Università degli Studi di Trieste

Department of Mathematics and Geosciences

Master's Degree in Data Science and Scientific Computing

MASTER'S THESIS

Data Augmentation Under DPG Constraints: Enhancing Diversity and Explainability

Candidate:
Gabriel Gustavo Costanzo

Advisor:
Prof. Sylvio Barbon Junior

Academic Year 2023-24

Abstract

Data augmentation addresses imbalanced datasets by generating synthetic examples, but traditional methods often neglect domain constraints, leading to unrealistic and less interpretable results. This thesis introduces a novel data augmentation technique guided by Decision Predicate Graphs (DPG) class bounds (constraints), a framework for capturing decision logic from ensemble models.

This approach, leveraging a Genetic Algorithm, generates diverse and constraint-abiding synthetic data within regions defined by DPG-derived class bounds. Two operational modes are explored: 'Original', prioritizing logical consistency, and 'Border', enhancing diversity near decision boundaries. The use of DPG constraints provides inherent explainability by enabling the tracing of synthetic samples back to specific, model-derived rules.

Experiments across diverse datasets demonstrate that DPG-guided augmentation significantly improves classification performance on minority classes while maintaining strict adherence to domain constraints. Unlike SMOTE and its variants, which tend to generate samples near existing data, DPG augmentation expands throughout the feature space, adding greater diversity. This controlled exploration, guided by interpretable DPG constraints, enhances model transparency and applicability.

Abstract (Italiano)

L'aumento dei dati affronta i set di dati sbilanciati generando esempi sintetici, ma i metodi tradizionali spesso trascurano i vincoli di dominio, portando a risultati non realistici e meno interpretabili. Questa tesi introduce una nuova tecnica di aumento dei dati guidata dai limiti di classe (vincoli) dei Decision Predicate Graphs (DPG), un framework per catturare la logica decisionale dai modelli di insieme.

Questo approccio, sfruttando un algoritmo genetico, genera dati sintetici diversi e che rispettano i vincoli all'interno di regioni definite dai limiti di classe derivati dal DPG. Vengono esplorate due modalità operative: 'Original', che dà priorità alla coerenza logica, e 'Border', che migliora la diversità vicino ai confini decisionali. L'uso dei vincoli DPG fornisce intrinseca spiegabilità, consentendo di ricondurre gli esempi sintetici a regole specifiche derivate dal modello.

Gli esperimenti su diversi set di dati dimostrano che l'aumento guidato dal DPG migliora significativamente le prestazioni del classificatore sulle classi minoritarie, pur mantenendo una rigorosa aderenza ai vincoli di dominio. A differenza di metodi come SMOTE e le sue varianti, che tendono a generare campioni vicino ai dati esistenti, l'aumento DPG si espande in tutto lo spazio delle features, aggiungendo maggiore diversità. Questa esplorazione controllata, guidata da vincoli DPG interpretabili, migliora la trasparenza e l'applicabilità del modello.

Contents

Chapter 1

Introduction

Modern machine-learning algorithms require large and representative datasets to learn effectively. In practice, however, real-world datasets often present challenges such as class imbalance, where certain categories (classes) have far fewer examples than others. Machine learning is increasingly relied upon in critical domains such as healthcare and finance, where accurate predictions are paramount [46]. For instance, in medical diagnosis, disease cases are usually rare compared to healthy cases [46]. These applications require large, high-quality datasets, but real-world data is often scarce, imbalanced, or difficult to collect. Another common issue is overfitting, where a model learns the training data too closely, including its noise or irrelevant patterns, resulting in poor generalization to new data [62]. To address these data quality challenges, researchers frequently employ synthetic data generation techniques, such as SMOTE (Synthetic Minority Oversampling Technique), GANs (Generative Adversarial Networks), and VAEs (Variational Autoencoders) [10] [62] [24] [19].

While these techniques can address data limitations, they often fail to respect essential domain constraints—logical rules that data must follow to remain realistic. For example, generating synthetic patient data with negative ages or physically impossible measurements violates fundamental domain constraints. Furthermore, existing methods can introduce subtle errors: SMOTE, relying on linear interpolation, may create implausible data points that don't reflect true data relationships; GANs, while statistically valid, might generate data that violates logical rules (e.g., improbable combinations of symptoms in a medical record); VAEs, focusing on capturing the global data distribution, are prone to producing "averaged" samples that fail to capture crucial minority class nuances [10] [24] [19]. Such inconsistencies undermine both the realism and interpretability (or explainability) of augmented datasets, especially in regulated areas like healthcare and finance, where decisions must be transparent and trustworthy [30] [23].

Hence, balancing the performance improvements offered by data augmentation against the strict adherence to domain-specific logic is a significant challenge. This thesis addresses this gap by exploring Decision Predicate Graphs (DPG) to guide the generation,

through a Genetic algorithm, of diverse and explainable augmented data, ensuring that the generated examples respect all essential domain constraints. By leveraging DPGs, this work derives logical decision rules from a trained ensemble model, ensuring the generated samples remain consistent with the underlying data structure and do not generate impossible scenarios.

1.1 Decision Predicate Graphs (DPG)

One promising framework for integrating model-derived constraints is the Decision Predicate Graph (DPG). As introduced by Arrighi et al. (2024) [5], DPG provides a global interpretation of ensemble models by capturing how features and decisions interact at each node (predicate) level. DPGs offer a model-agnostic approach to understand decision-making processes within ensembles such as Random Forests or Gradient Boosting Machines [5]. This is particularly valuable because these models, while powerful, are often considered "black boxes" due to their complexity.

To fully appreciate the benefits of DPGs, it is helpful to understand their construction and key components:

- *Predicate Extraction:* The DPG algorithm begins by traversing each decision tree in the ensemble. At each internal node of a tree, a decision rule is extracted. This rule, which involves a feature, a comparison operator (e.g., '<', '>', '='), and a threshold (for numerical features) or set of values (for categorical features), is termed a "predicate". Each predicate represents a condition that must be satisfied for a sample to follow a particular branch of the tree [5].
- *Graph Construction:* A directed graph is then constructed, where each node in the graph represents a unique predicate extracted from the ensemble. Edges between nodes indicate that the corresponding predicates co-occur in a decision path within at least one of the decision trees. The weight of each edge represents the frequency with which the two predicates are satisfied consecutively by samples in the training dataset [5].
- *Global Interpretation:* By aggregating predicate co-occurrence frequencies across all trees in the ensemble, the DPG provides a comprehensive picture of the model's decision logic. This allows to capture complex interactions across the ensemble, leading to a global interpretation [5]. This global interpretation has increased importance specially considering the local interpretability of some XAI methods.

These predicate-level constraints yield a robust blueprint for generating new data consistent with both the model logic and the realistic domain ranges. Furthermore, DPGs have interesting utilities for a ML expert [5]:

- *Visualisation*: Provides a immediate advantage by enabling the visualisation of the entire tree-based ensemble models through a single comprehensive graph.
- *Constraints*: Calculate the classification boundary values of each feature associated with each class.
- *Centrality*: Identify potential bottleneck nodes that correspond to crucial decisions.
- *Community*: identify groups of predicates that significantly contribute to classifying samples into specific classes.

Therefore, in our research, the DPG serves as the cornerstone for addressing the challenge of generating high-quality, explainable augmented data. We exploit the inherent logic encoded within DPGs to guide the generation process, ensuring that synthetic data points respect the relationships between features and decision boundaries learned by the ensemble model.

1.2 Research Objectives

The research objectives of this thesis are to develop a data augmentation technique that:

- Generates improved samples with enhanced explainability while avoiding infeasible samples.
- Enhance both diversity, by generating samples that cover underrepresented regions in the problem space, and explainability, by ensuring that augmented data strictly respect logical domain constraints identified through DPG.

In simpler terms, the main research goal is to create a new data augmentation method (DPG Data Augmentation) that generates realistic, diverse, and understandable augmented data by incorporating the logic of machine learning models and domain-specific rules into the data generation process. The GA optimization and two modes of operation are enablers to achieve this goal.

1.3 Motivation and Context

As machine learning models are deployed in increasingly sensitive domains such as healthcare and finance, the need for Explainable AI (XAI) has become paramount. XAI demands that models and augmentation techniques do more than produce accurate predictions; they must also elucidate their logic and structure [60]. Trust and transparency are essential when model decisions directly impact human lives or financial stability, necessitating that these decisions are interpretable by experts and regulators. To that end, oversampling methods can be broadly categorized as interpolation-based or generative-based, each presenting distinct advantages and limitations in balancing performance, explainability, and constraint adherence [73].

Interpolation-based methods, such as SMOTE [10] and its variants (e.g., Borderline-SMOTE [25], SVM-SMOTE [20]), generate new samples by interpolating between existing instances. These methods offer simplicity and computational efficiency, making them popular choices. However, their inherent design focuses on local neighborhood relationships, often overlooking broader feature dependencies or domain-specific constraints. Consequently, they can produce unrealistic samples by failing to respect feature relationships or generating values outside valid ranges. For example, in a credit risk model, SMOTE might create a synthetic applicant with an implausible combination of income and debt, violating lending guidelines. This issue arises because interpolation operates without awareness of the underlying rules that govern real-world data.

Generative methods, including GANs [24] [14] and VAEs [19], learn the underlying data distribution and generate new samples from it. While these methods can capture complex data patterns and relationships, they often require substantial training data and careful tuning to avoid generating logically inconsistent or unrealistic samples [17]. For instance, GANs, while capable of generating realistic images, may struggle to produce tabular data that adheres to specific business rules or regulations. Furthermore, generative methods often operate as "black boxes", making it difficult to understand "why" a particular synthetic sample was generated. While recent works aim for the interpretability of synthetic data, generated by generative approaches, these are still in the early stages [30].

Despite the strengths of existing oversampling methods, both interpolation-based and generative approaches often fail to systematically incorporate domain knowledge and the internal logic of trained models [30]. Even with increased attention from XAI research, existing synthetic data generation techniques often do not take into account the model's decision-making logic, increasing the chance that the new instances will not improve model performance or will be inadequate for critical applications. As a result, they risk producing synthetic samples that violate crucial feature relationships, logical constraints, or known dependencies. This Thesis addresses this critical gap by integrating Decision Predicate Graphs (DPG) into a novel data augmentation pipeline, specifically targeting the limitations of interpolation-based methods. DPGs provide a framework for representing and reasoning about the decision logic of ensemble models, enabling the generation of synthetic data that adhere to this decision-making logic, while still offering diversity and respecting domain-specific constraints. This approach enhances the reliability and explainability of augmented datasets, ensuring that synthetic samples reflect both the statistical properties of the data and the decision-making processes of the model.

1.4 Proposed Approach: DPG-Guided Data Augmentation

This thesis proposes a novel data augmentation approach that leverages DPG class bounds to guide data augmentation. Our method aims to enhance both diversity, by generating

samples that cover underrepresented regions in the feature space, and explainability, by ensuring that synthetic data strictly respect logical domain constraints identified through DPG. Specifically, the proposed approach takes advantage of previously defined DPG class constraints derived from an ensemble model (e.g., Random Forest) [5], ensuring the logical coherence and realism of newly generated examples.

At the core of this solution is a Genetic Algorithm (GA), chosen specifically due to its flexibility in accommodating complex domain constraints and its effectiveness in exploring diverse regions of the feature space [26] [15]. GA is particularly suited for this task because it can efficiently search through large and constrained solution spaces, iteratively optimizing populations of candidate solutions through processes inspired by biological evolution (selection, crossover, and mutation). The GA framework plays a central role in our solution, as it explicitly penalizes synthetic examples violating DPG constraints while rewarding those enhancing minority-class coverage.

The proposed GA-based framework incorporates a key hyperparameter called the "generation goal" or mode, which defines two operational states within a unified augmentation strategy:

- Original Mode: Synthetic samples are directly generated within predefined DPG-derived regions, explicitly promoting logical consistency and explainability.
- Border Mode: Synthetic samples are strategically biased towards regions close to class decision boundaries. This mode prioritizes the diversity and informativeness of generated samples, as these borderline regions are typically crucial for improving classifier performance on minority classes [25].

Integrating both modes into a single approach provides flexibility, allowing users to emphasize either strict constraint adherence or exploration near critical decision boundaries depending on the specific needs of their application domain.

1.5 Research Questions

Correspondingly, this thesis seeks to answer whether DPG-based augmentation can (1) improve classification performance and reduce class-specific bias compared to SMOTE variants, (2) enforce domain-specific constraints effectively, (3) enhance interpretability, (4) explore differences between “original” vs. “border” generation modes, (5) produce higher-quality augmented data than baseline oversamplers, and (6) remain robust across various dataset sizes and imbalance levels.

1.6 Methodology Overview

Our methodology involves the following clearly defined and justified steps:

- **DPG Constraints:** We first train an ensemble model, such as a Random Forest, on the original imbalanced dataset. This step allows us to derive a set of logical decision rules—decision predicates—that reflect the model's internal decision logic (*if–then conditions*) and enforce domain realism [5].
- **Constraint-Aware Genetic Algorithm (GA):** We propose using a GA framework because it naturally accommodates complex constraint spaces and effectively explores the data space for diverse sample generation. Our GA explicitly rewards: (a) correct minority-class classification, (b) increased diversity across the feature space, and (c) strict adherence to DPG-derived logical constraints. This design ensures that the generated synthetic data respects domain logic and significantly improves minority-class representation.
- **Generation Modes as Hyperparameters:** Our GA framework includes two operational modes controlled by a hyperparameter called the *generation goal*. Specifically, we evaluate (1) the *original mode*, which generates synthetic samples strictly within established DPG-derived regions, prioritizing logical consistency and explainability; and (2) the *border mode*, where samples are intentionally biased towards challenging, boundary regions between classes, maximizing diversity and potentially boosting minority-class recognition.
- **Classifier Selection and Evaluation:** To comprehensively evaluate the impact of our augmentation approach, we selected three widely-used and interpretable classifiers: Decision Trees (DT), Logistic Regression (LR), and k-Nearest Neighbors (kNN). These algorithms were specifically chosen because they are known to be sensitive to class imbalance issues, thus clearly revealing the effectiveness of our augmentation strategy in mitigating imbalance-induced biases [45] [63]. We measure their performance primarily using the F1 score, as this metric effectively balances precision and recall, making it especially suitable for imbalanced classification contexts. Additionally, we monitor the number of constraint violations to assess the realism and logical coherence of generated samples. For further validation, we benchmark our approach against established oversampling methods such as SMOTE and its variants [25] [8], as well as more advanced oversampling techniques [12].
- **Statistical Analysis:** Finally, we employ rigorous statistical tests—such as ranking-based Friedman tests with Nemenyi post-hoc comparisons [21] [53], [18]—across multiple datasets to systematically verify whether our DPG-guided augmentation approach leads to statistically significant improvements in classifier performance and interpretability compared to baseline methods. The Friedman test is a non-parametric test for detecting differences in multiple related samples, while the Nemenyi post-hoc test is used to determine which specific pairs of algorithms differ significantly from one another after a significant Friedman test result, offering a more granular analysis of

performance differences [27].

- **Diversity Analysis:** We analyze the distribution of the new samples generated by our proposed DPG augmentation algorithm versus standard oversampling techniques (SMOTE and its variants) on the *Mammographic Mass* dataset [UCI, 2023]. PCA scatter plots visually demonstrate differences in the diversity of generated samples at various augmentation levels (5%, 15%, 30%, 50%). This comparison illustrates how our two operational modes (*Original* and *Border*) uniquely influence augmented data distribution, either broadly across feature space or near decision boundaries, respectively.
- **Running Time Benchmark:** We measure and report the computational cost (average running time) for each augmentation approach at 50% augmentation. This benchmark highlights the computational trade-offs between our border-oriented GA (high computational cost) and the more efficient original mode, as well as other conventional methods (e.g., SMOTE variants). These insights help contextualize our method's practical viability in different application scenarios.

1.7 Contributions

Through this work, we aim to establish a robust paradigm for data augmentation under DPG constraints, bridging the gap between faithful data augmentation and explainability in real-world imbalanced classification tasks. We hypothesize that DPG-guided augmentation not only boosts performance but also preserves or improves global interpretability, offering a powerful tool for domains where trust, transparency, and fidelity are paramount [5] [46] [28].

Chapter 2

Background

2.1 Data Augmentation in the Context of Imbalanced Datasets

Data augmentation refers to generating new data samples from an existing dataset in order to improve model performance, reduce overfitting, and mitigate biases [62] [39]. Machine learning models often struggle with imbalanced datasets, where some classes have significantly fewer examples than others. This imbalance can lead to biased models that perform poorly on the minority class, which is often the class of greatest interest in critical applications such as medical diagnosis (detecting rare diseases) or fraud detection [25] [6].

Traditional approaches to address class imbalance often fall short. Oversampling techniques like SMOTE (Synthetic Minority Over-sampling Technique) generate synthetic minority class examples by interpolating between existing samples [10]. However, these methods often overlook important domain constraints—logical rules that data must follow to remain realistic. For example, generating synthetic patient data with negative ages or physically impossible lab values violates fundamental domain constraints. Such inconsistencies undermine both the realism and interpretability (or explainability) of augmented datasets, especially in regulated areas like healthcare and finance, where decisions must be transparent and trustworthy [30] [23].

Therefore, a key challenge in data augmentation for imbalanced datasets lies in balancing the performance improvements offered by oversampling against the strict adherence to domain-specific logic. This requires augmentation techniques that can generate diverse and informative synthetic data while ensuring constraint compliance. This chapter explores algorithms capable of producing diverse and explainable augmented data while ensuring the generated examples respect all essential domain constraints.

Beyond simply improving classification performance, data augmentation can address subtle objectives. For instance, in some applications, the minority class is extremely rare or especially critical (e.g., disease cases in medical data or fraudulent transactions in finance).

By creating new synthetic minority examples, a classifier can be trained to better handle these underrepresented cases [25] [6]. Likewise, augmented data can help the model handle borderline or corner-case scenarios if the real data rarely occupy those feature regions [62] [71].
However, data augmentation for tabular and time-series tasks is far from trivial. Many tabular features have logical constraints (e.g., age 0) or domain-specific relationships (e.g., consistent categories). Violating these constraints can degrade the overall data quality and hamper learning [28]. In addition, augmented data might replicate biases from the original dataset or inadvertently produce unrealistic records if the augmentation method does not respect domain distributions [49]. Ensuring both fidelity (closeness to real data) and diversity (covering minority gaps) can be difficult, especially in regulated fields with ethical or privacy concerns [19] [70].
Moreover, the choice of augmentation technique often depends on the imbalance severity, dimensionality, and nature of features (categorical vs. numeric), as well as on the computational budget or the need for interpretability [62]. Some methods (e.g., SMOTE) are simple to implement yet limited to local linear interpolation. Others (e.g., advanced generative models) capture complex data manifolds but require larger datasets and heavier training [44] [73]. Finally, advanced "explainable" or "counterfactual" augmentation can ensure that newly generated data remain consistent with minimal logical changes that flip class labels [66]. In the next section, we explore how simpler data augmentation compares to broader synthetic data generation.

2.2 Difference between Data Augmentation and Synthetic Data Generation

Though often conflated, data augmentation (local transformations) and synthetic data generation (global distribution modelling) can follow distinct motivations [62] [31]. Typically:

1. **Data augmentation:** Focuses on local, direct transformations of existing samples. In tabular contexts, one might apply linear interpolation between neighbors or add small random noise to features. These additions remain anchored to real data points, preventing overly unrealistic samples. For instance, an interpolation-based method (e.g., SMOTE) linearly blends a sample $\mathbf{x}$ with its neighbor $\mathbf{y}$, so the new instance $\mathbf{x}'$ is guaranteed to lie on the line segment between $\mathbf{x}$ and $\mathbf{y}$ [25].

2. **Synthetic data generation:** Involves training a global model (GAN, VAE, or Diffusion model) that learns the entire distribution, sampling from it to create new data potentially far from any single real point [31]. A well-trained generative model

can uncover multi-modal or nonlinear structures. However, it often requires more computational overhead, more data, and thorough hyperparameter tuning [19] [70].

These categories are not mutually exclusive. Some advanced pipelines combine local augmentation with partial generative logic. Others incorporate domain-driven constraints into generative approaches so that the synthetic output remains realistic [66]. Still, the difference remains: data augmentation is typically simpler, anchored, and strongly guided by existing points, while synthetic generation aims for a global notion of plausibility that can produce more "novel" data. For smaller or moderate datasets, local augmentation can suffice; for high complexity or multi-modal data, generative models might be necessary [73].

Additionally, the risk of out-of-distribution artifacts is typically lower in local augmentation since transformations revolve around known points [25]. Generative models can produce more radical samples if training or constraints are inadequate [44]. The final choice depends on domain complexity, data size, and interpretability or fairness needs [49].

2.3 Main Data Augmentation Methods

We now survey the principal augmentation approaches for tabular or non-visual data. A summary of these methods is presented in Table 1, categorizing each by general approach—"Interpolation-based," "Generative-based," "Evolutionary," or "Counterfactual"—and listing their typical domains, advantages, and limitations.

2.3.1 Interpolation Methods

SMOTE (Synthetic Minority Over-sampling Technique [10] is the classic interpolation-based method for class imbalances. Given a minority instance $\mathbf{x}$ and its neighbor $\mathbf{y}$, a new point $\mathbf{x}_{new}$ is obtained by:

$$\mathbf{x}_{new} = \mathbf{x} + \alpha\,(\mathbf{y} - \mathbf{x}), \quad \alpha \in [0, 1].$$

This local interpolation expands the minority region. SMOTE often boosts minority recall but can create overlapping or borderline confusion without further refinements [25].

Borderline-SMOTE [25] only interpolates minority points near the decision boundary, ignoring outliers or "safe" points. Let $\mathbf{x}$ be a minority instance heavily surrounded by majority neighbors; that "borderline" status triggers targeted interpolation:

$$\mathbf{x}_{border} = \mathbf{x} + \beta\,(\mathbf{z} - \mathbf{x}), \quad \beta \in [0, 1],$$

where $\mathbf{z}$ is a neighbor in the minority. This method focuses expansions where they are needed most: near the boundary.

SMOTE-ENN [8] first applies SMOTE, then removes suspicious or mislabeled points using Edited Nearest Neighbors. Suppose we have an oversampled dataset $\mathcal{D}_{SMOTE}$. For each sample $\mathbf{s}$, if the majority of its k-neighbors do not share its label, $\mathbf{s}$ is deleted. This cleaning step attempts to reduce borderline overlap and retain only consistent points.

SVM-SMOTE [20] leverages an SVM boundary to oversample near support vectors. If $\mathbf{x}$ is a support-vector minority point, we choose a minority neighbor $\mathbf{n}$ and generate:

$$\mathbf{x}_{svm-new} = \mathbf{x} + \gamma(\mathbf{n} - \mathbf{x}), \quad \gamma \in [0, 1].$$

This alignment with SVM margins often helps minority classification, albeit at higher computational cost for repeated SVM training [73].

MSMOTE [29] categorizes minority points as safe, borderline, or outlier via local density. Safe points undergo standard SMOTE. Borderline points are carefully interpolated. Outliers may be excluded or partially used to minimize noise risk. By distinguishing these categories, MSMOTE better tailors the interpolation process.

Polynomial-Fitting SMOTE [22] attempts a nonlinear approach. Rather than linear interpolation, it fits a polynomial curve among local minority neighbors:

$$\mathbf{x}_{poly}(t) = \sum_{d=0}^{D} \alpha_d \, t^d,$$

where coefficients $\{\alpha_d\}$ are estimated from actual points. Synthetic samples $\mathbf{x}_{poly}(t)$ are drawn along the curve, capturing more complex shapes.

LVQ-SMOTE [52] merges SMOTE logic with Learning Vector Quantization (LVQ). An LVQ codebook $\mathbf{c}$ represents a local cluster center for minority data. The new point could be:

$$\mathbf{x}_{lvq-new} = \mathbf{c} + \delta \, (\mathbf{z} - \mathbf{c}), \quad \delta \in [0, 1],$$

where $\mathbf{z}$ is an actual minority sample. This approach can reduce the random expansions from outliers, as codebooks reflect more central features [52].

DE_oversampling [12] uses Differential Evolution to oversample. Let $\mathbf{x}$ be a chosen minority "base," $\mathbf{x}_1, \mathbf{x}_2$ be two other minority samples. A new synthetic $\mathbf{v}$ is formed by:

$$\mathbf{v} = \mathbf{x} + F \cdot (\mathbf{x}_1 - \mathbf{x}_2),$$

where $F \in [0, 2]$ is a scaling factor. Then a crossover step merges $\mathbf{v}$ with $\mathbf{x}$, finalizing the synthetic point. This iterative evolutionary approach can discover better coverage if well-tuned.

2.3.2 Generative Methods

Whereas SMOTE-like methods rely on local interpolation, generative approaches attempt to model the entire data distribution to produce new points from a global perspective [19] [44].

GAN-based Methods

A Generative Adversarial Network (GAN) [44] comprises a generator $G(\mathbf{z}; \theta_g)$ and a discriminator $D(\mathbf{x}; \theta_d)$. The adversarial objective is:

$$\min_G \max_D \Big[\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \big(\log D(\mathbf{x}) \big) \; + \; \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} \big(\log \big(1 - D(G(\mathbf{z})) \big) \big) \Big].$$

Over training epochs, D learns to discriminate real from synthetic, while G learns to fool D. Once trained, one samples $\mathbf{z} \sim p_z$ (often Gaussian) and obtains $\mathbf{x}' = G(\mathbf{z})$. For tabular data, specialized conditional or discrete feature adaptations exist [73].
One extension is the "non-saturating loss," replacing $\log(1-D(G(\mathbf{z})))$ with $-\log(D(G(\mathbf{z})))$. Another is the Wasserstein loss for better gradient signals [44]. In imbalanced classification, one can push G to produce more minority-labeled samples, e.g., via conditional label embeddings or cost-sensitive training [2].
GANs excel at capturing complex distributions but can encounter **mode collapse**—where the generator produces limited variety—or training instability if the dataset is small or the hyperparameters poorly tuned. Yet, for large, complex data, a well-trained GAN can yield highly realistic synthetic points that surpass simpler interpolation [70].

Variational Autoencoders (VAEs)

VAEs [19] model data as latent variables. The encoder $q_\phi(\mathbf{z}|\mathbf{x})$ approximates the posterior of $\mathbf{z}$, and the decoder $p_\theta(\mathbf{x}|\mathbf{z})$ reconstructs $\mathbf{x}$. The optimization objective with the Kullback-Leibler term is:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \Big[\log p_\theta(\mathbf{x}|\mathbf{z}) \Big] - \mathrm{KL} \Big(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z}) \Big).$$

After training, we sample $\mathbf{z} \sim p(\mathbf{z})$ (often $\mathcal{N}(\mathbf{0}, \mathbf{I})$), then decode to get a new synthetic instance $\mathbf{x}'$. For a tabular dataset with partial discrete features, one can embed them in a "soft" representation or adopt Gumbel-softmax re-parameterization. VAEs typically run more stably than GANs but risk producing "averaged" samples that miss sharp minority clusters or tail phenomena [44].
In an imbalanced scenario, the VAE might trivially focus on the majority distribution unless guided. Some authors incorporate class conditioning or sample weighting so that the minority region is represented in the latent space. If done well, the VAE can produce

well-rounded expansions of the minority. However, if data are extremely limited or high-dimensional, the VAE might not capture minority nuances [66].

Diffusion-Based Models

Diffusion models proceed by a forward noising process and a learned reverse denoising. Let $\{\beta_t\}$ be a noise schedule. The forward step from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$ is:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\Big(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t \mathbf{I}\Big).$$

After T steps, $\mathbf{x}_T \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$. The reverse step is parameterized by a neural network $\boldsymbol{\epsilon}_\theta$ that predicts noise $\boldsymbol{\epsilon}$, so that:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\Big(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}\Big),$$

where $\mu_\theta(\mathbf{x}_t, t)$ depends on the predicted noise [44]. Sampling starts from noise $\mathbf{x}_T$ and iteratively applies reverse diffusion to yield $\mathbf{x}_0$.

For images, these approaches excel at high-fidelity generation with less mode collapse. For tabular data, while the concept remains similar, encoding discrete or multi-type features is an open challenge [70]. Although the heavy computational overhead might be impractical for many tabular tasks, advanced research aims to harness diffusion for complex or multi-modal data generation.

2.3.3 Mode Collapse Problem

Mode collapse is a significant challenge in generative models, particularly in Generative Adversarial Networks (GANs), where the generator fails to produce diverse outputs, instead learning to generate a small subset of patterns that consistently deceive the discriminator [24]. This issue arises from the adversarial training process, in which the generator focuses on a limited set of data samples rather than capturing the full distribution of the dataset [59]. As a result, the generated synthetic data lacks variety, which can be detrimental in applications requiring high diversity, such as image generation, tabular data synthesis, and data augmentation for machine learning models. Depending on the severity, mode collapse can manifest as complete mode collapse, where only one type of output is produced, or partial mode collapse, where limited variation exists but does not represent the full dataset [47].

The consequences of mode collapse are particularly problematic in domains that require balanced and representative datasets. In classification tasks, synthetic data generated from a model suffering from mode collapse may introduce biases by failing to represent underrepresented groups adequately [14].

2.4 Genetic Algorithms (GA)

Genetic Algorithms (GAs) are a class of population-based metaheuristics inspired by the process of natural evolution [26]. They provide a robust framework for exploring high-dimensional and complex solution spaces and, in this thesis, are particularly useful for generating diverse and constraint-abiding synthetic data samples. GAs mimic biological processes such as selection, crossover, and mutation to efficiently search the feature space while explicitly penalizing synthetic examples that violate DPG constraints. GAs operate on a population of candidate solutions—often referred to as *chromosomes*—which evolve over successive generations to optimize an objective function. Foundational texts such as *Essentials of Metaheuristics* [36] and *A Field Guide to Genetic Programming* [56] provide comprehensive overviews of these techniques, while recent studies have successfully applied GAs to synthetic data generation [14] and data augmentation for imbalanced datasets [46]. Mathematically, a single GA iteration can be described as follows. Let

$$P_t = \{\mathbf{g}_1^{(t)}, \mathbf{g}_2^{(t)}, \ldots, \mathbf{g}_N^{(t)}\}$$

denote the population at generation t. A selection operator $\mathcal{S}$ chooses a subset S_t based on fitness:

$$S_t = \mathcal{S}(P_t, F),$$

where the fitness function F evaluates each candidate. Next, a crossover operator $\mathcal{C}$ recombines pairs of candidates to produce offspring:

$$O_t = \mathcal{C}(S_t).$$

Finally, a mutation operator $\mathcal{M}$ introduces random variations:

$$P_{t+1} = \mathcal{M}(O_t).$$

This process is repeated until a convergence criterion or a predetermined number of generations is reached.

2.4.1 GA Conceptual Basis

Firstly, each GA **chromosome** can be represented as a vector

$$\mathbf{g} = (g_1, g_2, \ldots, g_n)$$

that encodes a candidate solution to the problem. The GA initializes a population $\{\mathbf{g}_1, \ldots, \mathbf{g}_N\}$ either randomly or via a heuristic. In each generation, the fitness $F(\mathbf{g}_i)$ of every candidate is computed; individuals with higher fitness are more likely to be se-

lected as parents. Crossover operators, such as uniform crossover, merge information from two parent vectors to produce offspring. For example, uniform crossover is defined as:

$$c_j = \begin{cases} p_j, & \text{with probability 0.5,} \\ q_j, & \text{with probability 0.5,} \end{cases}$$

where **p** and **q** are parent chromosomes. Mutation operators perturb offspring by adding a small random change δ to one or more genes:

$$c_j \leftarrow c_j + \delta.$$

Over successive generations, the population evolves, converging toward candidate solutions that yield higher values of the objective function.

Algorithm 1 The Genetic Algorithm (GA)

```
1. popsize ← desired population size
2. P ← {}
3. for popsize times do
|  4. P ← P ∪ {new random individual}
5. Best ← □
6. repeat Best is the ideal solution or we have run out of time
|  7. for each individual P_i ∈ P do
|  |  8. AssessFitness(P_i)
|  |  9. if Best = □ or Fitness(P_i) > Fitness(Best) then
|  |  |  10. Best ← P_i
|  |  end
|  11. Q ← {}
|  12. for popsize/2 times do
|  |  13. Parent P_a ← SelectWithReplacement(P)
|  |  14. Parent P_b ← SelectWithReplacement(P)
|  |  15. Children C_a, C_b ← Crossover(Copy(P_a), Copy(P_b))
|  |  16. Q ← Q ∪ {Mutate(C_a), Mutate(C_b)}
|  17. P ← Q
19. return Best
```

2.4.2 Selection

In Evolution Strategies (ES), we just lopped off all but the μ best individuals, a procedure known as Truncation Selection. We have more options because the Genetic Algorithm performs iterative selection, crossover, and mutation while breeding. Unlike Truncation Selection, the GA's `SelectWithReplacement` procedure can (by chance) pick certain individuals over and over again, and it also can (by chance) occasionally select some low-fitness individuals. In an ES, an individual is the parent of a fixed and predefined number of children, but not so in a GA [36].

The original `SelectWithReplacement` technique for GAs was called Fitness-Proportionate

Selection, sometimes known as Roulette Selection. In this algorithm, we select individuals in proportion to their fitness: if an individual has a higher fitness, it's selected more often. To do this, the individuals are "sized" according to their fitness as shown in Figure 2.4.1.

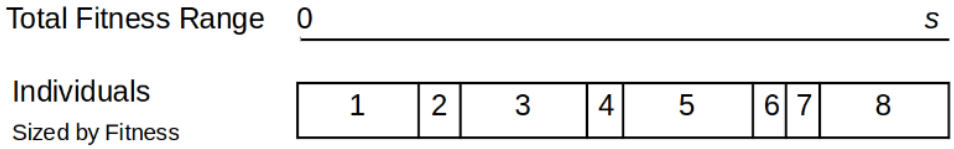

Figure 2.4.1: Array of individual ranges in Fitness Proportionate Selection. [Adapted from Sean Luke, 2013]

Let

$$s = \sum_i f_i \quad \text{(total fitness)}$$

be the sum fitness of all the individuals. A random number from 0 to s falls within the range of some individual, which is then selected.

There is a big problem with the methods described so far: they presume that the actual fitness value of an individual means something important. But often, we choose a fitness function such that higher ones are "better" than smaller ones, and don't mean to imply anything more. Even if the fitness function was carefully chosen, consider the following situation, where a fitness function goes from 0 to 10. Near the end of a run, all the individuals have values like 9.97, 9.98, 9.99, etc. We want to finesse the peak of the fitness function, so we want to pick the 9.99-fitness individual. But to Fitness-Proportionate Selection, all these individuals will be selected with nearly identical probability. The system has converged to just doing random selection. To fix this, we could scale the fitness function to be more sensitive to the values at the top end of the function. But to remedy the situation, we need to adopt a non-parametric selection algorithm, which throws away the notion that fitness values mean anything other than bigger is better and just considers their rank ordering. Truncation Selection does this, but the most popular technique by far is **Tournament Selection** [36], an astonishingly simple algorithm:

Algorithm 2 Tournament Selection

1. $P \leftarrow$ population
2. $t \leftarrow$ tournament size, $t \geq 1$
3. `Best` $\leftarrow$ random individual from P (with replacement)
4. **for** *i from 2 to t* **do**
5. `Next` $\leftarrow$ random individual from P
6. **if** *Fitness(`Next`)* $>$ *Fitness(`Best`)* **then**

end

7. `Best` $\leftarrow$ `Next`
8. **return** *`Best`*

We return the fittest individual of some t individuals picked at random, with replacement, from the population.Tournament Selection has become the primary selection technique used for the Genetic Algorithm and many related methods for several reasons. First, it's not sensitive to the particulars of the fitness function. Second, it's dead simple, requires no preprocessing, and works well with parallel algorithms. Third, it's tunable: by setting the tournament size t, you can change how selective the technique is. At the extremes, if t = 1, this is just a random search. If t is very large (much larger than the population size itself), then the probability that the fittest individual in the population will appear in the tournament approaches 1.0, and so Tournament Selection just picks the fittest individual each time (put another way, it approaches Truncation Selection with μ = 1) [36].

2.4.3 Elitism

Elitism is simple: we augment the Genetic Algorithm to directly inject into the next population the fittest individual or individuals from the previous population. These individuals are called the elites. By keeping the best individual (or individuals) around in future populations, this algorithm begins to resemble "$\mu + \lambda$" (μ parents plus the λ new children), and has similar exploitation properties. This exploitation can cause premature convergence if not kept in check: perhaps by increasing the mutation and crossover noise, or weakening the selection pressure, or reducing how many elites are being stored. A minor catch [36]. If you want to maintain a population size of popsize, and you're doing crossover, you'll need to have popsize, minus the number of elites, be divisible by two, as in this algorithm:

Algorithm 3 The Genetic Algorithm with Elitism

```
popsize ← desired population size
n ← desired number of elite individuals                ▷ popsize - n should be even
P ← {}
for popsize times do
    P ← P ∪ {new random individual}
Best ← □
repeat Best is the ideal solution or we have run out of time
    for each individual P_i ∈ P do
        AssessFitness(P_i)
        if Best = □ or Fitness(P_i) > Fitness(Best) then
            Best ← P_i
        end
    Q ← {the n fittest individuals in P, breaking ties at random}
    for (popsize - n)/2 times do
        Parent P_a ← SelectWithReplacement(P)
        Parent P_b ← SelectWithReplacement(P)
        Children C_a, C_b ← Crossover(Copy(P_a), Copy(P_b))
        Q ← Q ∪ {Mutate(C_a), Mutate(C_b)}
    P ← Q
return Best
```

2.4.4 GA for Data Augmentation

In the context of imbalanced classification, GAs can be effectively employed to generate synthetic data points that augment the minority class. In this application, each candidate solution is a synthetic data point represented as a feature vector $\mathbf{x} \in \mathbb{R}^n$. The objective is to evolve a population of synthetic instances that, when combined with the original training data, improve the overall performance of classifiers.

The process begins with the generation of an initial population by sampling uniformly from the observed feature bounds. The fitness function $F(\mathbf{x})$ is defined to capture multiple desirable properties:

$$F(\mathbf{x}) = w_1 \cdot \text{ClassifierScore}(\mathbf{x}) + w_2 \cdot \text{Diversity}(\mathbf{x}, P) - w_3 \cdot \text{Penalty}(\mathbf{x}),$$

where:

- ClassifierScore($\mathbf{x}$) quantifies the contribution of $\mathbf{x}$ to improving classification performance,
- Diversity($\mathbf{x}, P$) measures the dissimilarity of $\mathbf{x}$ relative to other synthetic instances in the population P,
- Penalty($\mathbf{x}$) imposes a cost for generating instances that deviate from desired statistical properties,
- w_1, w_2, and w_3 are weighting factors balancing these components.

Standard genetic operators such as crossover and mutation are then applied to evolve the population until the fitness converges or a set number of generations is reached [28, 46].

2.4.5 GA Data Augmentation Guided by Domain Constraints

An important enhancement to GA-based data augmentation is the integration of domain constraints into the fitness evaluation. Domain constraints ensure that the synthetic data adhere to known properties of the data, such as non-negativity of age or specified ranges for numerical features.

To enforce domain constraints, the fitness function is modified to include a penalty term for constraint violations:

$$F_{\text{constrained}}(\mathbf{x}) = F(\mathbf{x}) - \lambda \cdot C(\mathbf{x}),$$

where $C(\mathbf{x})$ quantifies the degree of constraint violation and λ is a scaling parameter that adjusts the impact of the penalty. For example, if a feature x_i must satisfy $x_i \geq 0$, then a simple penalty function for that feature is:

$$C_i(x_i) = \max(0, -x_i).$$

The overall penalty is given by:

$$C(\mathbf{x}) = \sum_{i=1}^{n} C_i(x_i).$$

Furthermore, adaptive mutation strategies can be employed to help maintain compliance with domain constraints. Specifically, the mutation magnitude for a gene x_i may be defined as:

$$\delta_i = \gamma \cdot \left(\frac{C_i(x_i)}{x_i + \epsilon} \right),$$

where γ is a scaling constant and ϵ is a small constant to avoid division by zero. This formulation increases the mutation step for genes that violate the constraints, thereby nudging the synthetic data toward compliance with the domain-specific rules.

By embedding these domain constraints directly into the GA's fitness function and mutation operator, the algorithm generates synthetic data that not only improve class balance but also maintain the essential properties of the original dataset. This approach results in higher-quality synthetic data that enhance downstream classifier performance [14] [56].

2.5 Counterfactual Data Augmentation

Counterfactual data augmentation takes advantage of **minimal feature changes** that flip a label from majority to minority [66]. Typically used for eXplainable AI, counterfactual pairs $(\mathbf{x}, \mathbf{p})$ reveal how a small difference in certain features can yield a different outcome. In tabular classification, one can systematically harness these pairs as "recipes" for generating new minority examples near the real boundary.

2.5.1 Mechanics

If we have a known pair $\mathbf{x} \to \mathbf{p}$ (majority to minority), and a separate majority instance $\mathbf{x}'$, we can produce a new minority instance $\mathbf{p}'$ by taking the difference-features from $\mathbf{p}$ and the match-features from $\mathbf{x}'$. Formally, if $\{\Delta\}$ is the feature subset that differs in the original pair, then:

$$p'_j = \begin{cases} p_j, & \text{if } j \in \{\Delta\}, \\ x'_j, & \text{otherwise.} \end{cases}$$

We thus anchor new synthetic points in real minimal changes. This approach has proven beneficial in certain real-world tasks where boundary coverage matters [66].

2.6 Metrics for Evaluating Augmented Data

One of the major gaps in the literature is the lack of standardized metrics for measuring the quality and utility of augmented data [70]. Below, we present common **intrinsic** (distributional similarity, fidelity, diversity) and **task-based** (classification performance, fairness, interpretability) metrics.

2.6.1 Intrinsic Metrics

Statistical Distance: Jensen-Shannon divergence or Wasserstein distance measure how closely the augmented distribution approximates the real. Let $\mathcal{D}_{real}$ be the real distribution, $\mathcal{D}_{aug}$ the augmented. A high JS or Wasserstein suggests the synthetic set might deviate significantly, potentially introducing unrealistic samples [71]. By contrast, a small distance indicates fidelity.

Coverage / Support: Coverage tracks how thoroughly synthetic data fill gaps in the minority region [28]. For instance, one might partition the minority feature space into cells or measure density-based coverage. High coverage but minimal overlap with the majority subspace is usually desirable. Low coverage suggests partial expansions; excessive coverage can cause borderline confusion.

Mode Count: In generative models, counting or estimating the number of modes detects possible "mode collapse" or incomplete coverage [44]. If real minority data have k clusters but the synthetic data produce fewer than k, one cluster remains under-sampled.

2.6.2 Task-Based Metrics

Classifier Performance: - **F1-score or G-mean** weigh both precision and recall [25] [6]. Specifically,

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}},$$

while G-mean is $\sqrt{\text{TPR} \times \text{TNR}}$. Both reflect minority detection without ignoring the majority. - **AUROC / AUPRC**: Threshold-insensitive measures capturing how well the model ranks minority above majority. The Area Under the ROC Curve (AUROC) focuses on TPR-FPR, whereas AUPRC highlights precision-recall trade-offs in heavily skewed data [73].

Fairness / Bias Measures: One may apply fairness metrics—**demographic parity, equalized odds**—if synthetic data must avoid amplifying bias [49]. If the new samples replicate or magnify existing biases, these metrics can drop. Checking fairness is critical in domains with legally or ethically protected features [71].

Explainability Scores: Some authors track how local or global model explanations (e.g., via LIME, SHAP) shift when augmented data are introduced [66]; [39]. If the synthetic set drastically alters feature attributions for the same predictions, augmentation might have introduced spurious patterns. A stable or modest shift suggests the augmentation is consistent with prior model logic [28].

Robustness Metrics:

- *Sensitivity to hyperparameters*: e.g., if a small tweak in the SMOTE neighborhood size or a GA mutation rate drastically changes results, the approach may be fragile [73].
- *Stress tests*: One can introduce random noise or adversarial perturbations to see if the augmented model remains stable [49]. Major performance drops might indicate that the synthetic data introduced borderline or overfitted expansions.

2.7 The DPG Algorithm and XAI Metrics

Explainable Artificial Intelligence (XAI) has witnessed rapid growth over the past few years as AI systems become increasingly complex and opaque. Numerous state-of-the-art methods—ranging from model-agnostic approaches such as LIME and SHAP to more model-specific techniques—have been developed to reveal the inner workings of machine learning models [1] [7]. These methods not only enhance user trust but also fulfill emerging regulatory requirements for transparency in critical applications such as healthcare,

Table 2.6.1: Major Augmentation Methods, Advantages and Limitations

Method	Typical Domain	Advantages	Limitations
		Interpolation-Based	
SMOTE (Chawla et al., 2002)	Tabular, time-series	Baseline for class imbalance; Simple linear interpolation; Widely recognized	May create borderline/overlapping noise; Does not handle outliers well
Borderline-SMOTE (Han et al., 2005)	Tabular, time-series	Focus on minority boundary; Improves recall for "risky" edge areas	Dependent on borderline detection; Risk of ambiguous expansions if overlap is severe
SMOTE-ENN (Batista et al., 2004)	Tabular	Cleans potential overlap post-SMOTE; Removes mislabeled examples	Can remove legitimate borderline points; Parameter tuning is domain-specific
SVM-SMOTE (Farquad & Bose, 2012)	Tabular	Uses SVM margin for oversampling; Strong synergy with boundary classification	Needs robust SVM training; Potentially computationally heavy
MSMOTE (Hu et al., 2009)	Tabular	Distinguishes safe/borderline/outliers; Reduces noise from outliers	Needs local density thresholds; Complex for very high-dimensional data
Polynomial-Fitting SMOTE (Gazzah & Essoukri Ben Amara, 2008)	Tabular	Nonlinear expansions capturing polynomial shapes; Avoids purely linear mixture	Can produce unnatural curves in small/noisy clusters; Requires polynomial degree selection
LVQ-SMOTE (Nakamura et al., 2013)	Tabular	Learns LVQ codebooks; Potentially robust to outliers in a cluster-based sense	Dependent on effective LVQ training; Less general if data do not cluster well
DE_oversampling (Chen et al., 2010)	Tabular	Leverages Differential Evolution; Explores wide solution space with mutation/crossover	Requires tuning (F, CR, population size); Risk of invalid samples if constraints not well-defined
		Generative-Based	
GAN (Margeloiu et al., 2024; Alauthman et al., 2023)	Images, text, tabular	Learns complex data distributions; Potentially realistic synthetic data	Training instability (mode collapse); Needs large data to train
VAE (El Emam et al., 2020)	Tabular (moderate dimension)	Stable training vs. GAN; Latent space allows continuous sampling	Tends to produce "averaged" samples; May miss minority-tail extremes if data are limited
Diffusion Models (Margeloiu et al., 2024; Wang et al., 2024)	Images, partial tabular	State-of-the-art distribution coverage; Less prone to mode collapse	Computationally heavy; Tabular usage is emergent, not widely proven
		Evolutionary	
Genetic Algorithm (GA) (Holland, 1975; Hu, 2025)	Tabular, time-series	Flexible domain constraints; Nonlinear transformations; Adaptive to hierarchical tasks	Potentially slow for large dimension; High chance of invalid points if constraints lacking
		Counterfactual / XAI-Based	
Counterfactual (Temraz & Keane, 2023)	Tabular	Minimal feature changes from majority→minority; Anchored in known boundary transitions	Depends on existing "native" counterfactual pairs; If pairs are rare, fewer new minority samples are generated

finance, and autonomous systems.
Despite the availability of various XAI techniques, there remains a strong need for methods that capture the inherent decision-making process of ensemble models in a holistic and interpretable manner. In this context, recent advances focus on converting the decision structure of complex models into explicit graph representations. One such approach is the Decision Predicate Graph (DPG) algorithm introduced by Arrighi L. and Barbon Junior S. in 2024 , which offers a model-agnost[5]ic, global interpretation of tree-based ensemble models by revealing the relationships among decision predicates.

2.7.1 The DPG Algorithm

The DPG algorithm transforms a tree-based ensemble model into a graph structure $G = (V, E)$. In this graph, each node $v \in V$ represents a predicate (i.e., a feature-value condition) extracted from the internal nodes of decision trees, while each edge $e = (v_i, v_j) \in E$ indicates that the predicates v_i and v_j co-occur in a decision path. Formally, if a decision tree T provides a sequence of predicates $P = \{p_1, p_2, \ldots, p_k\}$ for a given prediction, then for each consecutive pair (p_i, p_{i+1}) an edge is created with an associated weight $w(e)$ reflecting the frequency of such a transition across the ensemble:

$$w(e) = \frac{\text{Number of times } (p_i, p_{i+1}) \text{ occurs in } T}{\text{Total number of paths in } T}.$$

By aggregating these frequencies over all trees in the ensemble, the DPG provides a comprehensive picture of the model's decision logic [5].
A key DPG parameter that needs to be adjusted is the **pv <threshold_value>**. The parameter `pv` represents a threshold indicating the minimum proportion of ensemble trees in which a specific decision path must appear to be included in the Decision Predicate Graph (DPG). By adjusting `pv`, users can filter out less relevant or less frequent decision paths, **enhancing the clarity and interpretability of the resulting explanations provided by the DPG algorithm**.

2.7.2 XAI Metrics for the DPG Algorithm

Once the DPG is constructed, various XAI metrics can be computed on the graph to quantify the importance and influence of individual predicates. For example, the **Local Reaching Centrality (LRC** of a node v is defined as:

$$C_D(v) = \deg(v),$$

which measures the number of edges incident on v. A higher degree suggests that the predicate is frequently used in the decision process. The LRC serves as a metric for as-

sessing the importance of DPG's nodes. It gauges the extent to which decisions contained in these nodes are employed by diverse tree base learners for classifying samples in the training set. Additionally, the prominence of paths between highlighted predicates indicates their frequent utilisation, providing insights into how new samples can be classified with fewer decisions.
Another useful metric is the **betweenness centrality**, which reflects the extent to which a node lies on the shortest paths between other nodes:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

where σ_{st} is the total number of shortest paths from node s to node t, and $\sigma_{st}(v)$ is the number of those paths that pass through v. This metric helps identify predicates that serve as critical bridges within the decision structure. The decision contained in the node is essential to classify the elements of the dataset.
Lastly, **Communities** within the DPG offer another layer of interpretability. A community is defined as a subset of nodes characterised by dense interconnections among themselves and sparse connections with nodes outside the community. Communities are identified using the asynchronous label propagation algorithm, where each node determines its membership based on the majority of its neighbours' labels. Nodes sharing labels are grouped into communities, indicating common classification characteristics. Communities help identify groups of predicates that significantly contribute to classifying samples into specific classes, thus elucidating how certain features and associated values consistently influence model predictions. Furthermore, communities reveal insights into the complexity of classifying samples, indicating which classes are more distinguishable or challenging based on the diversity and density of predicates involved.

2.7.3 DPG Constraints

Constraints provide additional interpretability by defining intervals associated with features necessary for class assignment. Constraints are computed by considering all predicates connected by paths originating from nodes and culminating in a specific class. For each identified class in the DPG, we list all nodes connected by a path that culminates in the class node. For each feature within these node predicates, we delineate the most extensive possible interval using the values associated with the features. Formally, given a class node c, the constraint for each feature f is defined as the interval:

$$[f_{min}, f_{max}],$$

where f_{min} and f_{max} represent the smallest and largest feature values among predicates along all paths culminating at class c. If no upper or lower boundary for the feature

is explicitly given by predicates, the corresponding interval boundary corresponds to the maximum or minimum value of the feature. These intervals clearly delineate the ranges of feature values that determine class membership according to the ensemble's decision logic. Thus, constraints calculate the classification boundary values of each feature associated with each class, providing clear guidelines on sample classification and facilitating model validation and interpretation.

2.8 Interpreting Ensemble Classifier with DPG Algorithm

To illustrate how the Decision Predicate Graph (DPG) algorithm enhances the explainability of ensemble models, we analyze a concrete example using the *Mammographic Mass* dataset (Table 1) [41]. This dataset, used for breast cancer classification, includes predictive attributes such as patient age, BI-RADS assessment, shape, margin, and density of mammographic lesions to determine their severity as benign or malignant.
After training a Random Forest classifier on this dataset, the DPG algorithm (pv=0.06) was employed to create an interpretable representation of the ensemble model's decisions. The resulting graph highlights predicates and decision paths associated with each classification class, enabling the identification of feature conditions crucial for classifying lesions.
With the Random Forest classifier, we obtained a Hold-Out test accuracy of 0.84, an F1-score for class 0 (benign) of 0.85, and an F1-score for class 1 (malignant) of 0.84.

2.8.1 XAI Metrics for the DPG Algorithm

Once the DPG is constructed, XAI metrics are computed, clearly highlighting the decision structure utilized by the Random Forest classifier. The DPG visualization and metrics applied to the Mammographic Mass dataset are as follows:

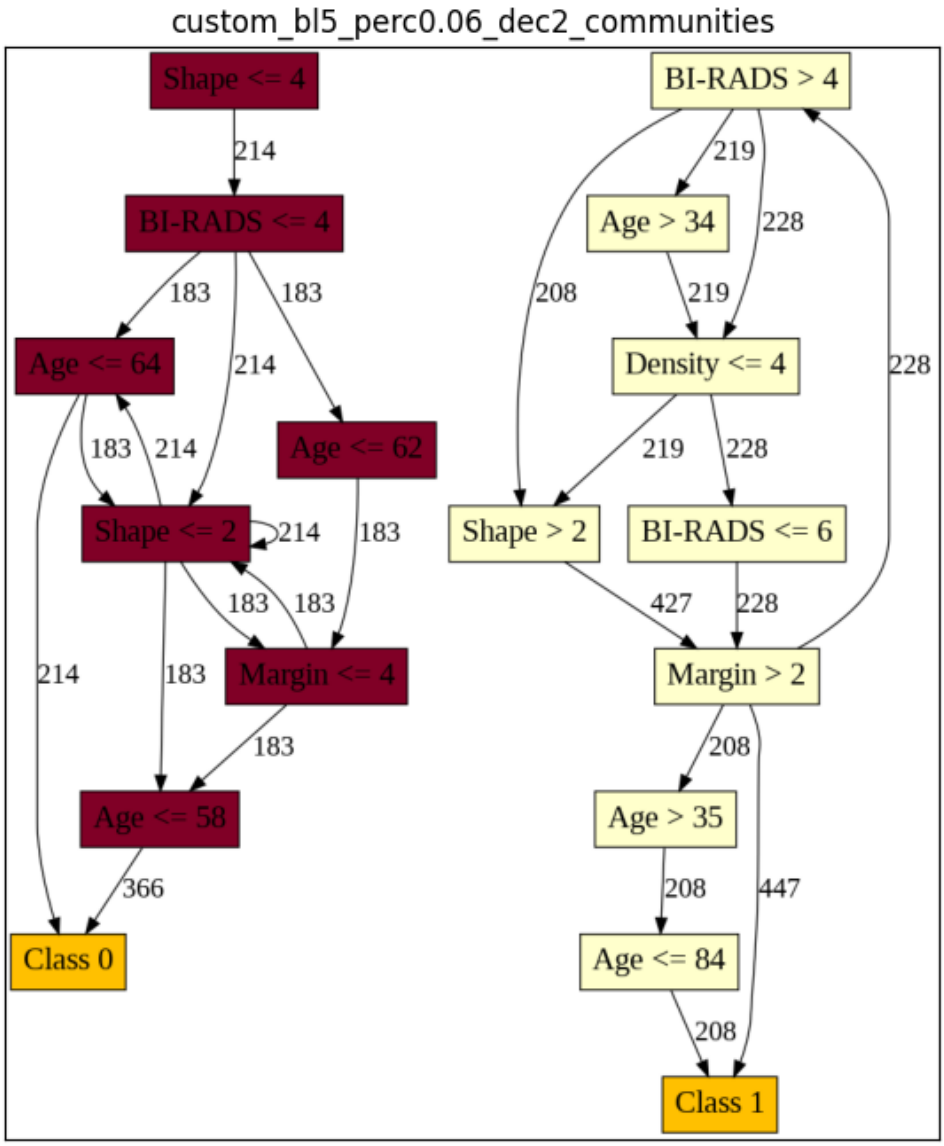

Figure 2.8.2: DPG Communities for Mammographic Mass Dataset

Communities reveal two well-defined groups (see Figure 2.8.2). One community associated exclusively with benign instances (Class 0) contains predicates *Age <= 64*, *Margin <= 4*, *Shape <= 4*, and *BI-RADS <= 4*, indicating these attributes strongly define benign conditions. Conversely, malignant cases (Class 1) form another distinct community characterized by predicates *BI-RADS between 4 and 6*, *Age between 34 and 84*, *Margin > 2*, *Shape > 2*, and *Density <= 4*.

These communities provide intuitive **clinical interpretations**. Benign masses commonly show more regular shapes, lower BI-RADS scores (<=4), and younger age ranges. Conversely, malignant masses frequently show irregular margins, higher BI-RADS scores (>4), and higher density characteristics. This community partitioning simplifies interpretability by explicitly clustering feature interactions significant for each diagnostic class.

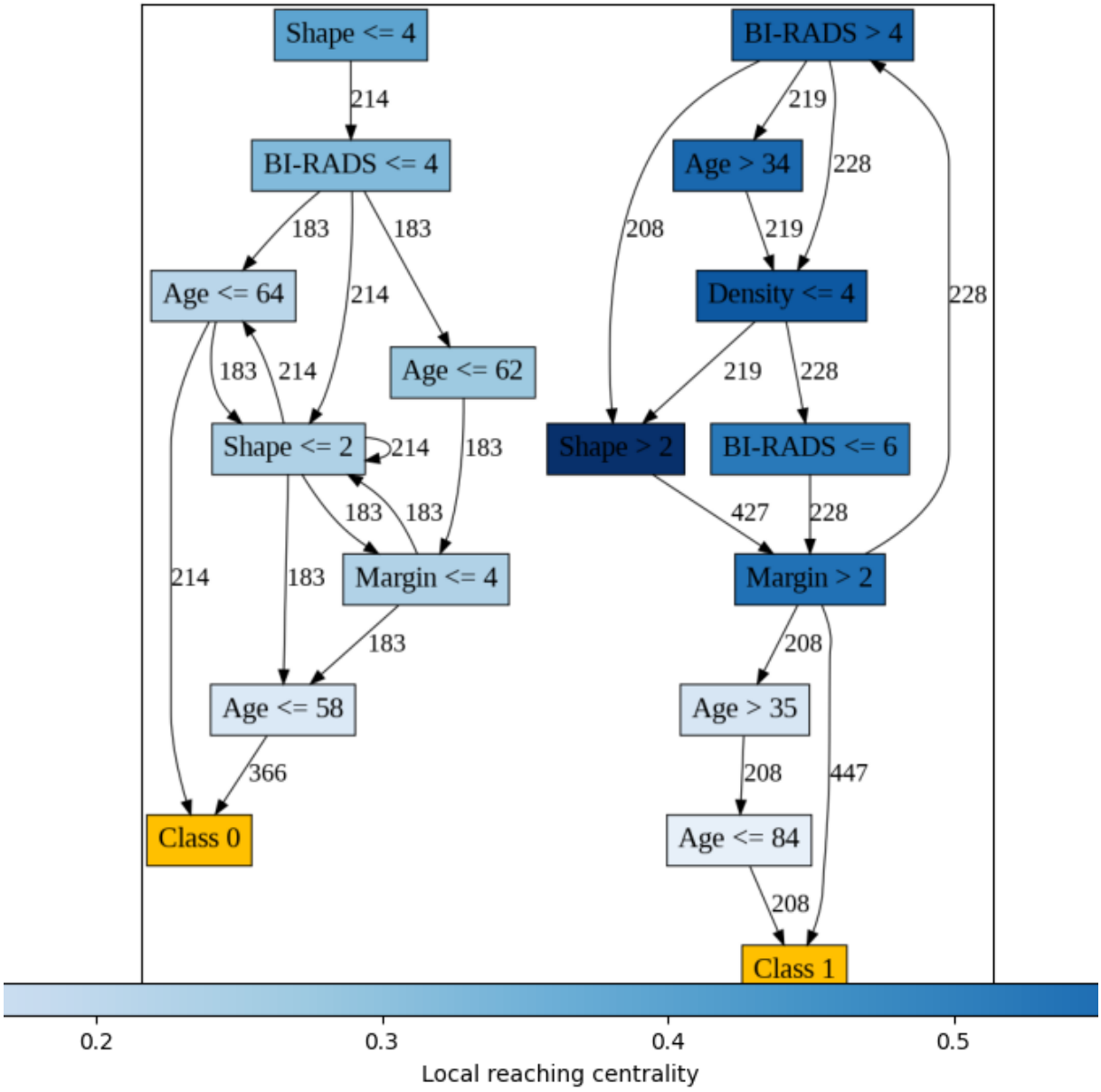

Figure 2.8.3: Local Reaching Centrality for Mammographic Mass Dataset

The **Local Reaching Centrality (LRC)** (Figure 2.8.3) metric quantifies the frequency and importance of predicates in classifying instances. Predicates like *"BI-RADS > 4"*, *"Margin > 2"*, and *"Density 4"* exhibit high centrality values (darker blue nodes in Figure 2), indicating their critical role in classifying lesions as malignant (Class 1). Conversely, predicates such as *"BI-RADS 4"* and *"Shape 2"* show significant centrality for benign lesion identification. Thus, predicates with high local reaching centrality serve as **strong diagnostic indicators** within the ensemble's logic.

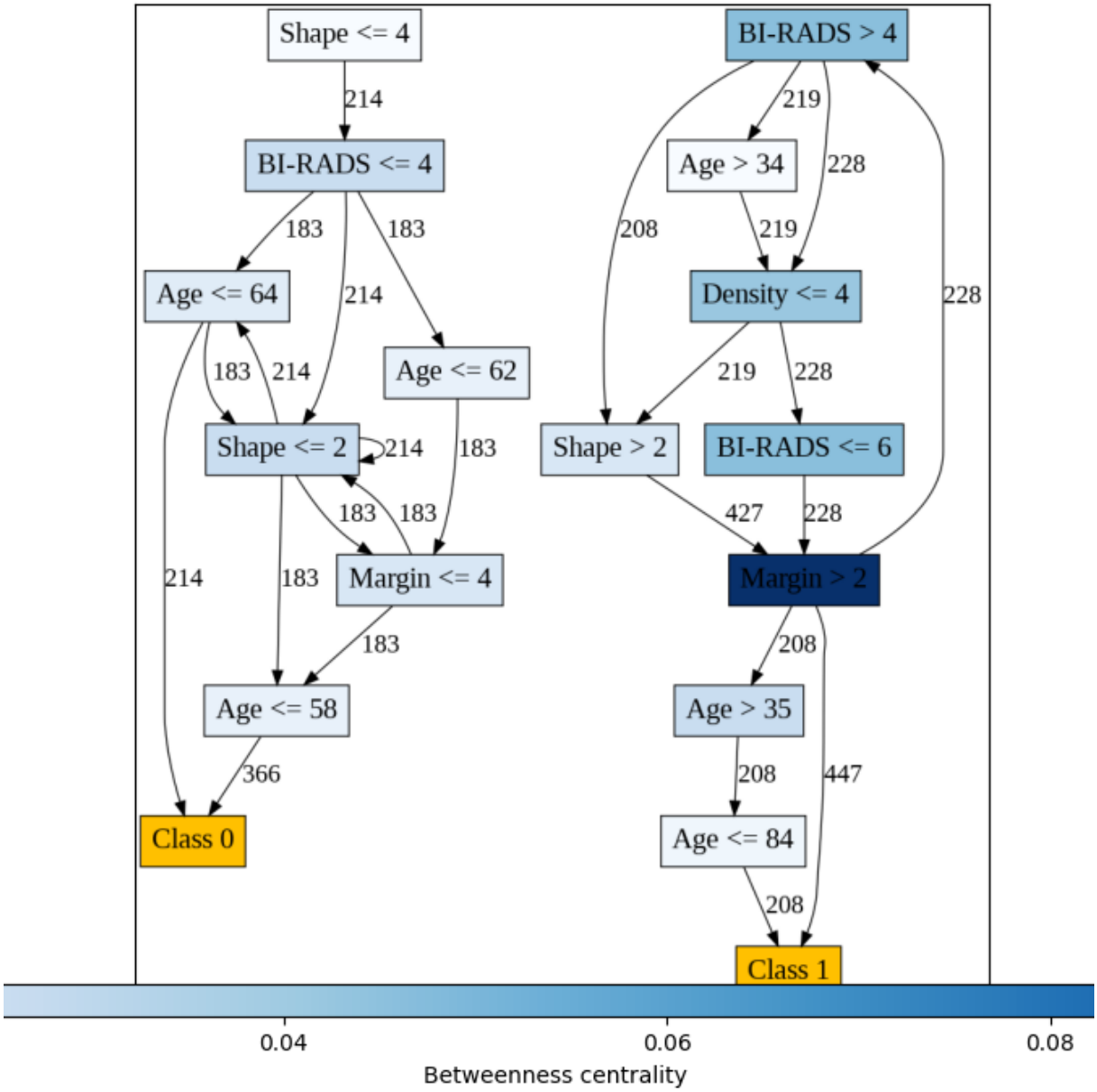

Figure 2.8.4: Betweenness Centrality for Mammographic Mass Dataset

The **Betweenness Centrality** (Figure 2.8.4) metric further enhances the analysis by highlighting predicates that act as critical bridges between decision paths. For instance, in the given DPG visualization, the predicate *"Margin > 2"* emerges with particularly high betweenness centrality, suggesting it is a **decisive boundary feature frequently determining lesion severity**. Therefore, predicates with higher **betweenness centrality should be prioritized for clinical interpretation**, as they critically influence the ensemble classification decision flow.

2.8.2 DPG Constraints

The **Constraints** derived from the DPG provide interpretable feature intervals associated with each class:

- Class 0 (benign): *Age <= 64*, *Margin <= 4*, *Shape <= 4*, *BI-RADS <= 4*.
- Class 1 (malignant): *BI-RADS between 4 and 6*, *Age between 34 and 84*, *Margin > 2*, *Shape > 2*, *Density <= 4*.

These intervals precisely delineate how Random Forest classification rules partition the feature space, offering clear insights into clinical decision thresholds. Such constraints enable practitioners to understand exactly under what clinical conditions predictions of benign or malignant masses are made, thereby **significantly improving model transparency and clinical applicability**.

Chapter 3

DPG Data Augmentation Algorithm

DPG algorithms in particular represent ensemble model logic in a graph form, capturing relationships between predictive features, thresholds, and resulting classifications (Arrighi et al., 2024). This graph-based interpretation offers clear visualizations of the decision-making processes within ensembles, making it easier for stakeholders to understand how input features influence model decisions. Leveraging such clarity, we propose a data augmentation algorithm guided explicitly by constraints derived from a trained DPG model. The DPG augmentation algorithm leverages these extracted predicates (DPG constraints) as guidelines for generating synthetic data points. The underlying hypothesis is that new synthetic samples generated under these constraints maintain logical consistency and reduce the risk of producing invalid data, addressing some major pitfalls of traditional data augmentation methods.

3.1 Properties and Guarantees

3.1.1 Definitions

Definition 3.1.1 (Feasible Region (DPG Constraints))**.** Let $\mathcal{F}$ denote the feature space, and $\mathcal{C}_k \subseteq \mathcal{F}$ be the feasible region for class k:

$$\mathcal{C}_k = \left\{ \mathbf{x} \in \mathcal{F} \,\middle|\, \forall f_i \in \text{features}, \, l_{i,k} \leq x_i \leq u_{i,k} \right\}$$

where $l_{i,k}, u_{i,k}$ are lower/upper bounds for feature f_i in class k, derived from the DPG.

Definition 3.1.2 (Genetic Algorithm Operators)**.**

- Mutation: Stochastic operator $M : \mathcal{C}_k \to \mathcal{C}_k$ where:

$$M(\mathbf{x}) = \mathbf{x}' \text{ s.t. } \mathbf{x}' \in \mathcal{C}_k$$

- Crossover: Recombination operator $C : \mathcal{C}_k \times \mathcal{C}_k \to \mathcal{C}_k$:

$$C(\mathbf{p}_1, \mathbf{p}_2) = \{\mathbf{c}_1, \mathbf{c}_2\} \subseteq \mathcal{C}_k$$

- Fitness Function: $\phi : \mathcal{C}_k \to \mathbb{R}^+$ guiding sample quality

3.1.2 Formal Evaluation Metrics

Given augmented dataset $\mathcal{D}_{\text{aug}} = \mathcal{D}_{\text{orig}} \cup \mathcal{D}_{\text{synth}}$, we define:

$$\Delta\text{F1} \triangleq \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{test}}} [\mathbb{I}(h_{\text{aug}}(\mathbf{x}) = y) - \mathbb{I}(h_{\text{orig}}(\mathbf{x}) = y)] \tag{3.1}$$

$$\nu \triangleq \frac{1}{|\mathcal{D}_{\text{synth}}|} \sum_{\mathbf{x}\in\mathcal{D}_{\text{synth}}} \mathbb{I}(\mathbf{x} \notin \mathcal{C}_k) \tag{3.2}$$

Interpretation:

- ΔF1 quantifies the classifier's performance improvement using the expectation operator $\mathbb{E}$ over test samples
- ν measures constraint violation rate through indicator function $\mathbb{I}$, with $\nu = 0$ indicating perfect adherence

3.2 Algorithm Design

3.2.1 DPG class bounds (constraints)

The feasible region $\mathcal{C}_k$ is derived through systematic analysis of all decision paths in the Decision Predicate Graph (DPG). Each path p represents a conjunction of feature constraints leading to class k:

$$\mathcal{C}_k = \bigcap_{p\in\text{Paths}(k)} \left\{ \mathbf{x} \,\middle|\, \bigwedge_{(f,o,t)\in p} (f(\mathbf{x})\, o\, t) \right\} \tag{3.3}$$

Key Components:

- Paths(k): All decision paths terminating in class k in the DPG
- Predicate (f, o, t): Feature f, comparison operator $o \in \{\leq, >, =, \neq\}$, and threshold t
- $\bigwedge$: Logical AND operation combining all path constraints
- $\bigcap$: Set intersection enforcing simultaneous satisfaction of all path constraints

Example: For a medical diagnosis DPG, paths to "Malignant" might require:

$$(\text{Age} \geq 40) \wedge (\text{MassSize} > 2.5\text{cm}) \wedge (\text{Margin} \in \{2, 4\})$$

3.2.2 Genetic Algorithm Framework

Population Initialization

The initial population is generated by uniform sampling within DPG-derived bounds:

$$x_i \sim \mathcal{U}(l_{i,k}, u_{i,k}) \quad \forall f_i \in \text{features} \tag{3.4}$$

Process Details:

- $l_{i,k}, u_{i,k}$: Lower/upper bounds from DPG analysis
- Ensures initial candidates satisfy all feature constraints
- Maintains interpretability by respecting original model's decision logic

Mutation Operator

Controlled perturbation maintains feasibility while exploring local neighborhoods:

$$x_i' = \text{clip}\left(x_i + \Delta,\ l_{i,k},\ u_{i,k}\right),\ \Delta \sim \mathcal{U}(-0.05r, 0.05r) \tag{3.5}$$

Where:

- $r = u_{i,k} - l_{i,k}$: Feature's valid range
- Δ: Small random perturbation (5 percent of feature range)
- clip(): Ensures mutated values stay within DPG bounds

Crossover Operator

Single-point recombination combines parental genetic material:

$$\mathbf{c}_1 = (p_{1,1}, \ldots, p_{1,j}, p_{2,j+1}, \ldots, p_{2,n}) \tag{3.6}$$

$$\mathbf{c}_2 = (p_{2,1}, \ldots, p_{2,j}, p_{1,j+1}, \ldots, p_{1,n}) \tag{3.7}$$

Properties:

- j: Randomly selected crossover point ($1 \leq \text{j} < \text{n}$)
- Preserves feasibility as children inherit parent constraints
- Promotes exploration of constraint space combinations

3.2.3 Fitness Functions

Original Mode: Enforces strict adherence to DPG constraints and correct classification:

$$\phi_{\text{orig}}(\mathbf{x}) = \mathbb{I}(h(\mathbf{x}) = k) + 0.25 \sum_{i=1}^{n} \mathbb{I}(x_i \in [l_{i,k}, u_{i,k}]) - \sum_{i=1}^{n} \mathbb{I}(x_i \notin [l_{i,k}, u_{i,k}]) \tag{3.8}$$

Term Analysis:

- $\mathbb{I}(h(\mathbf{x}) = k)$: Primary reward for correct classification
- $0.25 \sum \mathbb{I}(\cdot)$: Partial reward per satisfied constraint
- $-1.0 \sum \mathbb{I}(\cdot)$: Penalty for constraint violations.

Border Mode: Optimizes samples near decision boundaries while maintaining diversity:

$$\phi_{\text{border}}(\mathbf{x}) = 5.0 p_k(\mathbf{x}) + 3.0(1 - |p_k(\mathbf{x}) - p_j(\mathbf{x})|) + \lambda_{\text{div}} d_{\text{mean}} + \lambda_{\text{repel}} d_{\text{min}} \tag{3.9}$$

Component Breakdown:

- $5.0 p_k$: Strong reward for high target class probability
- $3.0(1 - |p_k - p_j|)$: Boundary proximity incentive (p_j = nearest competitor class)
- $\lambda_{\text{div}} d_{\text{mean}}$: Diversity promotion via average pairwise distance
- $\lambda_{\text{repel}} d_{\text{min}}$: Anti-clustering through minimum distance penalty

Original Mode: Vectorized Numpy Arrays

- **Efficiency**: Numpy arrays enable vectorized operations, allowing batch computations (e.g., mutations, fitness evaluations) on the entire population without explicit loops. This is critical for speed when working with large datasets or simple, uniform constraints.
- **Uniform Constraints**: Original mode generates samples strictly within predefined DPG constraints. Since all features follow uniform rules (e.g., Age $\in [18, 96]$), numpy's array operations simplify enforcing these constraints across the population.
- **Simpler Fitness Calculation**: The fitness function in original mode primarily checks:

  1. Whether a sample is classified as the target class
  2. Whether it adheres to DPG constraints

  These checks are easily vectorized using numpy's boolean indexing.

Border Mode: Lists of Individuals

- **Complex, Individualized Operations**: Border mode requires nuanced computations that vary per individual, such as:

- Boundary Proximity: Calculating distance to decision boundaries (e.g., using model probabilities)
- Diversity Maintenance: Ensuring samples are spread out (e.g., via pairwise distance checks)

Lists allow per-individual modifications and heterogeneous operations that are harder to vectorize.

- **Flexibility for Custom Logic**: Border mode's fitness function includes terms like:

```
fitness = 5.0 * p_target + 3.0 * boundary_score + diversity_bonus
```

Lists make handling these multi-component, state-dependent calculations easier for each individual.

- **Non-Uniform Adjustments**: Features near class boundaries may require different mutation strategies (e.g., larger perturbations for ambiguous features). Lists, unlike numpy's homogeneous array operations, allow fine-grained control over each feature's adjustment.

Example Workflow Comparison

Original Mode (Numpy):

1. Initialize population as a 2D numpy array
2. Vectorized mutation: Perturb all features simultaneously
3. Vectorized fitness: Check classification and constraints in bulk
4. Select parents via numpy indexing

Border Mode (Lists):

1. Initialize population as a list of individual lists
2. Iterate over each individual:
   - Mutate specific features based on boundary proximity
   - Calculate fitness using model probabilities and diversity metrics
3. Select parents via custom tournament logic

Why Not Use Lists/Numpy for Both?

- **Original Mode**: Lists would slow down simple operations (e.g., constraint checks) due to Python's loop overhead
- **Border Mode**: Numpy arrays would restrict flexibility for per-individual logic (e.g., diversity penalties)

3.2.4 Evolutionary Process

Algorithm 4 DPG-GA Evolutionary Optimization

Initialize population $\mathcal{P}^{(0)}$ with N_{pop} individuals
for $t = 1$ *to* N_{gen} **do**
    Evaluate $\phi(\mathbf{x}) \forall \mathbf{x} \in \mathcal{P}^{(t)}$ ▷ Fitness calculation
    Select parents via tournament selection ▷ Size-3 tournaments
    Apply crossover with $p_{\text{cross}} = 0.7$ ▷ 70% recombination rate
    Apply mutation with $p_{\text{mut}} = 0.3$ ▷ 30% mutation rate per gene
    Preserve elite individual ▷ Maintains best solution
    if $\max \phi^{(t)} - \max \phi^{(t-\tau)} < \epsilon$ **then**
        break ▷ Convergence detection ($\tau = 5$, $\epsilon = 0.001$)

3.2.5 Constraint Parsing

Algorithm 5 Constraint to Interval Conversion

procedure ParseConstraint(line)
    Tokenize line into $\{t_1, \ldots, t_m\}$ ▷ Split by whitespace
    $f \leftarrow$ non-numeric tokens joined with spaces ▷ Feature name reconstruction
    $b \leftarrow$ numeric tokens converted to floats ▷ Threshold extraction
    return $(f, (\min(b), \max(b)))$ ▷ Interval creation
end procedure

Example Transformation:

Input: "$30 < \text{Age} \leq 65$"

Output: ("Age", (30.0, 65.0))

3.2.6 Diversity Maintenance

`diversity_weight` vs. `repulsion_weight`

Core Definitions

- **`diversity_weight`** (α): Controls global spatial distribution of samples.
- **`repulsion_weight`** (β): Enforces minimum pairwise separation between samples.

Mechanisms

- **`diversity_weight`**:

$$\text{div\_bonus} = \alpha \cdot \frac{1}{M-1} \sum_{j \neq i} \|\mathbf{x}_i - \mathbf{x}_j\| \tag{3.10}$$

  where M = population size. Rewards *average distance* to all other samples.

- **`repulsion_weight`**:

$$\text{rep\_bonus} = \beta \cdot \min_{j \neq i} \|\mathbf{x}_i - \mathbf{x}_j\| \tag{3.11}$$

  Penalizes proximity to the *nearest neighbor*.

3.2.7 Operational Impacts

Aspect	**`diversity_weight`**	**`repulsion_weight`**
Scope	Global (population-wide distribution)	Local (pairwise separation)
Metric	Average distance to all samples	Minimum distance to any sample
Primary Role	Broad exploration of feature space	Prevention of clustering
Fitness Impact	Smooth gradient favoring spread	Sharp penalty for neighbor proximity

Practical Guidance

- **High α + Low β**: Samples spread widely but may form loose clusters.
- **Low α + High β**: Avoids clustering but limits coverage.
- **Balanced values**: Optimizes trade-off between global diversity and local uniqueness.

Summary

- `diversity_weight`: Governs *how broadly* samples explore the DPG-constrained space.
- `repulsion_weight`: Dictates *how tightly* samples cluster.

3.2.8 Boundary Proximity

Estimates decision boundary proximity using classifier confidence:

$$\text{BoundaryScore}(\mathbf{x}) = 1 - |p_k(\mathbf{x}) - \max_{j \neq k} p_j(\mathbf{x})| \tag{3.12}$$

Interpretation:

- p_k: Classifier's confidence for target class
- $\max p_j$: Highest non-target class confidence
- Score $\in [0, 1]$: 1 = exact boundary, 0 = clear classification

3.3 Complete Algorithm

Algorithm 6 Constraint Parsing, Initialization, and Fitness Evaluation

Input: Textual DPG constraints, trained model, target class, border class (optional), mode ("original" or "border"), feature ranges, population size, diversity weight, repulsion weight

Output: Validated feature ranges, initialized population, evaluated fitness

Function `Parse_Constraints(`*textual_constraints*`):`
- **for each** *line* $\in$ *textual_constraints* **do**
  - Extract feature_name, operator, bounds Update global_feature_bounds[feature_name]
- **return** validated_feature_ranges

Function `Initialize_Population(`*feature_ranges*`):`
- population $\leftarrow$ [] **for** $i = 1$ *to population_size* **do**
  - individual $\leftarrow$ [] **for each** *feature in feature_order* **do**
    - random_value $\sim U(feature\_ranges[feature].min, feature\_ranges[feature].max)$
    - individual.append(random_value)
  - population.append(individual)
- **return** population

Function `Evaluate_Fitness(`*individual, population*`):`
- **if** *mode = "original"* **then**
  - **if** *model.predict(individual)* $\neq$ *target_class* **then**
    - **return** $-\infty$
  - penalty $\leftarrow$ 0 **for each** *feature in individual* **do**
    - **if** *feature* $\notin$ *DPG_constraints* **then**
      - penalty $\leftarrow$ penalty +1.0
  - fitness $\leftarrow$ classification_confidence $-$ penalty
- **else** `// border mode`
  - $boundary\_score \leftarrow 1 - |P(target) - P(border\_class)|$ $diversity \leftarrow$ average_distance_to_others(individual, population) $repulsion \leftarrow$ min_distance_to_others(individual) fitness $\leftarrow (5 \cdot P(target)) + (3 \cdot boundary\_score) + (diversity\_weight \cdot diversity) + (repulsion\_weight \cdot repulsion)$
- **return** fitness

Algorithm 7 Mutation and Genetic Evolution Procedure

Input: DPG constraints, population size, mutation rate, crossover rate, generations, fitness evaluation function

Output: Augmented samples

Function `Mutate(`*individual*`):`

```
for each feature in feature_order do
    range ← 0.05 · (constraint.max − constraint.min)  Δ ∼ U(−range, range)
     individual[feature] ← Clamp(individual[feature] + Δ, constraint.min, constraint.max)
return individual
```

Procedure `Evolve():`

```
population ← Initialize_Population(DPG_constraints)  best_fitness ← −∞
 stagnation_counter ← 0
for generation = 1 to generations do
    parents ← Tournament_Selection(population, tournament_size=3)  offspring ←
     []
    for each (parent1, parent2) in parents do
        if rand() < crossover_rate then
            child1, child2 ← Uniform_Crossover(parent1, parent2)
        else
            child1, child2 ← parent1, parent2
        offspring.extend([child1, child2])
    for each individual in offspring do
        if rand() < mutation_rate then
            individual ← Mutate(individual)
    new_population ← [best_individual] + offspring  population ←
     new_population[0:population_size]
    if best_fitness not improved in last 5 generations then
        break
return filtered valid samples from population
```

3.4 Hyperparameters

The DPG-based data augmentation algorithm employs the following hyperparameters to control the evolutionary process. Their roles, valid ranges, and impacts are described below:

- **population_size** (integer ≥ 1): Number of candidate solutions in each generation. Larger values improve exploration but increase computational cost. Typical range: 50–500.
- **mutation_rate** (float $\in [0, 1]$): Probability of applying random perturbations to features. Higher rates increase diversity but risk destabilizing convergence. Typical range: 0.01–0.2.
- **crossover_rate** (float $\in [0, 1]$): Probability of combining parent solutions to create

offspring. Higher values promote genetic mixing but may slow optimization. Typical range: 0.6–0.9.

- **augmentation_mode** (string: `"original"` or `"border"`): Determines generation strategy. `"original"` creates samples within DPG constraints, while `"border"` focuses on class boundaries.
- **boundary_weight** (float ≥ 0; *border mode only*): Controls emphasis on proximity to class boundaries in fitness calculations. Higher values produce boundary-proximal samples. Typical range: 0.1–5.0.
- **diversity_weight** (float ≥ 0): Penalizes similarity between generated samples. Larger values increase spread but may reduce class-specific precision. Typical range: 0.1–2.0.
- **repulsion_weight** (float ≥ 0): Discourages clustering by penalizing small inter-sample distances. Higher values enforce spatial separation. Typical range: 0.1–1.0.
- **max_other_prob** (float $\in [0, 1]$; *border mode only*): Maximum allowable predicted probability for non-target/border classes. Lower values enforce stricter class separation. Typical range: 0.05–0.2.
- **re_inject_threshold** (float ≥ 0): Triggers population re-injection if diversity falls below this threshold. Lower values make re-injection more frequent. Typical range: 0.1–0.5.
- **re_inject_ratio** (float $\in [0, 1]$): Proportion of population replaced with random individuals during re-injection. Higher ratios reset exploration but discard progress. Typical range: 0.1–0.4.

Fixed Parameters (derived from DPG/model constraints):

- **class_bounds**: Feature intervals from DPG constraints (non-tunable)
- **feature_order**: Original feature names (non-tunable)
- **n_classes**, **target_class**, **border_class**: Dataset-specific constants

Adjusting these hyperparameters involves trade-offs between sample quality, diversity, and computational efficiency. For instance, increasing *diversity_weight* and *repulsion_weight* together enhances sample variety but may require compensation with larger *population_size* to maintain convergence.

3.5 Algorithm Analysis

We consider the following variables:

- M: Population size
- D: Number of features
- C: Model prediction cost per individual

- G: Number of generations
- S: Number of generations until early convergence, where $S \ll G$

3.5.1 Time Complexity

The algorithm's computational complexity depends on its operational mode (`original` or `border`):

- **Original Mode**:
  - *Fitness Evaluation*: $O(M \cdot C)$, where M = population size and C = model prediction cost (tree traversals in Random Forest)
  - *Mutation/Crossover*: $O(M \cdot D)$, where D = number of features
  - *Per Generation*: $O(M \cdot (C + D))$
  - *Total for G Generations*: $O(G \cdot M \cdot (C + D))$
- **Border Mode**:
  - *Fitness Evaluation*: $O(M^2 \cdot D)$ (due to pairwise diversity calculations)
  - *Mutation/Crossover*: $O(M \cdot D)$
  - *Per Generation*: $O(M^2 \cdot D)$
  - *Total for G Generations*: $O(G \cdot M^2 \cdot D)$

3.5.2 Space Complexity

- **Population Storage**: $O(M \cdot D)$
- **Fitness Values**: $O(M)$
- **Boundary Points (Border Mode)**: $O(B \cdot D)$, where B = number of boundary samples
- *Total*: $O(M \cdot D + B \cdot D)$

3.5.3 Asymptotic Analysis

- **Best Case** (Early Convergence):
  - Original Mode: $O(S \cdot M \cdot (C + D))$
  - Border Mode: $O(S \cdot M^2 \cdot D)$
- **Average Case**:
  - Original Mode: $O(G \cdot M \cdot C)$ (dominant term)
  - Border Mode: $O(G \cdot M^2 \cdot D)$
- **Worst Case** (No Early Stopping):
  - Matches total complexity for G generations

3.5.4 Order of Growth

- **Original Mode**: Linear in M and G, sensitive to model complexity C
- **Border Mode**: Quadratic in M, linear in G and D

3.5.5 Key Observations

Bottlenecks:

- Border mode's pairwise diversity calculations (M^2 term)
- Model prediction cost in original mode (C term)

The analysis highlights fundamental trade-offs between sample quality (enforced by DPG constraints) and computational efficiency, particularly in border mode where quadratic scaling limits scalability for large populations.

Chapter 4

Experimental results

In this chapter, we present the experimental results of our study, which aim to evaluate the effectiveness of the Decision Predicate Graph (DPG)-based data augmentation algorithm in improving classifier performance while preserving domain-specific logical constraints. Our experiments focus on two primary objectives:

1. Assessing the impact of DPG augmentation on model performance across multiple classifiers, with particular attention to mitigating class imbalance and refining decision boundaries.

2. Validating adherence to domain constraints to ensure synthetic data respects inherent dataset logic (e.g., non-negative ages, clinically plausible ranges).

To achieve this, we implemented a dual-mode DPG augmentation strategy across 4 real-world and 6 synthetic datasets spanning healthcare, finance, and industrial applications. These datasets were selected to evaluate the robustness of our method in diverse scenarios with varying feature types, class imbalances, and domain-specific constraints:

- Original Mode: Generates synthetic samples strictly within DPG-derived feature constraints.
- Border Mode: Creates data points near class decision boundaries while maintaining constraint compliance.

We evaluated these approaches using three distinct classifiers Decision Trees (DT), Logistic Regression (LR), and k-Nearest Neighbors (kNN)) measuring performance improvements through the F1 score, a critical metric for assessing bias mitigation in imbalanced classification scenarios.
To ensure real-world plausibility, we quantified the logical validity of synthetic samples by defining domain-specific constraints. For instance:

- *Healthcare*: Age ≥ 0, biologically feasible lab value ranges
- *Finance*: Transaction amounts within observed min/max bounds

- *Industry*: Sensor readings adhering to operational thresholds

Violation rates were calculated to verify that synthetic data respects these constraints, ensuring that augmented datasets retain practical utility.

Finally, to contextualize our results, we benchmarked the DPG augmentation against SMOTE and its variants, a widely used oversampling technique. This comparison highlights the unique advantages of constraint-aware synthetic data generation in preserving dataset integrity while addressing class imbalance, particularly in sensitive domains like healthcare and finance, where logical consistency is paramount.

The following sections detail our experimental setup, comparative analyses across diverse datasets, and findings, providing insights into the trade-offs between augmentation strategies and their implications for interpretable machine learning.

4.1 Datasets Description

This study evaluates the proposed DPG-based augmentation method using 10 datasets—4 real-world and 6 synthetic—spanning diverse domains such as healthcare, finance, manufacturing, and energy (Table 1). These datasets were selected to assess performance under varying conditions of class imbalance, feature complexity, and domain-specific constraints. A comparison with SMOTE and its variants further validates the robustness of our approach.

Table 4.1.1: Datasets Overview with Bibliographic References and Class Counts

Dataset	**Domain**	**Features**	**Instances**	**Imbalance ratio**	**Reference**
Real-World					
Mammographic Mass	Healthcare	BI-RADS, Age, Shape, Margin, Density	961	427:403	[UCI, 2023]
Pima Indians Diabetes	Healthcare	Gluc., BMI, Age, Ins., Preg., SkinT., BloodP. DiabPF.	768	500:268	[Kaggle, 2021]
Australian Credit	Finance	A1-A14 (14 features).	690	383:307	[UCI, 2023]
Meat quality	Industry	pH, WHL, CIELab(L, a, b)	405	135:135:54	[Barbon et al., 2016]
Synthetic					
Healthcare	Healthcare	Age, Cholesterol, BMI, Exercise	1,000	800:200	Generated for this study
Finance	Finance	Income, Credit Score, Marital Status	1,000	750:250	Generated for this study
Manufacturing	Industry	Temperature, Pressure, Speed	1,000	900:100	Generated for this study
E-commerce Fraud	Retail	Transaction Amount, Country	1,000	950:50	Generated for this study
Energy Grid	Energy	Usage, Voltage	1,000	850:150	Generated for this study
Education	Education	Attendance, Study Hours, Grades	1,000	700:300	Generated for this study

Note: Synthetic datasets were programmatically generated using custom Python scripts

implementing domain-specific constraints.

4.2 Experimental Setup

This study evaluated multiple data augmentation algorithms on ten distinct datasets, each representing an imbalanced classification problem. The datasets, listed in Table 1, include: Healthcare, Finance, Quality Control, Education, Meat Quality, Credit Approval, Fraud Detection, Energy, Mammographic Mass, and Pima Indians Diabetes. The key steps are summarized below:

- Data Splitting: Each dataset was randomly partitioned into training (70%) and test (30%) sets. Stratified sampling was used to preserve class proportions.
- Augmentation Percentages: Four levels of oversampling (5%, 15%, 30%, 50%) were tested, each indicating the ratio of synthetic minority samples to be added relative to the original minority-class size in the training data.
- Augmentation Methods: Classical SMOTE, BorderlineSMOTE, SVMSMOTE, SMOTEENN, LVQ_SMOTE, DE_oversampling, RandomOverSampler, MSMOTE, polynomial fit SMOTE, and our proposed DPG augmentation algorithm in two variants (*DPG_augm._original* and *DPG_augm._border*).
- Classifiers: Three algorithms—Decision Tree, Logistic Regression, and K-Nearest Neighbors—were trained on all data augmentation algorithms.
- Performance Metric: F1-Score was chosen to assess classification performance on the test set since it ensures a balanced evaluation of both majority and minority class performance, reducing the risk of overfitting to dominant classes. F1-Score is more informative for imbalanced tasks than raw accuracy
- Domain Constraints and Violations: Each dataset included domain-specific "logic constraints" (e.g., valid ranges for features). After synthetic points were generated, we checked each point for violations (e.g., negative or out-of-range values). Constraint violations were recorded to show how well each augmentation method respected real-world feasibility.
- Multiple Repetitions: To capture variability, each experimental condition was repeated several times with different random seeds. The mean F1 score and the occurrence of constraint violations were then summarized.

At the end of every experiment, the run time was measured for each combination of dataset, augmentation method, augmentation percentage, and classifier. This timing includes the generation of synthetic data and the retraining of the corresponding classifier. The running time results, while not the main focus, provide an additional performance indicator when choosing an augmentation strategy for practical scenarios.

4.3 Results and Discussion

4.3.1 Results and Discussion

A critical aspect of this study is the assessment of domain constraint violations introduced by different data augmentation methods. Unlike purely statistical measures, adhering to logical constraints ensures the generated synthetic data retains real-world plausibility. Figure 4.3.1 visualizes the cumulative total count of constraint violations for each augmentation method at a 50% augmentation level over the tested datasets (Table 1). This plot underscores that some augmentation strategies inherently tend to generate data falling outside defined domain boundaries. LVQ_SMOTE presents the largest number of constraint violations, followed by DE_oversampling and SVMSMOTE with considerably fewer counts of ill-defined data, and polymom_fit_SMOTE is at the last position among the methods that generate constraint violations. It's crucial to note that DPG Augmentation (both "original" and "border" modes) did not introduce any constraint violations. The fact that these methods did not introduce any violation values, confirms the robustness of this method and shows that it could be implemented to increase the number of data and improve the metrics on Machine Learning models by ensuring the domain compliance of the generated data.

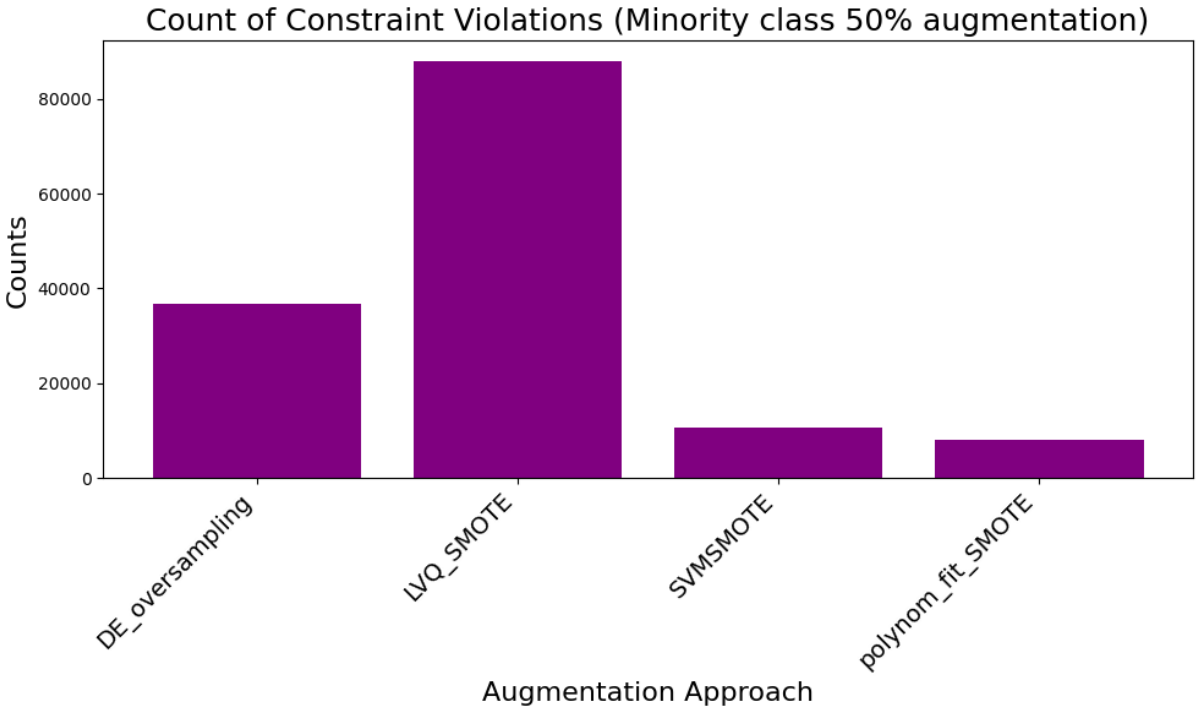

Figure 4.3.1: Count of Constraint Violations (Minority class 50% augmentation)

The impact of data augmentation on classifier performance is examined across three distinct algorithms susceptible to class imbalance: Decision Tree, Logistic Regression, and K-Nearest Neighbors (kNN). Figures 4.3.2, 4.3.3, and 4.3.4 display the average F1-score

obtained by each algorithm, over the tested datasets (Table 1), with vertical red lines indicating the methods prone to constraint violations. These visualizations allow us to correlate the tendency for constraint violations in the generated data by some methods and the related performance metrics obtained.

Decision Tree

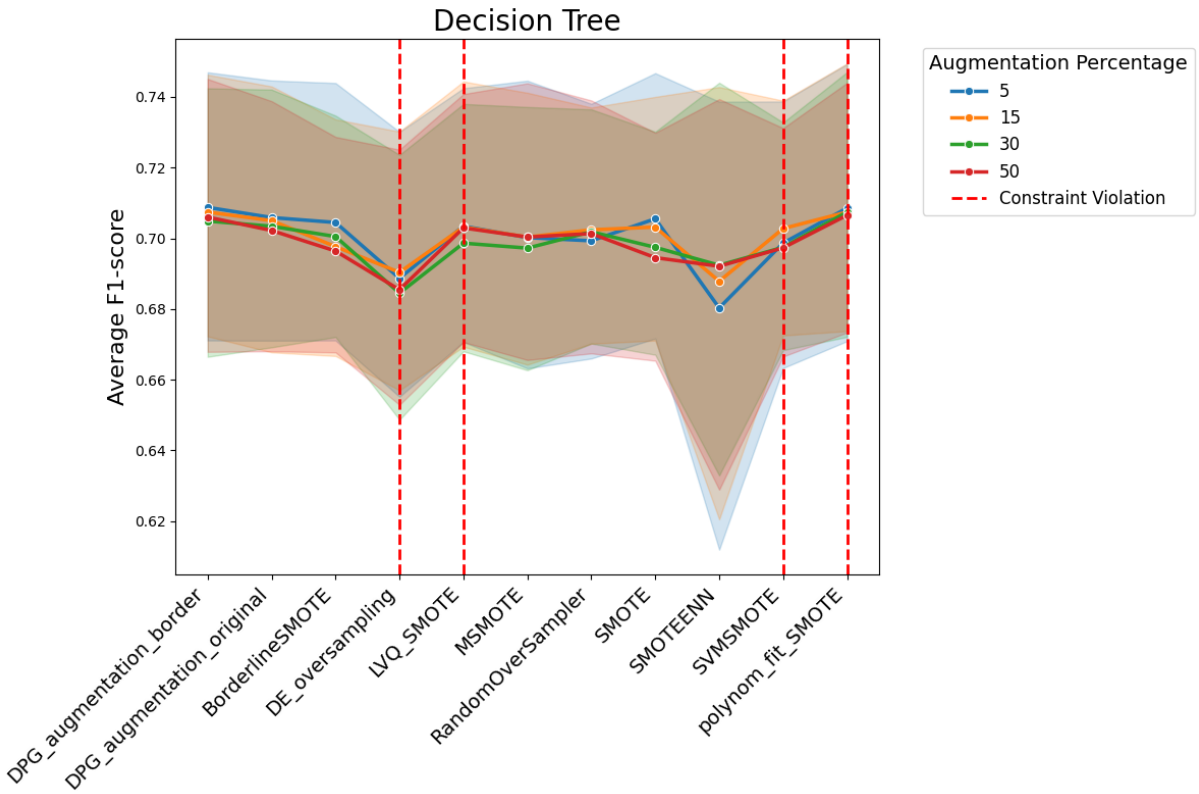

Figure 4.3.2: Decision Tree Performance with Constraint Violation Highlights

As visualized in Figure 4.3.2, the overall performance across the different augmentation algorithms is generally consistent. However, there's a subtle trend where BorderlineSMOTE and SMOTE exhibit a slight decrease in the average F1-score as the augmentation percentage increases. The subtle trend in these methods suggests that these methods are susceptible to the negative effects generated by augmentation techniques, where the addition of synthetic data generates an excess of similar values that could harm on F1-score's performance. It's important to note that methods such as LVQ_SMOTE, DE_oversampling, SVMSMOTE and polynomon_fit_SMOTE do not present a clear change or behaviour as they increase the number of data added to the training dataset. This is relevant, considering that these methods introduce domain constraint violations as shown in the bar plot, and might be harmful due to that as it is increased the data. By contrast, SMOTEENN consistently demonstrates the lowest performance at lower augmentation percentages, showing a performance improvement as the percentage of augmentation is

increased.

Logistic Regression

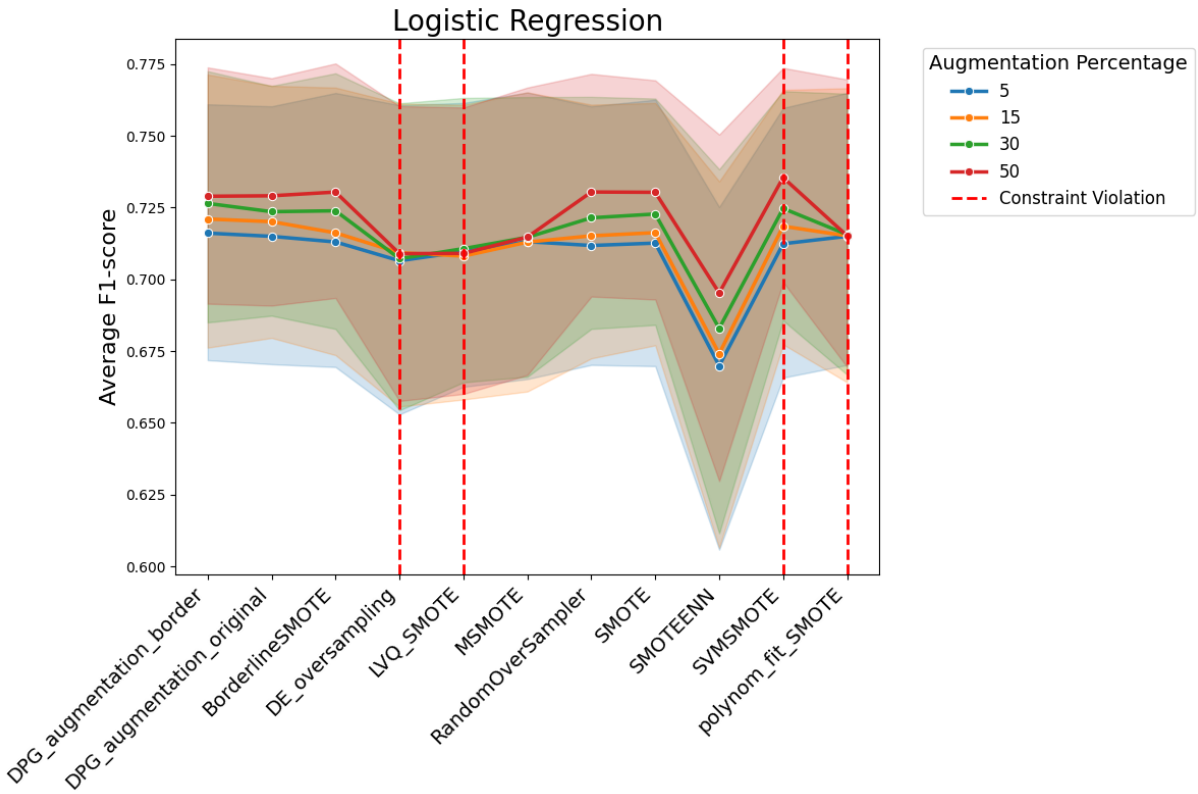

Figure 4.3.3: Logistic Regression Performance with Constraint Violation Highlights

Figure 4.3.3 shows the results for the Logistic Regression classifier. Most augmentation methods exhibit increased F1-score as the augmentation level increases, indicating that, in general, Logistic Regression benefits from larger and more balanced training datasets generated by these approaches. Notable exceptions include LVQ_SMOTE, DE_oversampling, MSMOTE, and polynomon_fit_SMOTE. Except for MSMOTE, those algorithms produce high rates of constraint violations, which could be a cause of the lower metric performance and non-upward trend, even with the augmentation. Although MSMOTE does not directly generate data outside defined bounds, its logic categorizes existing minority class data as safe, borderline, or outlier based on local density; if the original data already contain unusual or potentially erroneous values that do not technically violate constraints, MSMOTE might then preferentially oversample these atypical regions, leading to a skewed synthetic dataset concentrated around problematic, albeit technically valid, data points and ultimately harming model generalization. SMVSMOTE is another method that generates constraint violations and also shows an increasing trend in the F1-score as the augmentation level is raised. This suggests that some methods are particularly sensible, regardless of the domain constraint violations, the increase in data

tends to generate a performance improvement.

K-Nearest Neighbors (kNN)

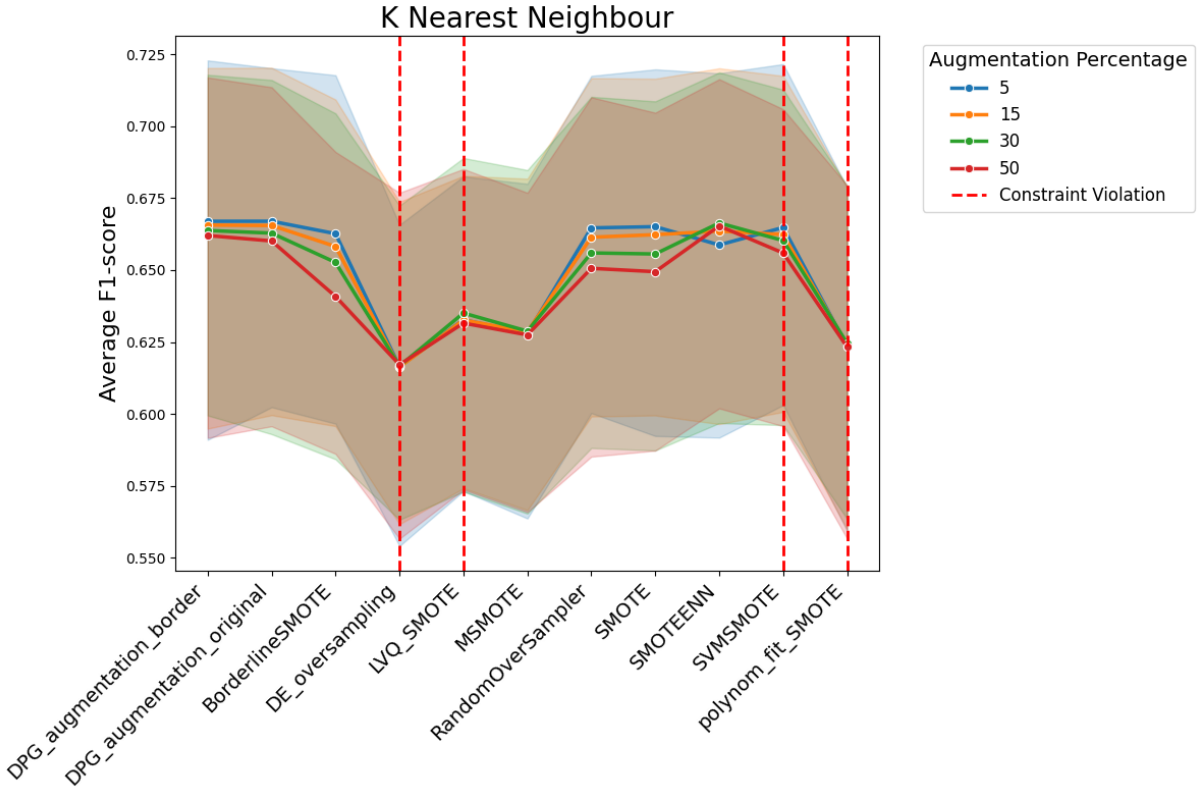

Figure 4.3.4: K Nearest Neighbour Performance with Constraint Violation Highlights

The KNN results, as depicted in Figure 4.3.4, demonstrate a distinct trend compared to the previous classifiers. Methods involving a vertical line (LVQ_SMOTE, DE_oversampling and polynomon_fit_SMOTE) produce particularly low F1-scores relative to the rest, confirming that the domain constraint violations associated with those approaches negatively impact kNN's performance; they produce the worst metrics.
The plot highlights another common behavior in all tested approaches but especially on: Borderlinesmote, Randomoversampling, SMOTE, and SMOTEENN methods. Most of these algorithms display a performance decline (decrease F1-score) when the augmentation level increases, this behavior highlights the effects of an excessive volume of generated synthetic data when using kNN: too much synthetic data can dilute the local distribution patterns, making this classifier highly sensitive to this factor. In addition to the general trend, MSMOTE also shows a reduced level of F1-score. While MSMOTE does not generate constraint violations directly, its process of categorizing and oversampling "borderline" instances might cause it to amplify noise in the training data; because kNN relies on identifying proximities between instances for the proper classification, that approach does not benefit this model type.

While it's tempting to directly equate constraint violations with degraded performance, the results suggest a more complex relationship. LVQ_SMOTE, DE_oversampling, and polynomon_fit_SMOTE all suffer from high violation rates and tend towards lower F1-scores overall, suggesting that kNN is particularly sensitive to data quality and real-world plausibility; this may be because kNN does not perform well when an ill-defined feature has high relevance. The situation of SMVSMOTE highlights the complexity of these relationships: this algorithm, which has some constraint violations, still shows a relatively good F1-score. This points to a trade-off between data generation and data quality: the overall trend for this method shows there is performance improvement because there are more data points for the model to be trained with, rather than generating and considering domain constraint features. Conversely, the DPG-based methods maintain more stable and relatively high-performance metrics with no constraint violations during all augmentation levels, which suggests a higher data quality generated for the kNN, improving at the same time the volume of data and ensuring that its quality does not generate bias or ill-performing results.

4.3.2 Classifier Performance and Data Augmentation Quality

Across all classifiers, a key observation is that indiscriminate data augmentation can sometimes be detrimental to performance, especially on certain classifiers and smaller datasets that are not severely imbalanced, such as Mammographic Mass (427:403) and Australian Credit (383:307). Therefore, as it happens in SVMs - Farquad and Bose (2012) - an excessive or inappropriate boundary definition (i.e., with augmented or synthetic data) leads to out-of-range values. The interplay between each classifier and poorly generated synthetic data creates a trade-off between the potential for increased performance and the final model's behavior, which may become skewed due to domain constraint violations or biased training [12, 17, 22].

The discussion shows that a key to perform well is not only about using data augmentation or syntethization, but also ensuring the quality of the generated data. What makes this effort valuable for improving metrics is a combination of techniques, for what it's relevant to focus on how the information behaves after those techniques are applied [6, 21, 31]

Critically, DPG augmentation offers a potential solution to this dilemma. As highlighted by González-Sendino et al. (2024) [19], generating transparent synthetic data is key to building trustworthy datasets and enabling interpretable models. Our results suggests that strategies such DPG tend to be a promising path as the data generated helps to build adherence of defined domain constraints [47, 59], without sacrificing model's performance. Future work should thus focus on and study the real benefits and potential of DPG-guided augmentation and its impact at diverse types of data, for its better results and not generating dataset level biased outputs.

4.3.3 Best performing methods

In addition to the per-classifier examination described in Section 4.3.1, a further statistical analysis was conducted focusing only on methods that produced no domainconstraint violations. Specifically, any algorithms that frequently generated invalid points (e.g., DE_oversampling or LVQ_SMOTE) were removed, leaving seven relatively "clean" approaches: DPG_augmentation_border, DPG_augmentation_original, BorderlineSMOTE, MSMOTE, RandomOverSampler, SMOTE, and SMOTEENN. The data were then grouped by dataset–classifier combinations, averaged by F1 scores, and subjected to a Friedman test with Nemenyi posthoc comparisons.

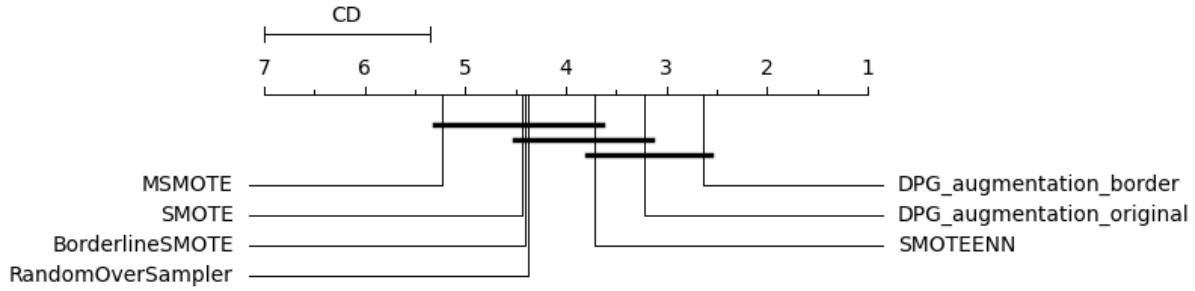

Figure 4.3.5: Nemenyi diagram comparing the average ranks of the methods with no constraint violations.

In Figure 4.3.4, each method is positioned on a horizontal axis according to its average rank (lower ranks are better). The horizontal bar labelled "CD" (critical distance) represents the threshold (1.644 in this case) beyond which two methods can be considered statistically different at the 5% significance level. Methods connected by a darker horizontal line are those for which the difference in mean ranks does not exceed the critical distance, indicating that these approaches are not statistically distinguishable from each other. This explains why, despite visibly lower performance in some earlier plots, SMOTEENN now appears in a higherranked group: when the data are averaged per dataset–classifier pair and highviolation methods are filtered out, the Nemenyi test finds no significant difference between SMOTEENN and the DPG variants. By contrast, DPG_augmentation_border and DPG_augmentation_original stand out to the left of the diagram and are not joined to most of the other methods by any dark line, reinforcing that their difference from the rest exceeds the critical distance and is, therefore, significant.
Hence, once algorithms that generate high violation counts are filtered out, DPG augmentation continues to respect the domain constraints and emerges as the strongest strategy overall among the remaining contenders. Meanwhile, SMOTEENN appears more competitive in this narrower field, highlighting the degree to which sample filtering and dataaggregation decisions can alter observed rank orderings.

4.4 Diversity analysis

The original data used in this experiment comes from the Mammographic Mass dataset (Table 1) [35]. In this section, we present and discuss the diversity of synthetic data produced by our DPG Augmentation algorithm compared to other augmentation methods (e.g., SMOTE variants). Figures 4.4.5 and 4.4.6 illustrate a set of PCA scatter plots, each subplot corresponding to a particular augmentation method or mode (e.g., *DPG Aug. Original* vs. *DPG Aug. Border*, or SMOTE vs. BorderlineSMOTE vs. SMOTEENN, etc.) and an augmentation level (5%, 15%, 30%, 50%). The *blue "X"* markers show newly generated synthetic points; *black* markers indicate the majority class; *orange* markers denote the minority class.

4.4.1 DPG Augmentation Overview

Our proposed DPG Augmentation algorithm uses domain constraints derived from DPG—an ensemble explanation technique—to guide data generation. As shown in the code snippet, we developed a Genetic Algorithm (GA) that evolves candidate samples to either:

1. *"Original" mode*: Draw new points *throughout* the feasible region (as defined by the DPG constraints for the target class).

2. *"Border" mode*: Generate new minority samples specifically along the *decision boundary* between minority and majority classes, while still respecting the same DPG constraints.

Because of this design, the "original" mode typically spreads synthetic samples more broadly, whereas the "border" mode concentrates them near the separating frontier (hence the somewhat denser cloud along the class boundary).

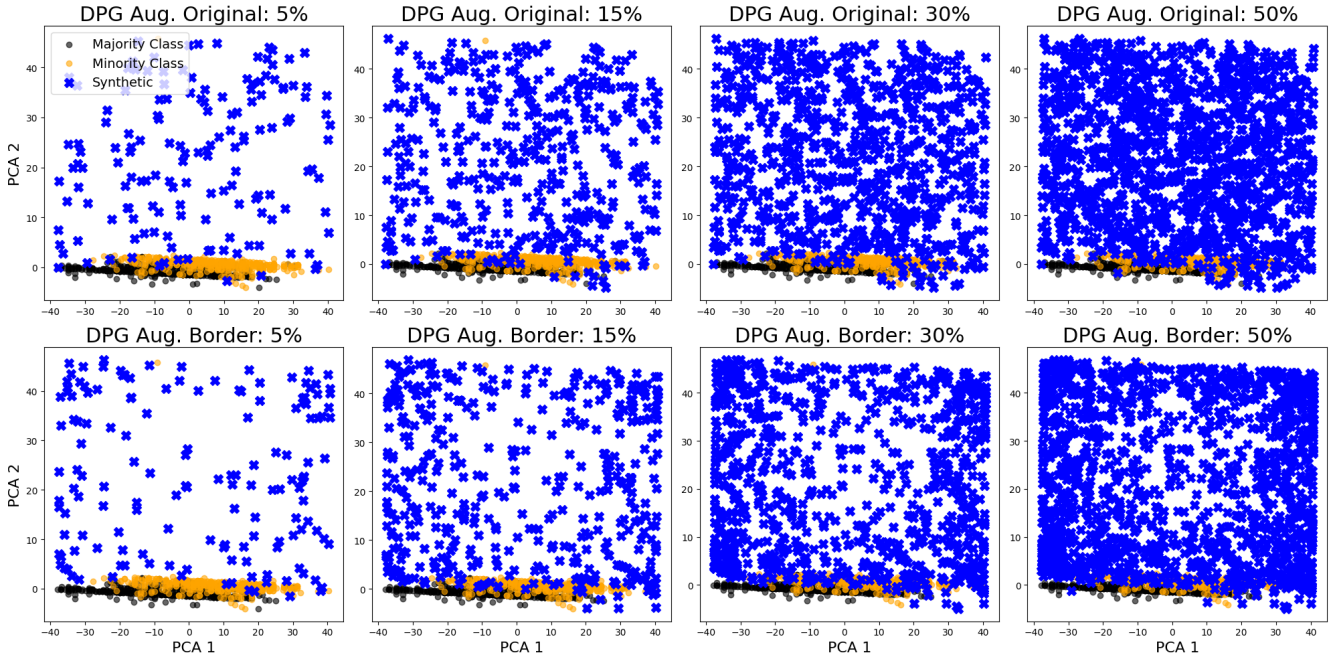

Figure 4.4.6: PCA scatter plots showing *DPG Aug. Original* and *DPG Aug. Border* modes at 5%, 15%, 30%, and 50% augmentation. Majority points are in black, minority points in orange, and synthetic samples in blue (X markers).

From the top row of Figure 4.4.5, for example, *DPG Aug. Original* with 5% augmentation populates the minority-class region widely in PCA space. As we increase the augmentation to 15%, 30%, and 50%, we see many more minority samples scattered throughout. For *DPG Aug. Border* (second row), the new points are also guided by "border awareness," resulting in samples that appear to cluster a bit more around the boundary regions—though the genetic search still injects diversity.

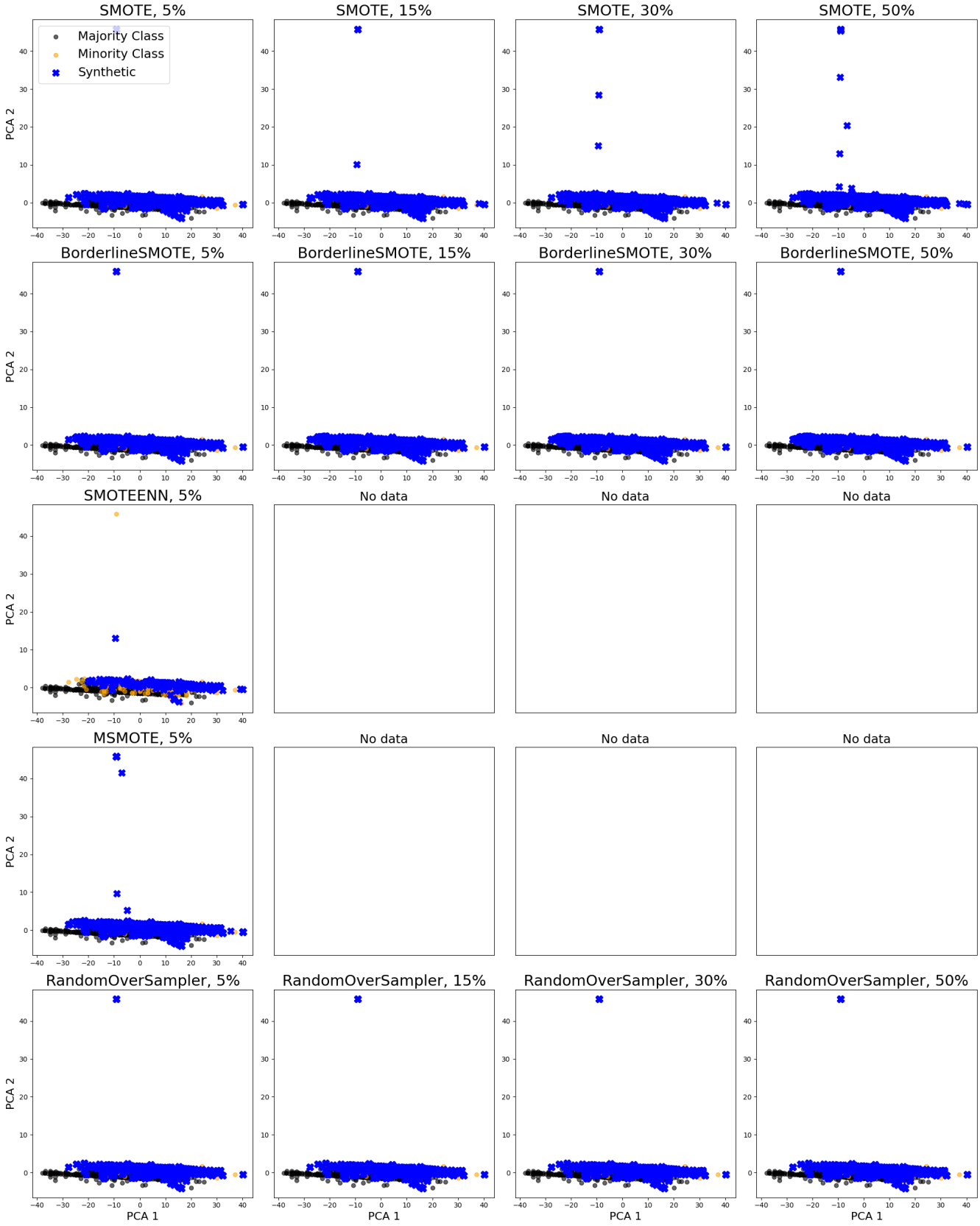

Figure 4.4.7: PCA scatter plots illustrating various SMOTE-based methods (SMOTE, BorderlineSMOTE, SMOTEENN, MSMOTE) at 5%, 15%, 30%, and 50% augmentation. Majority points are black, minority points are orange, and synthetic samples are blue (X markers).

4.4.2 Comparison with SMOTE Variants

Figure 4.4.6 shows classical SMOTE methods at equivalent percentages. SMOTE or BorderlineSMOTE often add minority points near existing minority samples, leading to comparatively narrower "bands" of new points. Further, methods like SMOTEENN may combine oversampling and undersampling, occasionally yielding *no net new samples* at certain percentages (hence the "No data" subplots). Similarly, MSMOTE (from `smote_variants`) computes how many new points to add based on a ratio relative to the majority class, so it sometimes falls outside the targeted percentages or fails to produce data for that exact bin.

In contrast, DPG Augmentation ensures all new synthetic samples are created under *DPG constraints*, showing valid feature bounds for "Age," "BI-RADS," etc., while boosting *diversity* via the GA's fitness function. By integrating domain constraints and using an evolutionary search to spread out samples, DPG Augmentation can produce synthetic data that either covers a wider region of the minority-class manifold or concentrates near the class boundary, depending on the chosen mode.

Overall, these plots confirm that DPG Augmentation -both *original* and *border* modes- provides a controllable way to diversify synthetic data while respecting logical or domain constraints derived from DPG. This approach can complement or outperform traditional SMOTE-based methods in terms of coverage, constraint adherence, and enhanced interpretability. Indeed, adhering to DPG constraints offers a clear explanatory advantage: every synthetic data point is traceable to specific, model-derived intervals or boundary conditions, thus ensuring that augmented samples are readily justified and verifiable within the domain.

4.5 Running Time Benchmark

This section reports the computational cost of each augmentation method, measured as average running time across multiple runs at a fixed augmentation level (50%). Figure 4.5.8 depicts the average run time for all methods, including the *DPG Aug. Border* algorithm, while Figure 4.5.9 excludes the *DPG Aug. Border* method to allow finer inspection of the remaining algorithms on a comparable scale.

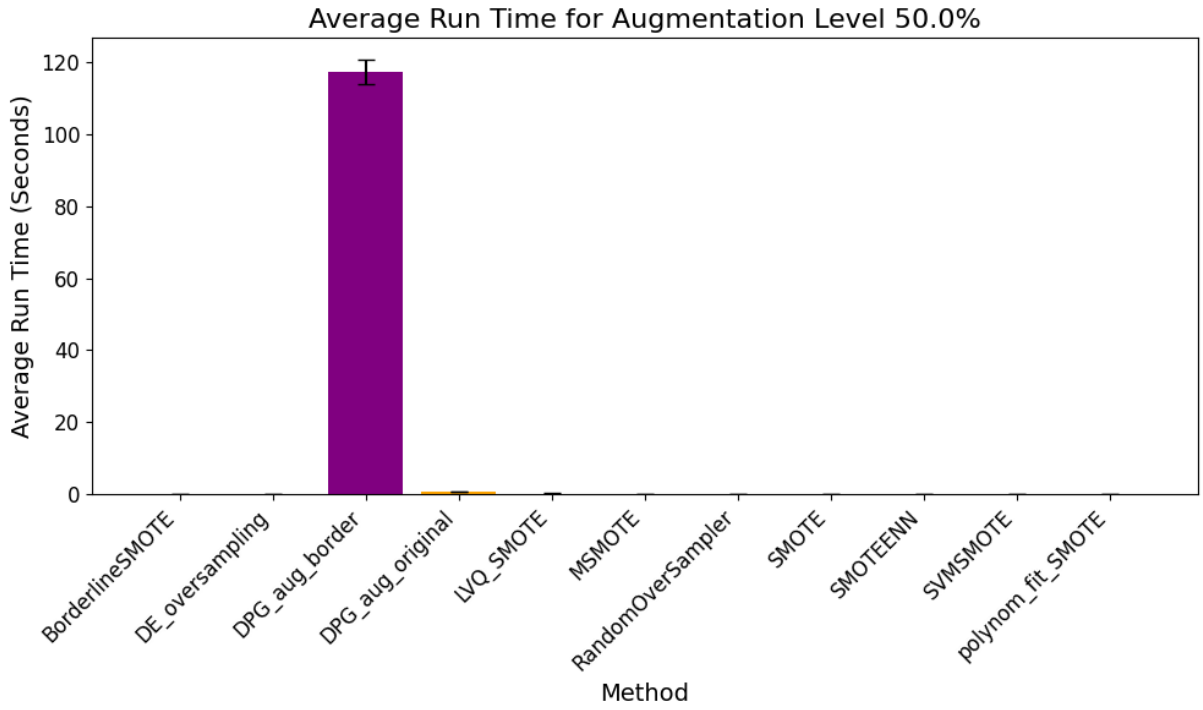

Figure 4.5.8: Average running time (in seconds) of each augmentation method at 50% augmentation, *including DPG Aug. Border*. The exceptionally high bar for *DPG Aug. Border* overshadows other methods, indicating that its genetic search—focusing on boundary samples with complex fitness functions—is computationally intensive.

From Figure 4.5.8, it is evident that the *DPG Aug. Border* approach presents the most significant computational overhead. Its boundary-oriented genetic algorithm iterates repeatedly to discover and refine borderline minority samples while preserving logical constraints, thus incurring a far greater run time compared to other interpolation-based or simpler evolutionary methods.
To better compare the remaining approaches on a single scale, we remove *DPG Aug. Border* from the plot. Figure 4.5.9 then shows that:

- *DPG Aug. Original* remains the slowest among the remaining methods; however, after excluding the border variant, the run time (sub-second or in the order of 0.7 s) is still much lower than the hundreds of seconds observed for *DPG Aug. Border*.
- Classical SMOTE variants (e.g., *SMOTE*, *BorderlineSMOTE*, *SMOTEENN*, *MSMOTE*) exhibit relatively quick execution times, typically well below 1 s, since they rely on either interpolation or limited undersampling logic rather than a GA-driven search.
- *DE_oversampling* and *LVQ_SMOTE* also remain comparable in speed.
- The *DPG Aug. Original* mode, while still GA-based, operates with simpler constraints and no explicit boundary calculations, accounting for the marked reduction (by over two orders of magnitude) in run time versus the border mode.

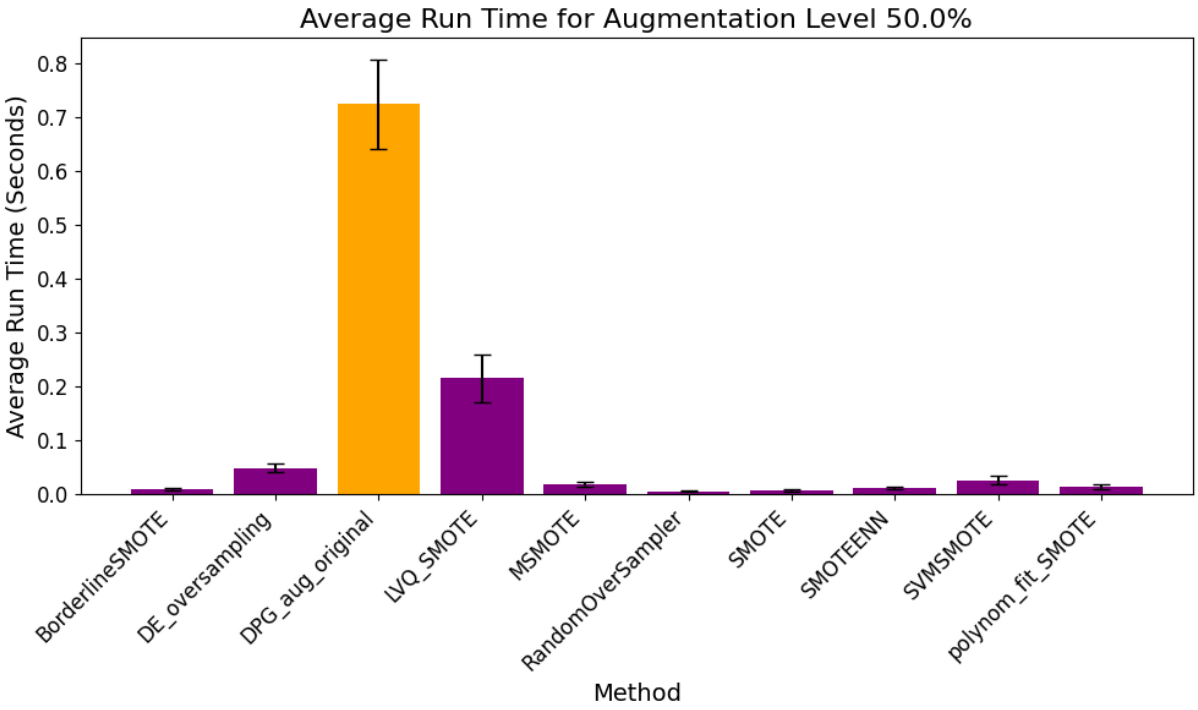

Figure 4.5.9: Average running time at 50% augmentation *excluding DPG Aug. Border*. Methods are now on a comparable scale. *DPG Aug. Original* shows the highest run time among the remaining methods, but is reduced by over 100× compared to the border-focused approach.

Chapter 5

Discussion

5.1 Discussion

5.1.1 Summary of Key Findings

This study proposed a novel data augmentation method guided by Decision Predicate Graphs (DPG) (Arrighi et. al. 2024) to enhance both diversity and explainability in synthetic data generation. The research objectives focused on (1) generating improved samples with enhanced explainability and avoiding infeasible samples, and (2) enhancing both diversity, by covering underrepresented regions, and explainability, by respecting logical domain constraints.

The experimental results demonstrate that the DPG-guided augmentation algorithm significantly improves classifier performance, particularly for mi- nority classes, while maintaining strict adherence to domain-specific constraints. Thus, we have successfully achieved the primary objective of generating high-quality synthetic data. Compared to traditional methods like SMOTE and its variants, the proposed approach gener- ated more realistic and logically consistent synthetic data, as evidenced by no constraint violation rates and higher F1 scores across multiple datasets. This shows the achievement of the second objective, increasing diversity without sacrificing explainability by enforcing domain constraints.

5.1.2 Interpretation of Results

The results align with our hypothesis that integrating DPG constraints into the data augmentation process would yield more interpretable and domain-compliant augmented data. For instance, the Border mode of the DPG-guided algorithm consistently produced augmented samples near class decision boundaries, which proved particularly effective in improving the performance of classifiers like k-Nearest Neighbors (kNN). This finding is consistent with prior research highlighting the importance of boundary samples in

imbalanced classification tasks [25].
However, the performance gains varied across classifiers. While Logistic Regression benefited significantly from the increased diversity of augmented data, Decision Trees showed only marginal improvements, suggesting that the effectiveness of the DPG-guided approach may depend on the classifier's sensitivity to feature space transformations. This observation aligns with the literature on the varying impacts of data augmentation on different machine learning models [62].

5.1.3 Implications of the Results

The practical implications of this work are substantial, particularly in regulated domains such as healthcare and finance, where synthetic data must adhere to strict logical and domain-specific constraints. By ensuring that augmented data respects these constraints, the DPG-guided approach reduces the risk of generating unrealistic or invalid records, which could otherwise lead to flawed model predictions. For example, in healthcare applications, the ability to generate synthetic patient data that adhere to clinical guidelines could enhance the training of diagnostic models without compromising patient privacy or data integrity.
Moreover, the improved explainability of the augmented data is a significant advantage. By tracing synthetic samples back to specific DPG constraints, stakeholders can better understand how the model makes decisions, fostering trust and transparency in AI systems. This is particularly critical in high-stakes applications where model interpretability is a legal or ethical requirement.

5.1.4 Strengths of the Proposed Approach

The primary strength of the DPG-guided augmentation method lies in its ability to balance diversity and explainability. Unlike traditional methods like SMOTE, which rely on local interpolation and may produce unrealistic samples, the DPG-guided approach ensures that all synthetic data points adhere to the logical rules derived from the ensemble model. This not only improves the quality of the augmented data but also enhances the interpretability of the final model.
Another key advantage is the flexibility of the algorithm, which operates in two distinct modes: Original and Border. The Original mode is well-suited for general augmentation tasks, while the Border mode is particularly effective for improving minority-class recognition by focusing on decision boundaries. This dual-mode approach allows practitioners to tailor the augmentation process to their specific needs.

5.1.5 Limitations of the Study

Despite its strengths, the proposed method has some limitations. First, the computational cost of the DPG-guided algorithm, particularly in Border mode, is significantly higher than that of traditional methods like SMOTE. This is due to the iterative nature of the genetic algorithm, the complexity of the fitness function (which evaluates both constraint adherence and boundary proximity), and the use of lists as the data structure in the genetic algorithm tailored for DPG augmentation in Border mode. While lists are necessary to add diversity and maximize differences in the newly generated data, they introduce additional computational overhead compared to more efficient data structures like NumPy arrays.

Second, the effectiveness of the DPG-guided approach depends on the quality of the DPG constraints extracted from the ensemble model. In cases where the underlying model is poorly calibrated or the feature space is highly complex, the extracted constraints may not accurately reflect the true decision boundaries, leading to suboptimal augmentation.

Third, this study primarily focused on binary classification tasks, with only one dataset (the Meat Quality dataset) featuring a 3-class target variable. Even in this case, only one minority class was present, limiting the evaluation of the DPG-guided approach in more complex multiclass scenarios with multiple minority classes. This restricts the generalizability of the findings to datasets with more intricate class structures.

Finally, the benchmark was performed without hyperparameter tuning, which may have limited the performance of our approach and some baseline methods. While this decision was made to ensure a fair comparison across methods, future work could explore the impact of hyperparameter optimization on the results.

Chapter 6

Conclusion and Feature work

Overall, the genetic search strategy employed by DPG Aug. Border yields the most balanced and boundary-focused augmented samples, but its cost in computational time is extremely high. By contrast, DPG Aug. Original mitigates this overhead, although it remains more expensive than standard SMOTE-like methods. Practitioners who need strict boundary coverage or have specific domain constraints may still prefer the Border mode despite its runtime, whereas DPG Aug. Original may offer a trade-off between reasonable computational cost and constraint-respecting diversity improvements.
In conclusion, this study demonstrates that DPG-guided data augmentation offers a robust and interpretable solution to the challenges of data augmentation in imbalanced classification tasks. By incorporating domain-specific constraints and focusing on decision boundaries, the proposed method improves both the quality and explainability of augmented data, making it a valuable tool for applications where trust and transparency are critical. While there are limitations to be addressed, the results suggest that the DPG-guided approach represents a significant step forward in the field of data augmentation and explainable AI.

6.0.1 Future Work

Several directions for future research emerge from this study. First, efforts could be made to optimize the computational efficiency of the DPG-guided algorithm, potentially through parallelization, the use of more efficient data structures, or the adoption of advanced search strategies. Reducing the runtime would make the method more accessible for large-scale applications.
Second, the approach could be extended to other types of data, such as time-series or text, where domain constraints and explainability are equally important. For example, in natural language processing, DPG constraints could be used to ensure that synthetic text adheres to grammatical rules or semantic coherence.
Third, to address the issue of mode collapse—a challenge in both generative models

and GA-based synthetic data generation—future work could explore techniques such as diversity-promoting loss functions, adaptive mutation rates, balanced crossover operators, and multi-objective optimization [42]. These techniques could help ensure that the synthetic data covers the full range of decision paths, thereby enhancing interpretability and reducing the risk of generating biased or incomplete samples.

Fourth, future studies should evaluate the DPG-guided augmentation method on datasets with multiclass target variables, particularly those with multiple minority classes. This would involve adapting the algorithm to handle more complex decision boundaries and interactions between minority classes, which are common in real-world applications.

Finally, future work could explore the integration of the DPG-guided approach with other advanced augmentation techniques, such as counterfactual augmentation, generative adversarial networks (GANs), or variational autoencoders (VAEs). Combining the strengths of these methods with the interpretability of DPG constraints could yield even more powerful and flexible data augmentation tools.

Bibliography

[1] Adadi, A., & Berrada, M. (2018). *Peeking Inside the Black Box: Explainable AI (XAI) Techniques*. IEEE Access, 6, 52138–52160.

[2] Alauthman, M., Aldweesh, A., Al-qerem, A., Pozi, M. S. M., Din, A. M., Al-Smadi, Y., Abaker, A. M., & Alzubi, B. (2023). *Tabular Data Generation to Improve Classification of Liver Disease Diagnosis*. Applied Sciences, 13(4), 2678.

[3] Alqudah, R., Al-Mousa, A. A., Hashyeh, Y. A., & Alzaibaq, O. Z. (2023). *A Systemic Comparison Between Using Augmented Data and Synthetic Data as Means of Enhancing Wafermap Defect Classification*. In Proceedings of the 2023 IEEE Conference on Semiconductor Manufacturing (pp. 1–10).

[4] Arjovsky, M., Chintala, S., & Bottou, L. (2017). *Wasserstein GAN*. ICML, 214–223.

[5] Arrighi, L., Pennella, L., Tavares, G. M., & Barbon Junior, S. (2024). *Decision Predicate Graphs: Enhancing Interpretability in Tree Ensembles*. (Preprint).

[6] Azhar, N. A., Pozi, M. S. M., Din, A. M., & Jatowt, A. (2023). *An Investigation of SMOTE Based Methods for Imbalanced Datasets With Data Complexity Analysis*. IEEE Transactions on Knowledge and Data Engineering, 35(7), 6651–6668.

[Barbon et al., 2016] Barbon, A. P. A. C., Barbon Jr., S., Mantovani, R. G., Fuzyi, E. M., Peres, L. M., & Bridi, A. M. (2016). Storage time prediction of pork by Computational Intelligence. *Computers and Electronics in Agriculture*, 127, 368-375. `https://doi.org/10.1016/j.compag.2016.06.023`

[7] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). *Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI*. Information Fusion, 58, 82–115.

[8] Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). *A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data*. Information Sciences, 267, 1–17.

[9] Castelli, M., Cattaneo, G., Manzoni, L., & Vanneschi, L. (2019). *A Distance Between Populations for n-Points Crossover in Genetic Algorithms*. In Genetic and Evolutionary Computation Conference (GECCO) (pp. 287–295).

[10] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique*. Journal of Artificial Intelligence Research, 16, 321–357.

[11] Chapelle, R., & Falissard, B. (2024). *Statistical Properties and Privacy Guarantees of an Original Distance-Based Fully Synthetic Data Generation Method*. [Preprint].

[12] Chen, L., Cai, Z., Chen, L., & Gu, Q. (2010). *A Novel Differential Evolution-Clustering Hybrid Resampling Algorithm on Imbalanced Datasets*. Third International Conference on Knowledge Discovery and Data Mining, 81–85.

[13] Chen, Y., Elliot, M., & Sakshaug, J. (2016). *A Genetic Algorithm Approach to Synthetic Data Production*. In Privacy in Statistical Databases (pp. 188–199). Springer.

[14] Chen, L., Elliot, T., & Smith, T. (2018). *Hybrid Genetic-GAN Approaches for Data Generation*. Expert Systems, 35(3), 77–85.

[15] Chen, Y. (2020). *Genetic Algorithms and Their Applications to Synthetic Data Generation*. Ph.D. Thesis, University of Manchester.

[16] Chen, Y., Elliot, M., & Sakshaug, J. (2023). *Genetic Algorithms in Matrix Representation and Its Application in Synthetic Data*. [Preprint / Conference Paper].

[17] Danese, D. (2024). *Optimizing Synthetic Data from Scarcity: Towards Meaningful Data Generation in High-Dimensional Low-Sample Size Domains*. [Preprint].

[18] Demsar, J. (2006). *Statistical Comparisons of Classifiers over Multiple Data Sets*. Journal of Machine Learning Research, 7(Jan), 1–30.

[19] El Emam, K., Mosquera, L., & Hoptroff, R. (2020). *Practical Synthetic Data Generation: Balancing Privacy and the Broad Availability of Data*. O'Reilly Media.

[20] Farquad, M. A. H., & Bose, I. (2012). *Preprocessing Unbalanced Data Using Support Vector Machine*. Decision Support Systems, 53(1), 226–233.

[21] Friedman, M. (1937). *The use of ranks to avoid assumption of normality implicit in the analysis of variance*. Journal of the American Statistical Association, 32(200), 675–701.

[22] Gazzah, S., & Essoukri Ben Amara, N. (2008). *New Oversampling Approaches Based on Polynomial Fitting for Imbalanced Data Sets*. In Proc. of the Eighth IAPR Int. Workshop on Document Analysis Systems, 677–684.

[23] González-Sendino, R., Serrano, E., & Bajo, J. (2024). *Mitigating Bias in Artificial Intelligence: Fair Data Generation via Causal Models for Transparent and Explainable Decision-Making*. [Preprint].

[24] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). *Generative Adversarial Networks*. NeurIPS, 1–9.

[25] Han, H., Wang, W., & Mao, B. (2005). *Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning*. Lecture Notes in Computer Science, 3644, 878–887.

[26] Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

[27] Hollander, M., & Wolfe, D. A. (1973). *Nonparametric Statistical Methods*. New York: John Wiley & Sons.

[28] Hu, X. (2025). *Context-Aware Genetic Algorithm for Fault Localization in Complex Systems*. Future Generation Computer Systems, 140, 45–59.

[29] Hu, S., Liang, Y., Ma, L., & He, Y. (2009). *MSMOTE: Improving Classification Performance When Training Data is Imbalanced*. In Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence (pp. 13–17).

[30] Iqbal, A., & Sikdar, B. (2023). *Are Classifiers Trained on Synthetic Data Reliable? An XAI Study*. Expert Systems with Applications, 210, 118300.

[31] Jordon, J., Szpruch, L., Houssiau, F., Bottarelli, M., Cherubin, G., Maple, C., Cohen, S. N., & Weller, A. (2022). *Synthetic Data: What, Why and How?* [Preprint].

[32] Nethravathi, P. S., & Karibasappa, K. (2017). *Augmentation of Customer's Profile Dataset Using Genetic Algorithm*. International Journal of Research and Scientific Innovation, 4(6), 33–38.

[33] Kummer, A., Ruppert, T., Medvegy, T., & Abonyi, J. (2022). *Machine learning-based software sensors for machine state monitoring - The role of SMOTE-based data augmentation*. Applied Sciences, 12(21), 11267.

[34] Li, Z., Song, Y., Li, R., Gu, S., & Fan, X. (2022). *A Novel Data Augmentation Method for Improving the Accuracy of Insulator Health Diagnosis*. Sensors, 22(21), 8466.

[35] Longo, L., Brcic, M., Cabitza, F., Choi, J., Confalonieri, R., Del Ser, J., Guidotti, R., Hayashi, Y., Herrera, F., Holzinger, A., Jiang, R., Khosravi, H., Lecue, F., Malgieri, G., Páez, A., Samek, W., Schneider, J., Speith, T., & Stumpf, S. (2024). *Explainable Artificial Intelligence (XAI) 2.0: A Manifesto of Open Challenges and Interdisciplinary Research Directions*. Information Fusion, 106, 102301.

[36] Luke, S. (2013). *Essentials of Metaheuristics*. Lulu. Second Edition, Available for free at `http://cs.gmu.edu/~sean/book/metaheuristics/`

[37] Matharaarachchi, S., Domaratzki, M., & Muthukumarana, S. (2024). *Enhancing SMOTE for Imbalanced Data with Abnormal Minority Instances*. [Preprint].

[38] Machado, P. F. C. (2022). *Conception and Evaluation of Data Augmentation Techniques for Tabular Data*. (Master's Dissertation, University of Minho). [Early version or preprint date].

[39] Machado, P. F. C. (2023). *Conception and Evaluation of Data Augmentation Techniques for Tabular Data*. Master's Dissertation, University of Minho.

[40] Macià, N., Orriols-Puig, A., & Bernadó-Mansilla, E. (2008). *Genetic-Based Synthetic Data Sets for the Analysis of Classifiers Behavior*. In Proceedings of the 2008 International Conference on Machine Learning and Applications (pp. 539–545).

[41] Institute of Radiology, University Erlangen-Nuremberg. *Mammographic Mass Dataset*. Available from: `https://archive.ics.uci.edu/dataset/161/mammographic+mass`, accessed March 2024.

[42] Manzoni, L., Mariot, L., & Tuba, E. (2020). *Balanced Crossover Operators in Genetic Algorithms*. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), 2758–2765.

[43] Manzoni, L., Mariot, L., & Tuba, E. (2022). *The Influence of Local Search over Genetic Algorithms with Balanced Representations*. In 2022 Genetic and Evolutionary Computation Conference Companion (pp. 1896–1904).

[44] Margeloiu, A., Jiang, X., Simidjievski, N., & Jamnik, M. (2024). *TabEBM: A Tabular Data Augmentation Method with Distinct Class-Specific Energy-Based Models*. [Preprint].

[45] Menardi, G., & Torelli, N. (2010). *Training and assessing classification rules with imbalanced data*. Data Mining and Knowledge Discovery, 25(1), 63–92.

[46] Meneses Villegas, C., Littin Curinao, J., Coo Aqueveque, D., Guerrero-Henríquez, J., & Vargas Matamala, M. (2024). *Data Augmentation and Hierarchical Classification*

to Support the Diagnosis of Neuropathies Based on Time Series Analysis. Biomedical Signal Processing and Control, 95, 106302.

[47] Metz, L., Poole, B., Pfau, D., & Sohl-Dickstein, J. (2017). *Unrolled Generative Adversarial Networks*. ICLR.

[48] Moreno-Barea, F. J., Franco, L., Elizondo, D., & Grootveld, M. (2022). *Application of Data Augmentation Techniques Towards Metabolomics*. In Proceedings of the 2022 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (pp. 1–8).

[49] Mumuni, A., & Mumuni, F. (2022). *Data Augmentation in Automated Machine Learning: A Review of Trends and Gaps*. Knowledge and Information Systems, 65(3), 455–480.

[50] Mumuni, A., & Mumuni, F. (2025). *Data Augmentation with Automated Machine Learning: Approaches and Performance Comparison with Classical Data Augmentation Methods*. [Preprint or future publication].

[51] Murtaza, H., Ahmed, M., Khan, N. F., Murtaza, G., Zafar, S., & Bano, A. (2023). *Synthetic Data Generation: State of the Art in Health Care Domain*. [Conference or Journal details TBA].

[52] Nakamura, M., Kajiwara, Y., Otsuka, A., & Kimura, H. (2013). *LVQ-SMOTE—Learning Vector Quantization Based Synthetic Minority Over-sampling Technique for Biomedical Data*. BioData Mining, 6, 16.

[53] Nemenyi, P. (1963). *Distribution-Free Multiple Comparisons*. Princeton University.

[54] Nethravathi, P. S., & Karibasappa, K. (2017). *Augmentation of Customer's Profile Dataset Using Genetic Algorithm*. International Journal of Research and Scientific Innovation, 4(6), 33–38.

[55] Pezoulas, V. C., Zaridis, D. I., Mylona, E., Androutsos, C., Apostolidis, K., Tachos, N. S., & Fotiadis, D. I. (2024). *Synthetic Data Generation Methods in Healthcare: A Review on Open-Source Tools and Methods*. [Preprint or Journal details TBA].

[56] Poli, Riccardo, Langdon, William B., and McPhee, Nicholas F. (2008). *A Field Guide to Genetic Programming*. Published via Lulu. Available for free at `http://www.gp-field-guide.org.uk/`. ISBN 978-1-4092-0073-4 (softcover).

[57] Raj, R., Mathew, J., Kannath, S. K., & Rajan, J. (2022). *Crossover-Based Technique for Data Augmentation*. In 2022 International Conference on Machine Learning and Cybernetics (ICMLC), 312–318.

[58] Saleem, N., Balu, A., Jubery, T. Z., Singh, A., Singh, A. K., Sarkar, S., & Ganapathysubramanian, B. (2024). *Class-Specific Data Augmentation for Plant Stress Classification*. [Preprint or Journal details TBA].

[59] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). *Improved Techniques for Training GANs*. NeurIPS, 2234–2242.

[60] Saranya, A., & Subhashini, R. (2023). *A Systematic Review of Explainable Artificial Intelligence Models and Applications: Recent Developments and Future Trends*. [Journal or Conference details TBA].

[61] Schwalbe, G., & Finzel, B. (2024). *A Comprehensive Taxonomy for Explainable Artificial Intelligence: A Systematic Survey of Surveys on Methods and Concepts*. Data Mining and Knowledge Discovery, 38, 3043–3101.

[62] Shorten, C., & Khoshgoftaar, T. (2019). *A survey on image data augmentation for deep learning*. Journal of Big Data, 6(60), 1–48.

[63] Sun, Y., Wong, A. K. C., & Kamel, M. S. (2009). *Classification of imbalanced data: A review*. International Journal of Pattern Recognition and Artificial Intelligence, 23(04), 687–719.

[64] Stepin, I., Alonso, J. M., Catala, A., & Pereira-Fariña, M. (2021). *A Survey of Contrastive and Counterfactual Explanation Generation Methods for Explainable Artificial Intelligence*. IEEE Access, 9, 11974–12001.

[65] Strelcenia, E., & Prakoonwit, S. (2023). *A Survey on GAN Techniques for Data Augmentation to Address the Imbalanced Data Issues in Credit Card Fraud Detection*. In 2023 IEEE Symposium on Computational Intelligence (pp. 1–7).

[66] Temraz, M. N., & Keane, M. T. (2023). *Solving the class imbalance problem using a counterfactual method for data augmentation*. Machine Learning with Applications, 9, 100375.

[67] Terauchi, A., & Mori, N. (2021). *Evolutionary Approach for AutoAugment Using the Thermodynamical Genetic Algorithm*. In 2021 Genetic and Evolutionary Computation Conference (GECCO), 1567–1575.

[68] van Dyk, D. A., & Meng, X.-L. (2001). *The Art of Data Augmentation*. Journal of Computational and Graphical Statistics, 10(1), 1–50.

[69] Veedhi, B. K., Das, K., Mishra, D., Mishra, S., & Behera, M. P. (2025). *Balancing Data Imbalance in Biomedical Datasets Using a Stacked Augmentation Approach with STDA, DAGAN, and Pufferfish Optimization*. [Preprint or Journal details TBA].

[70] Wang, Z., Wang, P., Liu, K., Wang, P., Fu, Y., Lu, C.-T., Aggarwal, C. C., Pei, J., & Zhou, Y. (2024). *A Comprehensive Survey on Data Augmentation*. [Preprint or Journal details TBA].

[71] Wen, Y., Jerfel, G., Muller, R., Dusenberry, M. W., Snoek, J., Lakshminarayanan, B., & Tran, D. (2021). *Combining Ensembles and Data Augmentation Can Harm Your Calibration*. In 2021 International Conference on Machine Learning (ICML), 250–258.

[72] Zelenkov, Y. A., & Lashkevich, E. V. (2024). *Counterfactual Explanations Based on Synthetic Data Generation*. [Preprint or Journal details TBA].

[73] Zhang, Y., & Liu, Q. (2022). *On IoT Intrusion Detection Based on Data Augmentation for Enhancing Learning on Unbalanced Samples*. Future Generation Computer Systems, 133, 213–227.

[UCI, 2023] UCI Machine Learning Repository. (2023). Mammographic Mass Data Set. `https://archive.ics.uci.edu/dataset/161/mammographic+mass`

[Kaggle, 2021] Kaggle. (2021). Pima Indians Diabetes Database. `https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database`

[UCI, 2023] UCI Machine Learning Repository. (2023). Statlog (Australian Credit Approval) Data Set. `https://archive.ics.uci.edu/dataset/143/statlog+australian+credit+approval`