats to validity of our experiments in Section VI, and finally make a conclusion to this study in Section VII.

II. BACKGROUND

A. SAMPLING DATA

Data sampling techniques either over-sample or under-sample the data to achieve better, informative, and balanced data. Over-sampling is usually done by generating synthetic data using a generative model which learns patterns from the original dataset. Under-sampling methods also exist, which simply prune the data for balancing the class distribution of the dataset. More specifically, the sampling process can be described as follows:

Given a dataset D: [(x_1, y_1), (x_2, y_2),...., (x_n, y_n)] with features f_1, f_2,....,f_k, having class imbalance in terms of its label distribution (minority class and majority class), we either over-sample the minority class using generative model M, or we under-sample/prune the majority class (efficiently) to achieve a balanced dataset D’. The newly created balanced dataset D’ is supposed to result in better model performance of the downstream ML model, which uses D’ in training.

Researchers extensively focus on over-sampling techniques in improving the model’s performance (quantified by accuracy, precision, recall, and F1-Score) [5] while trying to optimize for resource-efficiency [4]. However, as we discuss later in this paper, an intelligent way of data pruning not only achieves better model performance but is also much more resource-efficient in terms of data storage capacity and compute time. We also provide a statistical distribution analysis to give a broader analysis of different sampling techniques in comparison to our proposed method.

B. RELATED WORK

Active learning offers a promising approach for imbalanced datasets as explored in [15], where Support Vector Machines (SVM) is used to iteratively train on a small, labeled subset of a dataset. The model then strategically selects the most uncertain or informative samples from the remaining unlabeled data for labeling. This targeted approach achieves high performance with fewer labeled instances compared to traditional supervised learning, which utilize the entire labeled data at once.

Under-sampling techniques provide another avenue for handling imbalanced data. Works like BalanceCascade and EasyEnsemble in [28] leverage different strategies: the former eliminates easily learnable data points, while latter combines predictions from models trained on multiple, strategically chosen subsets of the majority class. The effectiveness of clustering-based under-sampling is further supported by research in [29], [30], [31], [32]. These studies propose successful algorithms tailored to various problem domains, highlighting the positive trend of this approach towards data under-samping.

Intelligent over-sampling techniques to create synthetic data using Gaussian disturbance (e.g., [36]) and SMOTE with

SVM (e.g., [37]) exist, in addition to genetic algorithms like GenSample [35], but the potential for introducing errors, as argued in [35], necessitates exploring alternatives. Building on the guidelines of [27] regarding representativeness and randomness in software engineering research, where we have a "generalizability crisis" due to a lack of sophisticated sampling strategies, we focus on under-sampling the majority class to identify the most informative samples sufficient for achieving good performance in downstream learners.

III. METHODOLOGY

A. SAMPLING TECHNIQUES

In this section, we illustrate different undersampling and oversampling baseline techniques considered in our study, along with the proposed Intelligent Data Pruning Technique.

1) No Sampling

The very first baseline that we consider is no-sampling, where we neither do oversampling nor undersampling and use the original data points as it is for training the models.

2) Random Pruning

For an intuitive baseline for undersampling, we randomly prune from the majority class at different pruning ratios until we get the same number of samples as in the minority class.

3) Random Oversampling

Similarly, we establish an intuitive baseline for oversampling, by randomly increasing the number of samples of the minority class at different oversampling ratios till we get the same number of samples as in the majority class [16]. We have made use of the inbuilt Scikit-Learn Random-Oversampling function for implementation.

4) Synthetic Minority Over-sampling Technique (SMOTE) Oversampling

In our study, we make use SMOTE [8] to oversample minority classes at different ratios. SMOTE randomly selects a minority class instance ’a’ and finds its ’k’ nearest minority class neighbors. It then creates a synthetic instance by randomly selecting one of these neighbors ’b’ and connecting ’a’ and ’b’ to form a line segment in the feature space, generating synthetic instances as a convex combination of ’a’ and ’b’. The SMOTE process is explained further in Algorithm 1. We have used the Scikit-Learn library inbuilt SMOTE function as part of this study.

5) SVM-SMOTE Oversampling

Apart from SMOTE, we utilize another oversampling technique of SVM-SMOTE [9], an advanced version of SMOTE that leverages Support Vector Machines (SVMs) to improve the selection of synthetic samples by using Support Vectors. Using this we oversampled the minority class at various sampling ratios. We have used the Scikit-Learn library inbuilt SVM-SMOTE function for implementation purpose.

Algorithm 1 SMOTE Oversampling Algorithm

```
Require: Minority class dataset $D_{\text{min}}$, Number of synthetic
    samples $N_{\text{syn}}$, Number of nearest neighbors $k$
1: Initialize synthetic dataset $D_{\text{syn}}$ as empty
2: while $|D_{\text{syn}}| < N_{\text{syn}}$ do
3:     Randomly select a minority sample $x_{\text{min}}$ from $D_{\text{min}}$
4:     Find $k$ nearest neighbors of $x_{\text{min}}$ in $D_{\text{min}}$, excluding
       itself
5:     Randomly select one of the $k$ neighbors, $x_{\text{nn}}$
6:     Generate a synthetic sample, $x_{\text{new}}$, as a convex combi-
       nation of $x_{\text{min}}$ and $x_{\text{nn}}$
7:     Add $x_{\text{new}}$ to $D_{\text{syn}}$
8: end while
9: return $D_{\text{syn}}$
```

6) SDV-G Oversampling

As another baseline, the Synthetic Data Vault-Guassian Coupula (SDV-G) library [6] is made part our study. SDV-G generates synthetic data by modeling relationships between tables and columns in a relational database. It uses Conditional Parameter Aggregation for table relationships and Gaussian Copula for column relationships, supporting model-based and knowledge-based synthesis. Hence, making it a versatile framework for generating synthetic data across datasets. We use the SDV public API for its implementation in our study. We also provide a pseudo code for the SDV-G pipeline in Algorithm 2.

Algorithm 2 SDV-G Oversampling Algorithm

```
Require: Original dataset $D$, Desired number of synthetic
    samples $N$, Correlation matrix $\Sigma$
1: Initialize empty synthetic dataset $D_{\text{synthetic}}$
2: for each attribute $A$ in $D$ do
3:     Fit a Gaussian copula to the data in $A$ to estimate the
       marginal distribution and correlation
4:     Generate $N$ samples from the fitted copula to get
       synthetic values for $A$
5:     Append the synthetic values to $D_{\text{synthetic}}$
6: end for
7: return $D_{\text{synthetic}}$
```

7) RRP Oversampling

We establish another baseline using Recursive Random Projection(RRP) based oversampling. RRP oversampling introduced in [7], is designed to tackle class imbalance in datasets. It operates in two main stages: first, data is clustered to group similar instances together, allowing for the creation of synthetic samples within these clusters. Second, within each cluster, mutation and crossover operators are applied to data points. Mutation introduces random perturbations to data points, while crossover combines attributes from different points to create new synthetic samples, hence helping to generate diverse synthetic samples that closely resemble the minority class. We provide the intuitive algorithm for RRP as Algorithm 3.

Algorithm 3 RRP Oversampling Algorithm

```
Require: Original dataset $D$, Desired number of synthetic samples $N$
 1: Initialize empty synthetic dataset $D_{\text{synthetic}}$
 2: Split the data into different clusters using FASTMAP.
 3: Mutation & crossover operators which mutate the data points in the same cluster to generate synthetic data.
 4: return $D_{\text{synthetic}}$
```

8) Intelligent Data Pruning

We propose our undersampling technique - 'Intelligent Data Pruning' to balance imbalanced datasets by selectively removing samples from the majority class. Instead of random selection, it involves clustering the majority class samples and pruning more from the largest cluster. This approach allows for more targeted pruning, with control over both overall pruning ratios (percentage of majority class to prune) and per-cluster pruning ratios (percentage of samples to prune from a cluster in decreasing order of cluster size). The goal is to prune the majority class samples while ensuring that the number of remaining majority class samples matches that of the minority class, thus improving class balance for better model performance. We also provide a pseudocode for the proposed "Intelligent Data Pruning" technique for a better understanding as part of Algorithm 4.

Algorithm 4 Our Proposed - Intelligent Data Pruning Algorithm

```
Require: Majority class dataset $M$, Number of Samples to prune size $S$, Per-Cluster Pruning Ratio $r_c$
 1: Initialize cluster list $C$
 2: Perform clustering using K-Means on $M$ to fill $C$
 3: Compute the size of each cluster in $C$ and sort $C$ in decreasing order of size
 4: for each cluster $c$ in $C$ do
 5:     Calculate the number of samples to prune $n_c$ from $c$: $n_c = |c| \times r_c$
 6:     Randomly select $n_c$ samples from $c$ for pruning
 7:     Remove the selected samples from $c$
 8:     Subtract $n_c$ from $S$: $S$ = $S$ - $n_c$
 9: end for
10: Recompute the size of each cluster in $C$ and sort $C$ in decreasing order of size
11: while $S$ >0 do
12:     for each cluster $c$ in $C$ do
13:         Calculate the number of samples to prune $n_c$ from $c$: $n_c = |c| \times r_c$
14:         Prune $n_c$ samples from $c$
15:         Subtract $n_c$ from $S$: $S$ = $S$ - $n_c$
16:     end for
17: end while
```

B. MODELS

1) Logistic Regression

Logistic regression [17] is a type of supervised learning algorithm used for classification tasks. Despite its name, logistic regression is used for binary classification, where the output variable is categorical and has two classes. It models the probability that an instance belongs to a particular class using the logistic function, which maps any real-valued input into a value between 0 and 1. Logistic regression estimates the parameters of the logistic function by minimizing a loss function, typically the logistic loss or cross-entropy loss. It is interpretable, computationally efficient, and works well with linearly separable data.

$$P(y=1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\ldots+\beta_n x_n)}}$$

where

$$P(y=1|x)$$

is the probability that the target variable y equals 1 given the input features x.

$$\beta_0, \beta_1, ..., \beta_n$$

are coefficients of logistic regression model and

$$x_1, x_2, ..., x_n$$

are input features

2) Decision Tree

Decision trees [18] are a versatile and widely used supervised learning method for both classification and regression tasks. They partition the feature space into a set of regions, each associated with a particular class or value. Decision trees recursively split the feature space based on the feature that provides the best separation of the classes or minimizes the variance within the resulting regions. The splits s are chosen to maximize information gain or minimize impurity measures such as Gini impurity or entropy.

$$X_j \leq s \rightarrow \text{left child node}$$

$$X_j > s \rightarrow \text{right child node}$$

Decision trees are easy to interpret and visualize, handle both numerical and categorical data, and can capture complex relationships between features and target variables.

3) Support Vector Machine

Support vector machines (SVMs) [19] are powerful supervised learning algorithms used for classification and regression tasks. SVMs find the optimal hyperplane that separates instances of different classes in the feature space while maximizing the margin between the classes. SVMs can handle linearly separable as well as non-linearly separable data by using different kernel functions, such as linear, polynomial, radial basis function (RBF), or sigmoid kernels. SVMs are effective in high-dimensional spaces, robust to overfitting, and can capture complex decision boundaries. However, SVMs can be computationally expensive, especially for large datasets, and may require careful selection of hyperparameters. Mathematically, the decision boundary is represented as:

$$w^T x + b = 0$$

where w is the weight vector. x is the input feature vector. b is the bias term.

The goal is to find the weight vector w and bias term b that maximize the margin between the support vectors (instances closest to the decision boundary) while satisfying the constraint

$$y_i(w^T x_i + b) \geq 1$$

for all training instances. As an optimization problem, this cann represented as:

$$\min_{w,b} \frac{1}{2}||w||^2$$

IV. EXPERIMENTAL SETUP

In this section, we will illustrate the following things: (1) Benchmark Datasets, (2) Hyperparameters Used, (3) Evaluation Metrics, and (4) the Statistical Analysis Procedure(Scott-Knott Test). All of our experiment scripts are written in Python using different modules such as - Numpy, Pandas, Scikit-Learn, and others. Our code is publicly available at https://github.com/adipai/data-decent.

A. BENCHMARK DATASETS

In this study, we have considered 8 Software Engineering based binary classification-based datasets: the Breast Cancer dataset, and the Churn dataset, which focuses on customer churn prediction in a telecommunications setting from the PMLB library [21]. Additionally, we used the JS_Vuln dataset [39], which deals with vulnerabilities in JavaScript projects, and the Ambari_Vuln dataset [39], which focuses on vulnerabilities in Apache Ambari. Furthermore, we included the Defect_Eclipse_JDT_Core dataset [20], which relates to defects in the Eclipse JDT Core project, and the Defect_Eclipse_PDE_UI dataset [20], which pertains to defects in the Eclipse PDE UI project. Lastly, we utilized the Moodle_Vuln [39] dataset, which involves vulnerabilities in the Moodle e-learning platform, and the Defect_Mylyn dataset [20], which concerns defects in the Mylyn project. These datasets were chosen to cover a range of domains and problem types in Software Engineering, providing a comprehensive evaluation of the proposed sampling methodologies. After extracting the datasets, we follow a common pre-processing pipeline of (1) removing duplicate rows, (2) replacing nan values with mean/mode, and (3) normalizing the columns using StandardScaler(). After doing all the data extraction & pre-processing steps, we summarize all the datasets in Table 1.

B. HYPERPARAMETERS USED

Benchmark	#Rows	#Cols	Class-Distribution %age
breast_cancer	286	10	70:30
churn	5000	21	85:15
JS_Vuln	6271	36	85:15
Ambari_Vuln	1000	102	97:3
Defect_Eclipse_JDT	997	82	80:20
Defect_Eclipse_PDE	1497	82	86:14
Moodle_Vuln	1861	82	98:2
Defect_Myln	2056	14	86:14

TABLE 1. Summary of Benchmark datasets used in our study

Benchmark	Optimal Number of K-Means Clusters
breast_cancer	3
churn	3
JS_Vuln	2
Ambari_Vuln	3
Defect_Eclipse_JDT	4
Defect_Eclipse_PDE	4
Moodle_Vuln	3
Defect_Myln	3

TABLE 2. Optimal number of K-Means for all 8 datasets using the ELBO-Curve Method in case of Intelligent Pruning

1) Sampling-Percent

As part of this study, we played with various hyperparameters for all the sampling techniques. We varied hyperparameters of *Sampling-Percent* of 20%, 40%, 60%, 80% & 100% (undersampling ratios for pruning algorithms and oversampling ratios for oversampling algorithms). *Sampling-Percent* refers to how much we want to oversample/undersample the minority/majority class respectively until we achieve a completely balanced training set, i.e., 100% sampling.

2) Per-Cluster-Sampling-Percent & K-Means Clusters

Furthermore, we introduced another hyperparameter called *Per-Cluster-Sampling-Percent* for our proposed Intelligent Data Pruning technique, which we varied as 50%, 70% & 100%. *Per-Cluster-Sampling-Percent* means how much we want to prune from the clusters of the majority class in descending order of cluster size. The reason for playing out with different percentages is that we don't want to completely prune all the samples from the largest sub-cluster and prune from other clusters as well but in a lesser amount as they are lesser in number and more unique/informative.
Apart from the Per-Cluster-Sampling-Percent, we used K-Means [34] as the clustering algorithm for the majority class while doing Intelligent Pruning, due to its simplicity, adaptability & scalability to large datasets. Also, to tune the number of clusters to define for a particular dataset we adopted the ELBO-Curve method [33] which facilitates choosing the optimal number of clusters to form for the given dataset using K-Means. The number of clusters formed for all 8 datasets using K-Means are summarized in Table 2.

3) Other Miscellaneous Hyperparameters

Other hyperparameters we used are - the train-test size, random seeds & default parameters of models (Logistic Regression, Decision Tree & SVM) in the Scikit-Learn library. For the train-test size, we use the widely used standard 8:2 ratio. For random seeds we used 10 different random seeds to reduce stochasticity of different sampling techniques and models. We provide all our Skott-Knott plots for all 8 benchmarks across all metrics in our github repository readme. Lastly, we stuck to the default parameters of the Logistic Regression, Decision Tree & SVM models as part of the Scikit-Learn library to report our results.

C. EVALUATION METRICS

1) Accuracy

Accuracy measures the overall correctness of the model's predictions. It is calculated as the ratio of the number of correct predictions to the total number of predictions made. Mathematically, accuracy is represented as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

While accuracy is a useful metric, it can be misleading in cases of imbalanced datasets where one class dominates the others. In such cases, high accuracy can be achieved by simply predicting the dominant class for all instances.

2) Precision

Precision measures the accuracy of positive predictions made by the model. It is calculated as the ratio of true positive predictions to the total number of positive predictions made by the model, represented as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Precision is especially important in scenarios where the cost of false positives is high.

3) Recall

Recall measures the ability of a model to correctly identify positive instances. It is calculated as the ratio of true positive predictions to the total number of actual positive instances in the dataset and represented as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Recall is particularly important in scenarios where missing positive instances can have severe consequences, such as in medical diagnostics or anomaly detection.

4) F1 score

The F1 score is a harmonic mean of precision and recall, providing a balance between the two metrics and calculated as:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score ranges from 0 to 1, where higher values indicate better model performance. It is particularly useful in scenarios where both precision and recall are important, such as when the cost of false positives and false negatives are similar.

5) Avg Wins

We introduce another performance metric for different sampling techniques, called *Avg Wins*. "Avg Wins" refers to the average number of times a sampling technique comes in Rank 0 of the Skott-Knott plots across all 8 datasets, irrespective of the hyperparameters (Sampling-Percent & Per-Cluster-Sampling-Percent). *Avg Wins* for a sampling technique is represented as (# means number of times a Sampling Technique occurs):

$$\text{Avg Wins} = \frac{\text{\# Sampling Technique in Rank 0 for all datasets}}{\text{Total Number of datasets}}$$

Avg Wins gives a broader picture of the occurrence of a sampling technique in the best Rank 0, i.e., whether or not the sampling technique is reliable.

D. SCOTT-KNOTT ANALYSIS

The Scott-Knott analysis [22], [23] is a popular non-parametric statistical method in empirical software engineering to compare multiple treatments/groups in experimental studies. It provides a way to rank treatments based on observed outcomes while simultaneously identifying statistically significant differences between them. In our case, we compare the different sampling techniques on their performance metrics obtained on the datasets for 10 random seeds since they are stochastic. The ranked list C of sampling methods is then partitioned into subgroups C_1 & C_2 using a statistical test to determine if the differences between adjacently ranked sampling methods are statistically significant based on the expected mean before and after the split, while maximizing the expected mean of the delta [25], [26]. The expected mean of the delta is calculated as follows:

$$E(\Delta) = \frac{l(C_1) \cdot |\mu(C_1) - \mu(C)| + l(C_2) \cdot |\mu(C_2) - \mu(C)|}{|l(C)|}$$

where l is the length function to count the length of each list.

After the split is done, Scott-Knott utilizes the Cliff's Delta procedure [24] to check if the two subgroups differ significantly by

$$\Delta = \frac{1}{|C_1| \cdot |C_2|} \sum_{x \in C_1} \sum_{y \in C_2} \begin{cases} +1, & \text{if } x > y \\ -1, & \text{if } x < y \\ 0, & \text{if } x = y \end{cases}$$

Finally, within each subgroup, the sampling methods are considered statistically homogeneous, meaning that the observed differences between treatments within the same subgroup are not statistically significant.

Sampling Percent	Algorithm	Benchmarks								Avg Wins
		breast_ cancer	churn	JS_ Vuln	Ambari_ Vuln	Defect_ Eclipse_JDT	Defect_ Eclipse_PDE	Moodle_ Vuln	Defect_ Mylyn	
20%	random_oversampling	0.38	0.74	0.54	0.00	0.53	0.31	0.00	0.33	0.25
	smote	0.43	0.72	0.55	0.00	0.53	0.31	0.00	0.32	0.38
	svm_smote	0.38	0.72	0.55	0.00	0.53	0.25	0.00	0.35	0.25
	gaussian_copula	0.38	0.69	0.53	0.00	0.48	0.14	0.00	0.30	0.13
	RRP	0.42	0.72	.0.52	0.00	0.55	0.30	0.00	0.36	0.38
	random_pruning	0.36	0.70	0.53	0.00	0.48	0.09	0.00	0.34	0.13
	intelligent_pruning	0.36	0.71	0.53	0.00	0.49	0.13	0.00	0.33	0.50
40%	random_oversampling	0.37	0.74	0.56	0.00	0.58	0.37	0.00	0.32	0.38
	smote	0.42	0.70	0.57	0.00	0.57	0.34	0.00	0.33	0.75
	svm_smote	0.42	0.71	0.57	0.00	0.57	0.35	0.00	0.34	0.38
	gaussian_copula	0.40	0.70	0.53	0.00	0.49	0.15	0.00	0.30	0.13
	RRP	0.43	0.71	0.54	0.00	0.56	0.35	0.00	0.30	0.25
	random_pruning	0.40	0.72	0.53	0.00	0.55	0.16	0.00	0.33	0.38
	intelligent_pruning	0.38	0.73	0.53	0.00	0.58	0.18	0.00	0.36	1.25
60%	random_oversampling	0.38	0.72	0.59	0.00	0.58	0.40	0.06	0.37	0.88
	smote	0.43	0.70	0.59	0.00	0.58	0.34	0.04	0.38	0.88
	svm_smote	0.40	0.71	0.59	0.00	0.58	0.37	0.00	0.31	0.38
	gaussian_copula	0.41	0.67	0.53	0.00	0.50	0.18	0.00	0.30	0.13
	RRP	0.43	0.71	0.53	0.00	0.58	0.41	0.05	0.36	0.63
	random_pruning	0.41	0.72	0.54	0.00	0.58	0.27	0.00	0.35	0.38
	intelligent_pruning	0.43	0.72	0.54	0.00	0.61	0.30	0.00	0.36	1.13
80%	random_oversampling	0.41	0.71	0.58	0.00	0.55	0.39	0.05	0.37	0.63
	smote	0.41	0.69	0.58	0.00	0.59	0.37	0.04	0.38	0.75
	svm_smote	0.41	0.70	0.58	0.00	0.59	0.37	0.00	0.36	0.38
	gaussian_copula	0.40	0.63	0.53	0.00	0.49	0.18	0.00	0.32	0.13
	RRP	0.41	0.68	0.55	0.00	0.56	0.38	0.05	0.40	0.50
	random_pruning	0.46	0.71	0.56	0.00	0.58	0.38	0.00	0.37	0.38
	intelligent_pruning	0.43	0.72	0.56	0.00	0.59	0.38	0.00	0.37	1.50
100%	random_oversampling	0.44	0.69	0.56	0.00	0.52	0.40	0.04	0.37	0.38
	smote	0.42	0.67	0.55	0.00	0.57	0.39	0.06	0.38	0.88
	svm_smote	0.41	0.70	0.55	0.00	0.57	0.39	0.00	0.36	0.50
	gaussian_copula	0.40	0.61	0.53	0.00	0.49	0.18	0.00	0.30	0.13
	RRP	0.46	0.65	0.53	0.00	0.55	0.04	0.04	0.38	0.38
	random_pruning	0.43	0.61	0.52	0.08	0.53	0.39	0.05	0.31	0.50
	intelligent_pruning	0.47	0.72	0.57	0.00	0.57	0.39	0.05	0.34	1.50

TABLE 3. Model comparison scores for different sampling ratios for SVM(Results for DT & LR are in Appendix B). All the tasks are evaluated by F1-Score using different sampling algorithms for various sampling ratios. Higher values are better. The dark grey cells mark the algorithms in the Rank 0 of the Skott Knott plots. Avg Wins refers to the average number of times a sampling algorithm for a fixed sampling ratio came in Rank 0 across all the 8 datasets. Due to space constraints Recursive Random Projection is simplified to RRP. In the case of random & intelligent pruning we undersample the dataset, in no-sampling we don't do any sampling and by rest of the techniques we oversample.

V. RESULTS & DISCUSSIONS

In this section, we present our experimental findings and and answer the RQs based on the results.

RQ1: When considering the data imbalance problem, which undersampling/over-sampling technique performs the best by considering performance metrics of precision, recall, f1 score, and accuracy?
To evaluate the performance of undersampling/oversampling techniques, we train different learners SVM, LR & DT (results of LR & DT are in Appendix A) on the oversampled/undersampled dataset using different sampling ratios of 20%, 40%, 60%, 80% & 100% for all 8 datasets as seen in Table 3. We also add another Table 4 that denotes how a sampling technique helps in improving the model's performance(Accuracy, Precision, Recall, F1-Score) considering all sampling ratios at once. In Table 3, we report the best averaged F1-Score(across 10 random runs) for each sampling technique as the performance metric since the training dataset is still imbalanced until we reach 100% sampling, be it oversampling or undersampling. From the results, we infer that both undersampling and oversampling techniques improve the model's overall performance in classifying the data point into its respective class, in comparison to the original dataset without any undersampling & oversampling across all the metrics of Accuracy, Precision, Recall & F1-Score. There seems to be no clear winner across all 8 case studies. Similar results are observed for other ML models of DT & LR as well which are explained in Appendix Section A. However, one interesting thing to note from Table 3 is that 100% undersampling leads to at par or better performance in comparison to 100% oversampling for at least 6 datasets. Hence, our answer to RQ1 is as follows
If model performance is the only criteria for handling the data imbalance issue, then we there is no clear winner and we recommend using any sampling technique. However, there are other metrics as well which are important as discussed in RQ2-RQ4.

RQ2: Which sampling technique is statistically more significant while considering model performance, using average

Metric	Algorithm	Benchmarks								Avg Wins
		breast_ cancer	churn	JS_ Vuln	Ambari_ Vuln	Defect_ Eclipse_JDT	Defect_ Eclipse_PDE	Moodle_ Vuln	Defect_ Mylyn	
Accuracy	random_oversampling	0.76	0.93	0.91	0.97	0.84	0.87	0.98	0.87	1.00
	smote	0.78	0.93	0.91	0.97	0.84	0.87	0.97	0.87	1.62
	svm_smote	0.76	0.93	0.91	0.97	0.84	0.87	0.98	0.87	0.75
	gaussian_copula(SDV-G)	0.72	0.93	0.91	0.97	0.83	0.86	0.98	0.88	2.37
	RRP	0.78	0.93	0.91	0.96	0.84	0.86	0.98	0.87	0.75
	no_sampling	0.78	0.93	0.91	0.97	0.82	0.86	0.99	0.87	0.87
	random_pruning	0.76	0.93	0.91	0.97	0.85	0.86	0.99	0.87	2.75
	intelligent_pruning	0.78	0.93	0.91	0.97	0.85	0.87	0.99	0.87	8.25
Precision	random_oversampling	0.57	0.83	0.93	0.00	0.75	0.50	0.04	0.50	0.5
	smote	0.58	0.80	0.94	0.00	0.75	0.54	0.03	0.53	0.87
	svm_smote	0.60	0.82	0.90	0.00	0.71	0.54	0.00	0.50	0.37
	gaussian_copula(SDV-G)	0.56	0.82	0.89	0.00	0.70	0.56	0.00	0.71	1.25
	RRP	0.60	0.70	0.87	0.00	0.73	0.50	0.03	0.42	0.75
	no_sampling	0.75	0.93	0.96	0.00	0.80	0.00	0.00	0.75	0.62
	random_pruning	0.57	0.89	0.96	0.04	0.75	0.53	0.03	0.57	1.25
	intelligent_pruning	0.71	0.90	0.96	0.00	0.78	0.50	0.03	0.67	4.00
Recall	random_oversampling	0.40	0.80	0.58	0.00	0.64	0.53	0.17	0.56	0.62
	smote	0.44	0.72	0.61	0.00	0.59	0.45	0.29	0.56	0.25
	svm_smote	0.42	0.74	0.60	0.00	0.60	0.42	0.00	0.42	0.12
	gaussian_copula(SDV-G)	0.40	0.84	0.38	0.00	0.59	0.11	0.00	0.10	0.37
	RRP	0.39	0.82	0.45	0.00	0.70	0.51	0.20	0.58	0.62
	no_sampling	0.25	0.52	0.35	0.00	0.28	0.00	0.00	0.04	0.00
	random_pruning	0.53	0.87	0.42	0.60	0.68	0.59	0.43	0.53	0.87
	intelligent_pruning	0.61	0.82	0.58	0.60	0.72	0.64	0.43	0.58	1.62
F1-Score	random_oversampling	0.44	0.74	0.59	0.00	0.58	0.4	0.06	0.37	2.50
	smote	0.43	0.72	0.59	0.00	0.59	0.39	0.06	0.08	3.62
	svm_smote	0.42	0.72	0.59	0.00	0.59	0.39	0.00	0.36	1.87
	gaussian_copula(SDV-G)	0.41	0.70	0.53	0.00	0.50	0.18	0.00	0.18	0.62
	RRP	0.46	0.72	0.53	0.00	0.58	0.41	0.05	0.40	2.12
	no_sampling	0.38	0.65	0.52	0.00	0.38	0.00	0.00	0.08	0.12
	random_pruning	0.46	0.72	0.56	0.08	0.58	0.39	0.05	0.33	1.75
	intelligent_pruning	0.47	0.74	0.56	0.00	0.61	0.40	0.05	0.37	5.87

TABLE 4. Model comparison scores for SVM(Results for LR & DT are in Appendix A). All the tasks are evaluated by Accuracy, Precision, Recall, and F1-Score using different sampling algorithms. Higher values are better. The dark grey cells mark the algorithms in the Rank 0 of the Skott Knott plots. Avg Wins refers to the average number of times a sampling algorithm came in Rank 0 across all the 8 datasets. Due to space constraints Recursive Random Projection is simplified to RRP.

wins (Avg Wins) metric?

To measure which sampling technique performs best statistically, we adopted a non-parametric test called Skott-Knott, which ranks algorithms based on their performance across 10 random runs. To determine which sampling technique is statistically superior across model performance, we introduced a metric termed "Avg Wins," which denotes the average number of times an algorithm appears in Rank 0 of the Skott-Knott plots across all 8 datasets, providing us with a bigger picture of how statistically better an algorithm is over the others. From our results in Table 3 and Table 4, we infer that Intelligent Data Pruning outperforms all the other techniques significantly in terms of Avg Wins for all 8 case studies. Hence, our answer to RQ2 is as follows

Undersampling techniques, especially Intelligent Pruning, are promising sampling algorithms that can balance the dataset by eliminating redundant samples and retaining informative ones while achieving significantly good model metrics and Avg Wins in comparison to oversampling algorithms. Hence, if we consider both the model's performance and Avg Wins (statistical analysis), we would recommend Intelligent Data Pruning.

RQ3: Which way of undersampling(pruning) is better - Random Pruning or Intelligent Pruning?

From Table 4, we observe that Intelligent Pruning outperforms Random Pruning in terms of different model metrics, including Precision, Recall, and F1-Score. Additionally, the number of Avg Wins for Intelligent Pruning is significantly higher than for Random Pruning. This observation is intuitive because Intelligent Pruning focuses on removing redundant samples from larger clusters of the majority class while preserving unique, informative small-sized clusters. In contrast, random pruning may remove unique, informative small samples, potentially decreasing the model's performance. Hence, we answer RQ3 as follows

Intelligent Pruning is better than Random Pruning and results in better performance metrics and Avg Wins across all the datasets by just keeping informative and removing redundant samples. Therefore, from our observations, Intelligent Pruning is a better undersampling technique than Random Pruning.

RQ4: Which sampling technique is more scalable in cases of resource-constrained environment(insufficient data stor-

age) and limited compute time?

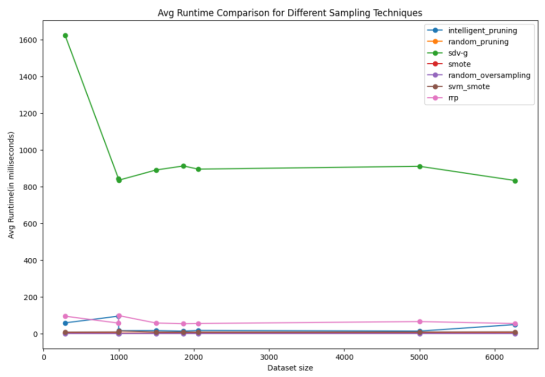

FIGURE 1. Average Runtime Performance of Different Sampling Algorithms

In cases where an organization can't afford to store more than a specific amount of data while restoring class balance in the dataset, better sampling techniques (possibly undersampling only) are required to be not only data storage efficient but also to maintain model performance. Therefore, based on our experimental results in Table 3, we infer that by undersampling, especially intelligent data pruning, we achieve comparable and/or better model performance across all datasets. In other words, we achieve significantly good performance even with 100% undersampling compared to 100% oversampling. This ensures that the data storage capacity is not exceeded while maintaining significant model performance.

Apart from storage capacity, another important metric to consider for measuring scalability is the sampling technique's compute time. Hence, we recorded the average execution time of each sampling algorithm across 10 random seeds. Fig. 1, shows the average runtime of each sampling algorithm. To analyze scalability, we initially measured the runtime of each algorithm in all 8 case studies. The scatter plots display points (x, y), where x represents the dataset size and y represents the runtime. Subsequently, we plot a line curve to depict the runtime trend as the dataset size grows. Notably, for small dataset sizes, all algorithms exhibit comparable runtimes. However, as the size of the dataset becomes larger, the runtime of the Gaussian-Copula-based SDV (green line) is significantly higher compared to other techniques. On the contrary, our proposed Intelligent Data Pruning (blue line) along with Random Undersampling, Random Oversampling, SMOTE, SVM-SMOTE, and RRP demonstrate better efficiency even for larger datasets. Thus, we answer RQ4 as follows

When scalability is a major issue, i.e, data storage capacity and compute time are limited—we can't resort to oversampling techniques. Hence, undersampling techniques such as Intelligent Pruning are promising sampling algorithms that utilize minimal storage and run sufficiently fast while achieving fairly good model performance.

RQ5: What suggestions can we provide from analyzing the conclusions in RQ1 to RQ4?

While evaluating various sampling techniques, we need to consider multi-dimensional criteria. To empirically evaluate all performance from RQ1 to RQ4, we transfer each of the criteria(model performance, scalability, and Avg Wins) to a 0-1 range and use the radar chart to visualize the performance. Specifically

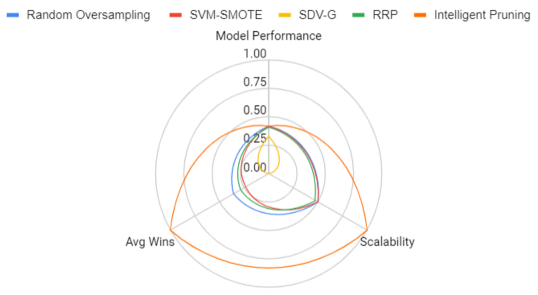

FIGURE 2. Radar Chart of Scores in multi-dimensional criteria

For *Model Performance*, the overall score for a sampling technique is computed by calculating the geometric mean of all the F1-Scores across all 8 datasets for the particular algorithm from Table 4. We then apply min-max scaling to achieve a model performance score in the range of 0 to 1. The higher the score, the better the technique for improving the model's performance.

For *Scalability*, the overall score for a sampling technique is computed by considering two factors simultaneously: (1) storage capacity (the number of samples used to make the prediction) and (2) the compute time of the algorithm. For storage capacity, we consider 100% sampling (pruning data from the majority class for undersampling and adding data to the minority class for oversampling). We then compute the geometric mean of the final number of samples used after sampling across all 8 datasets and apply min-max scaling. We subtract all of these scores from 1, as we want 1 to refer to the algorithm that requires the least amount of data and the opposite for 0. For compute time, we calculate the geometric mean of the average compute time of every algorithm across all datasets, followed by min-max scaling. We subtract these scores from 1 such that 1 refers to the algorithm that is the fastest and 0 means the algorithm is the slowest. Lastly, we take an average of the storage capacity and compute time scores of each algorithm, and scale them to the range of 0 to 1. The higher the score, the more scalable the sampling technique is.

For *Avg Wins*, the Avg Wins for a sampling technique is computed as described in IV-D. We refer to the Avg Wins results reported in Table 4 and scale them to the range of 0 to 1. The higher the score, the more times the sampling technique appeared in Rank 0 of the Skott-Knott analysis, hence more statistically significant.

We select 5 algorithms to be presented in our chart. The first one is random oversampling, which is one of the intuitive random-sampling baselines, followed by SVM-SMOTE, an advanced version of SMOTE oversampling. Next, we include RRP Oversampling, as it gave promising results in [7] compared to Howso Engine. Additionally, we include SDV-G, as it is one of the state-of-the-art techniques for oversampling. Finally, we include our proposed Intelligent Data Pruning method.

Figure 2 presents the radar chart of (a) Random Oversampling, (b) SVM-SMOTE, (c) SDV-G, (d) RRP and (e) Intelligent Data Pruning. As we can see, the area covered by Intelligent Data Pruning(orange line) is much higher than other techniques. Hence, based on our observations we provide the following recommendation and answer to RQ5:

- **If model performance is the only concern that needs to be fulfilled then any of the sampling techniques can be adopted as they are not significantly different in their metric values.**
- **However, if reliability based on statistical analysis and scalability of the algorithm is an important concern for the organization, then Intelligent Pruning should be considered as it uses way few but informative samples to achieve at par model performance and quite scalable in comparison to other techniques.**

VI. THREAT TO VALIDITY

Construct Validity is primarily concerned with how various parameters and the construction of a model can lead to different results. In our research, the threat of construct validity can arise from (a) selecting parameters in different sampling techniques and (b) the choice of settings when evaluating the synthetic data. For example, in the proposed Intelligent Data Pruning framework, the number of K-Means clusters formed for the majority class can influence the efficient undersampling of the data. Hence, we adopted the ELBO-Curve method [33] to find the optimal number of K-Means clusters. Additionally, when evaluating synthetic data through machine learning model training, we used an 80% train-test split for both original and synthetic data. Different split ratios can lead to different outcomes, but we chose this ratio as it is commonly used in machine learning studies. To mitigate this threat, we have packaged our experimental scripts as a Python package, allowing researchers to replicate our experiments with their chosen parameters.

Conclusion validity refers to the risk posed by using different evaluation metrics when drawing conclusions. To address this risk, we employ three distinct metrics(multi-dimensional criteria) to assess the sampling techniques: a statistical comparison using Avg Wins, model performance comparison metrics including Accuracy, Precision, Recall, and F1-Score, and scalability-based analysis, which encompasses many evaluation aspects of sampling techniques found in existing literature. Researchers should be aware that different conclusions may be drawn when different metrics are applied to our methods.

Internal validity concerns the accuracy of the treatment in causing the outcome. To minimize the impact of this threat, we collected eight widely used Software Engineering (SE) benchmarks from PMLB and papers from top SE conferences. We then ran all algorithms on these eight benchmarks. Additionally, we controlled and tested different sampling ratios until we achieved 100% class balance (using either undersampling or oversampling) in the dataset under consideration.

External validity refers to the concern about the generalizability of the experiment to other fields. To address this concern, our study's experimental focus is on machine learning, a widely applicable area in real-world applications. Furthermore, our scripts (written in Python) can be easily adapted to other datasets to explore various real-world applications.

VII. CONCLUSION

In this study, we explore different oversampling and undersampling techniques to address class imbalance in datasets and discuss their performance using various evaluation metrics. We propose our own Intelligent Data Pruning framework, which efficiently undersamples redundant samples from the majority class while preserving unique, rare, informative samples without any loss of information. We compare our proposed undersampling technique with existing techniques such as Synthetic Data Vault, Recursive Random Projection, SVM-SMOTE, and others. We evaluate these sampling techniques using multidimensional criteria, including (a) Model Performance metrics such as accuracy, precision, recall, and F1-score, (b) Statistical-based comparison using Avg Wins, and (c) Scalability comparison involving data storage capacity and compute time.

In the model performance comparison, we observe that all the algorithms perform equally well across all the case studies with no clear winner. However, when both Avg Wins (statistical analysis) and Scalability (storage capacity & compute time) come into play, Intelligent Data Pruning outperforms all the other algorithms by a significant margin while achieving almost similar and/or better model performance. Hence, from our extensive empirical analysis, we conclude that if model performance is the only criterion, then any of the sampling techniques considered in this study can be used. On the contrary, in cases where resources and time are limited, causing scalability issues, we recommend Intelligent Data Pruning since it scales fast and achieves competitive model performance. In future work, we will explore more sampling techniques, as well as more benchmarks across multiple domains. Moreover, the current Intelligent Data Pruning framework is constrained to binary classification tasks and uses K-Means as the clustering algorithm. Therefore, in the future, we aim to extend the framework to multi-classification tasks and use more sophisticated clustering techniques.

REFERENCES

[1] B. Saha and D. Srivastava, "Data quality: The other face of Big Data," 2014 IEEE 30th International Conference on Data Engineering, Chicago, IL, USA, 2014, pp. 1294-1297, doi: 10.1109/ICDE.2014.6816764.

[2] Liu, J., Li, J., Li, W. and Wu, J., 2016. Rethinking big data: A review on

the data quality and usage issues. ISPRS journal of photogrammetry and remote sensing, 115, pp.134-142.

[3] L. Gong, S. Jiang and L. Jiang, "Tackling Class Imbalance Problem in Software Defect Prediction Through Cluster-Based Over-Sampling With Filtering," in IEEE Access, vol. 7, pp. 145725-145737, 2019, doi: 10.1109/ACCESS.2019.2945858.

[4] Nascimento, D.C., Pires, C.E. and Mestre, D.G., 2016. Applying machine learning techniques for scaling out data quality algorithms in cloud computing environments. Applied Intelligence, 45, pp.530-548.

[5] Malhotra, R. and Khanna, M., 2017. An empirical study for software change prediction using imbalanced data. Empirical Software Engineering, 22, pp.2806-2851.

[6] N. Patki, R. Wedge, and K. Veeramachaneni, “The synthetic data vault,” in Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA), Montreal, QC, Canada, Oct. 2016, pp. 399–410, doi: 10.1109/dsaa.2016.49.

[7] Ling, X., Menzies, T., Hazard, C., Shu, J. and Beel, J., 2024. Trading Off Scalability, Privacy, and Performance in Data Synthesis. IEEE Access, 12, pp.26642-26654.

[8] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, pp.321-357.

[9] H. M. Nguyen, E. W. Cooper, K. Kamei, “Borderline over-sampling for imbalanced data classification,” International Journal of Knowledge Engineering and Soft Data Paradigms, 3(1), pp.4-21, 2009.

[10] Kubat, M. and Matwin, S., 1997, July. Addressing the curse of imbalanced training sets: one-sided selection. In Icml (Vol. 97, No. 1, p. 179).

[11] Parnami, A. and Lee, M., 2022. Learning from few examples: A summary of approaches to few-shot learning. arXiv preprint arXiv:2203.04291.

[12] Wang, Y., Yao, Q., Kwok, J.T. and Ni, L.M., 2020. Generalizing from a few examples: A survey on few-shot learning. ACM computing surveys (csur), 53(3), pp.1-34.

[13] Li, M., Zhang, H., Wu, R. and Zhou, Z.H., 2012. Sample-based software defect prediction with active and semi-supervised learning. Automated Software Engineering, 19, pp.201-230.

[14] He, Q., Shen, B. and Chen, Y., 2016, June. Software defect prediction using semi-supervised learning with change burst information. In 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC) (Vol. 1, pp. 113-122). IEEE.

[15] Schohn, G. and Cohn, D., 2000, June. Less is more: Active learning with support vector machines. In ICML (Vol. 2, No. 4, p. 6).

[16] G Menardi, N. Torelli, “Training and assessing classification rules with imbalanced data,” Data Mining and Knowledge Discovery, 28(1), pp.92-122, 2014.

[17] Cox, D.R., 1958. The regression analysis of binary sequences. Journal of the Royal Statistical Society Series B: Statistical Methodology, 20(2), pp.215-232.

[18] Quinlan, J.R., 1986. Induction of decision trees. Machine learning, 1, pp.81-106.

[19] Cortes, C. and Vapnik, V., 1995. Support-vector networks. Machine learning, 20, pp.273-297.

[20] Song, Qinbao, Yuchen Guo, and Martin Shepperd. "A comprehensive investigation of the role of imbalanced learning for software defect prediction." IEEE Transactions on Software Engineering 45.12 (2018): 1253-1269.

[21] Olson, R.S., La Cava, W., Orzechowski, P., Urbanowicz, R.J. and Moore, J.H., 2017. PMLB: a large benchmark suite for machine learning evaluation and comparison. BioData mining, 10, pp.1-13.

[22] Abaei G, Selamat A, Fujita H (2015) An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction. Knowl-Based Syst 74:28–39

[23] Ghotra B, McIntosh S, Hassan AE (2015b) Revisiting the impact of classification techniques on the performance of defect prediction models. In: 37th ICSE-vol 1. IEEE Press, pp 789–800

[24] Macbeth, G., Razumiejczyk, E. and Ledesma, R.D., 2011. Cliff's Delta Calculator: A non-parametric effect size program for two groups of observations. Universitas Psychologica, 10(2), pp.545-555.

[25] Xia, T., Krishna, R., Chen, J., Mathew, G., Shen, X. and Menzies, T., 2018. Hyperparameter optimization for effort estimation. arXiv preprint arXiv:1805.00336.

[26] Tu, H., Papadimitriou, G., Kiran, M., Wang, C., Mandal, A., Deelman, E. and Menzies, T., 2021, May. Mining workflows for anomalous data transfers. In 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR) (pp. 1-12). IEEE.

[27] Baltes, S. and Ralph, P., 2022. Sampling in software engineering research: A critical review and guidelines. Empirical Software Engineering, 27(4), p.94.

[28] Liu, X.Y., Wu, J. and Zhou, Z.H., 2008. Exploratory undersampling for class-imbalance learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(2), pp.539-550.

[29] Onan, A., 2019. Consensus clustering-based undersampling approach to imbalanced learning. Scientific Programming, 2019.

[30] Tsai, C.F., Lin, W.C., Hu, Y.H. and Yao, G.T., 2019. Under-sampling class imbalanced datasets by combining clustering analysis and instance selection. Information Sciences, 477, pp.47-54.

[31] Rayhan, F., Ahmed, S., Mahbub, A., Jani, R., Shatabda, S. and Farid, D.M., 2017, December. Cusboost: Cluster-based under-sampling with boosting for imbalanced classification. In 2017 2nd international conference on computational systems and information technology for sustainable solution (csitss) (pp. 1-5). IEEE.

[32] Liu, S. and Zhang, K., 2020. Under-sampling and feature selection algorithms for S2SMLP. IEEE Access, 8, pp.191803-191814.

[33] Blei, D.M., Kucukelbir, A. and McAuliffe, J.D., 2017. Variational inference: A review for statisticians. Journal of the American statistical Association, 112(518), pp.859-877.

[34] MacQueen, J., 1967, June. Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

[35] A. S. Tarawneh, A. B. Hassanat, G. A. Altarawneh and A. Almuhaimeed, "Stop Oversampling for Class Imbalance Learning: A Review," in IEEE Access, vol. 10, pp. 47643-47660, 2022

[36] Gu, X., Angelov, P.P. and Soares, E.A., 2020. A self-adaptive synthetic over-sampling technique for imbalanced classification. International Journal of Intelligent Systems, 35(6), pp.923-943.

[37] Xie, W., Liang, G., Dong, Z., Tan, B. and Zhang, B., 2019. An improved oversampling algorithm based on the samples' selection strategy for classifying imbalanced data. Mathematical Problems in Engineering, 2019.

[38] Karia, V., Zhang, W., Naeim, A. and Ramezani, R., 2019. Gensample: A genetic algorithm for oversampling in imbalanced datasets. arXiv preprint arXiv:1910.10806.

[39] Majumder, S., Chakraborty, J. and Menzies, T., 2024. When less is more: on the value of “co-training” for semi-supervised software defect predictors. Empirical Software Engineering, 29(2), pp.1-33.

.

APPENDIX A MODEL PERFORMANCE SCORES FOR LOGISTIC REGRESSION(LR) AND DECISION TREE(DT)

We also provide the model performance comparison for different techniques using Logistic Regression and Decision Tree. The results are available in our GitHub repository in A1.pdf.

APPENDIX B SAMPLING RATIOS PERFORMANCE FOR LOGISITC REGRESSION(LR) AND DECISION TREE(DT)

We provide the model performance scores for LR and DT as well across multiple sampling ratios. The results are available in our GitHub repository in A2.pdf.