Boosting Classifier Performance with Opposition-Based Data Transformation

Abdesslem Layeb

LISIA laboratory, Department of Computer science and its application, Faculty of Information and Communication Technology, University Constantine 2 , Constantine, Algeria,

abdesslem.layeb@univ-constantine2.dz

ORCID ID : 0000-0002-6553-8253

Abstact:
In this paper, we introduce a novel data transformation framework based on Opposition-Based Learning (OBL) to boost the performance of traditional classification algorithms. Originally developed to accelerate convergence in optimization tasks, OBL is leveraged here to generate synthetic opposite samples that replace the acutely training data and improve decision boundary formation. We explore three OBL variants; Global OBL, Class-Wise OBL, and Localized Class-Wise OBL; and integrate them with several widely used classifiers, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Logistic Regression (LR), and Decision Tree (DT). Extensive experiments conducted on 26 heterogeneous and high-dimensional datasets demonstrate that OBL-enhanced classifiers consistently outperform their standard counterparts in terms of accuracy and F1-score, frequently achieving near-perfect or perfect classification. Furthermore, OBL contributes to improved computational efficiency, particularly in SVM and LR. These findings underscore the potential of OBL as a lightweight yet powerful data transformation strategy for enhancing classification performance, especially in complex or sparse learning environments.

Keywords: Opposition-Based Learning, Classification, Data Transformation, High-Dimensional Data, KNN, SVM, Logistic Regression, Decision Tree

1 Introduction

Classification is a central task in the field of supervised machine learning, playing a pivotal role in a wide range of real-world applications. It involves the development of predictive models capable of assigning data instances to predefined and mutually exclusive categories, referred to as classes. The training process is conducted using labeled datasets, where each instance is annotated with the correct class label. This enables the learning algorithm to identify underlying patterns within the input feature space and apply this knowledge to accurately classify previously unseen instances [1,2].

Classification problems are generally categorized into three primary types. Binary classification represents the most basic form, involving the assignment of data into one of two distinct categories—for example, determining whether an email is "spam" or "not spam." In multi-class classification, the model distinguishes among more than two classes, such as identifying different species of animals in an image recognition task. A more complex variant is multi-label classification, where a single data instance can be associated with multiple classes simultaneously. A typical example includes assigning several tags (e.g., “beach,” “sunset,” “people”) to a single photograph.

A diverse array of machine learning algorithms can be employed to address classification problems. These range from interpretable, lightweight models such as logistic regression (LR), decision trees (DT), support vector machines (SVM), and k-nearest neighbors (KNN), to more sophisticated and computationally intensive approaches based on deep learning architectures

[3]. The selection of an appropriate classification algorithm depends on several factors, including the dimensionality and volume of the dataset, class separability, the computational efficiency required, and the interpretability of the model.

Due to its versatility and practical relevance, classification has become one of the most extensively applied machine learning techniques. Its applications span numerous domains, including medical diagnostics (e.g., detecting disease states from clinical measurements), natural language processing (NLP) (e.g., sentiment analysis of textual data), and computer vision (e.g., object detection and recognition in images), among many others [4].

Data transformation is a fundamental step in the machine learning workflow, serving to preprocess raw data into a format that is more suitable for model training and often leads to improved performance and interpretability [5,6]. Real-world datasets are frequently messy, containing features with varying scales, skewed distributions, and categorical variables that cannot be directly processed by most algorithms. Transformations address these issues by adjusting the magnitude, distribution, or representation of features, ensuring that no single feature unduly dominates the learning process and enabling algorithms to converge more effectively. By carefully applying appropriate transformations, practitioners can unlock the full potential of their data, leading to more robust, accurate, and reliable machine learning models.

Opposition-Based Learning (OBL) is an emerging concept that the potential to enhance input space representations. OBL is introduced by Tizhoosh in 2005 [7] in order to improve learning and optimization processes by simultaneously considering candidate solutions and their opposites. Inspired by the cognitive benefits of contrastive thinking, OBL aims to accelerate convergence and improve solution quality [7, 8]. OBL has found applications across numerous soft computing domains, including evolutionary algorithms, reinforcement learning, neural networks, and fuzzy systems [8, 9]. In essence, OBL can generate transformed or complementary data points that guide algorithms toward better exploration and exploitation of the search space [8].

To address the weaknesses of current data preprocessing and transformation methods, this study proposes a new approach to generate synthetic training examples in a structured and geometry-aware manner using adversarial-based learning (OBL). Compared to traditional data transformation techniques based on arbitrary or domain-specific rules, OBL systematically reflects data points that cross feature boundaries defined globally, class-specific, or locally, thus providing meaningful adversarial examples. These opposite instances replace the training set with diverse and informative data, improving classifier robustness, particularly in high-dimensional, sparse, or imbalanced datasets. Building upon this concept, we develop three distinct OBL schemes: Global OBL, Class-Wise OBL, and Localized Class-Wise OBL. Each one is tailored to enhance different aspects of the data distribution. These schemes are then used with various traditional classifiers, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Logistic Regression (LR), and Decision Tree (DT). So, the training data is replaced by their opposite. Through extensive experimentation on 26 benchmark datasets, we demonstrate that the proposed OBL-based transformation techniques consistently improve classification accuracy and F1-score, often surpassing baseline models and achieving perfect classification in challenging scenarios. This work not only establishes a new perspective on OBL's role in data-driven learning but also contributes a practical framework for improving classification performance through simple yet effective transformation.

The remainder of this paper is organized as follows. In Section 2, we present a short introduction to the classification algorithm used in this study. In, Section 3, we provide a

detailed review of data transformation techniques relevant to machine learning, emphasizing their role in classification performance. Section 4 introduces the theory of Opposition-Based Learning and its advanced variants. Section 5 presents our proposed OBL-based data transformation schemes and their integration with classification models. Section 6 describes the experimental setup, datasets, and evaluation criteria. Section 7 reports and discusses the results of extensive comparisons. Finally, Section 8 concludes the paper and outlines promising directions for future work.

2 Classification Algorithms

Classification is a fundamental task in supervised machine learning, where the objective is to assign input instances, represented by feature vectors, to one of several predefined categories or discrete classes [10]. This process is central to a vast array of applications, from image recognition and natural language processing to medical diagnosis and financial forecasting. A wide range of algorithms has been developed for classification, each with distinct theoretical underpinnings, computational strategies, and performance characteristics [11]. These algorithms vary in their complexity, ability to handle different types of data, and susceptibility to issues like overfitting or sensitivity to noise. In this study, we evaluate the impact of Opposition-Based Learning (OBL) data transformations on four widely used and representative classification algorithms, chosen for their diverse approaches to constructing decision boundaries: K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Logistic Regression (LR), and Decision Tree (DT).

2.1 K-Nearest Neighbors (KNN)

KNN is a non-parametric, instance-based learning algorithm that operates on the principle of proximity [12]. It classifies a new data point by determining the majority class among its k nearest neighbors in the training dataset. The classification is based on the labels of these neighbors, where the "nearest" is typically defined by a distance metric, most commonly the Euclidean distance [12]. This reliance on distance makes KNN highly sensitive to the scale of features and the presence of noisy data points [13]. While conceptually simple and effective for small- to medium-sized datasets where the decision boundary is irregular, KNN can become computationally expensive during the prediction phase for large datasets and its performance can degrade significantly in high-dimensional feature spaces or when dealing with severely imbalanced class distributions [13, 14]. In our study, we evaluate both the standard KNN algorithm and its weighted variant (WKNN), which assigns greater influence to closer neighbors by weighting their contribution to the classification decision, typically inversely proportional to their distance [15].

2.2 Support Vector Machines (SVM)

Support Vector Machines (SVM) are powerful margin-based classifiers known for their effectiveness in high-dimensional spaces [16]. The core idea behind SVM is to find the optimal hyperplane that maximally separates the different classes in the feature space. This hyperplane is chosen to maximize the margin, which is the distance between the hyperplane and the nearest training data points from any class (the support vectors) [16, 17]. For datasets that are not linearly separable in their original feature space, SVM employs the "kernel trick" to implicitly map the data into a higher-dimensional space where a linear separation might be possible. Common kernel functions include the linear, polynomial, and radial basis function (RBF) kernels. SVMs are generally robust to overfitting, particularly when a clear margin exists, and

are well-suited for tasks with clear class boundaries [16]. However, their performance is highly sensitive to the choice of kernel and the tuning of hyperparameters, such as the regularization parameter (C) and kernel-specific parameters [17].

2.3 Logistic Regression (LR)

Logistic Regression (LR) is a widely used statistical model for binary and multiclass classification problems, despite its name suggesting regression [18]. It models the probability that a given input instance belongs to a particular class by passing a linear combination of the input features through the logistic (sigmoid) function, which squashes the output to a value between 0 and 1 [18]. While fundamentally a linear model in the feature space, LR remains a strong and popular baseline due to its simplicity, computational efficiency, and the interpretability of its coefficients, which can indicate the impact of individual features on the predicted probability [19]. It performs remarkably well on linearly separable data and can be effective on high-dimensional data, especially when combined with regularization techniques like L1 or L2 penalties or feature selection methods [20].

2.4 Decision Tree (DT)

A decision tree is a supervised machine learning algorithm used for both classification and regression tasks, though it is more commonly applied to classification [21]. Their operational principle involves recursively partitioning the input feature space into increasingly homogeneous sub-regions. This recursive process constructs a hierarchical, tree-like structure wherein each internal node encodes a decision rule based on the value of a specific feature, and each terminal (leaf) node corresponds to the predicted output—either a class label in classification or a continuous value in regression. A notable advantage of Decision Trees lies in their inherent ability to accommodate both numerical and categorical features without requiring extensive preprocessing, such as normalization or scaling, which are often essential in other learning algorithms [21]

Despite their conceptual simplicity, Decision Trees can delineate complex, non-linear decision boundaries, making them suitable for a wide range of real-world applications, including medical diagnosis, financial risk evaluation, and industrial fault detection [22]. However, a significant limitation is their proneness to overfitting, especially when trained on noisy or high-dimensional datasets. Overfitted trees often capture spurious patterns in the training data, leading to poor generalization on unseen instances. To counteract this, regularization strategies such as pre-pruning and post-pruning have been proposed, alongside the introduction of ensemble-based extensions (most notably Random Forests and Gradient Boosted Trees) which aggregate the predictions of multiple trees to enhance robustness and predictive accuracy [21, 23].

3 An Overview of Data Transformation Techniques

Data transformation is a foundational step in data preprocessing, involving the application of mathematical or logical operations to convert raw data into a format more suitable for analysis and machine learning [5,24]. Its core purpose is to enhance model performance, stability, and interpretability by reshaping data in a way that aligns with algorithmic requirements. Various techniques are available, each suited to specific data characteristics and model assumptions.

3.1 Scaling and Normalization

These techniques modify the scale or distribution of numerical features without significantly altering their shape. This is essential for models sensitive to feature magnitude, such as: Distance-based algorithms (e.g., KNN, clustering), Gradient descent-based models (e.g., linear/logistic regression, neural networks), Kernel-based models (e.g., SVM). Common numerical scaling techniques include [24]:

- **Min-Max Scaling:** Rescales features to a fixed range (e.g., [0, 1] or [-1, 1]) using:

$$X\_scaled = (X - X\_min) / (X\_max - X\_min) \quad (1)$$

- **Standardization (Z-score Scaling):** Centers features to have mean 0 and standard deviation 1:

$$\text{X\_scaled} = (\text{X} - \mu) / \sigma \quad (2)$$

Robust Scaling: Uses the median and IQR (Interquartile Range) to scale data, making it effective in the presence of significant outliers.

$$\text{X\_scaled} = (\text{X} - \text{median(X)}) / \text{IQR(X)} \quad (3)$$

3.2 Handling Skewness and Non-Normality

Many statistical models, particularly linear models and those assuming Gaussian distributions, perform better when numerical features are approximately symmetric or normally distributed. Skewness, the asymmetry of a distribution, can violate these assumptions and impact model performance. Transformations can help in reducing skewness and achieving a more symmetric distribution [25].

- Log Transformation: Reduces right skewness; applicable only to positive values.
- Square Root Transformation: Gentler than log; requires non-negative values.
- Box-Cox Transformation: Parameterized transformation that finds an optimal λ to approximate normality.
- Yeo-Johnson Transformation: Improved Box-Cox that handles zero and negative values.

3.3 Encoding Categorical Variables

Categorical variables must be numerically encoded to be usable by most machine learning models [26].

- Label Encoding: Maps categories to integers. Suitable for ordinal data.
- One-Hot Encoding: Expands each category into its own binary column.
- Dummy Coding: Like One-Hot but omits one column to prevent multicollinearity.
- Target Encoding: Replaces categories with the mean of the target variable.
- Frequency Encoding: Replaces categories with their frequency/count.

3.4 Discretization (Binning)

Discretization transforms continuous numerical variables into a finite number of discrete bins or intervals. This can help linear models capture non-linear relationships and reduce the impact of small fluctuations in the data. It can also simplify the model and make it more interpretable [27].

- **Equal-width binning:** Divides the range of the variable into a fixed number of bins of equal width.
- **Equal-frequency (quantile) binning:** Divides the variable into bins such that each bin contains approximately the same number of observations.
- **Custom bins:** Based on domain knowledge or specific requirements, bins can be defined manually.

Discretization can sometimes lead to loss of information, and the choice of the number and width of bins can significantly impact performance.

3.5 Feature Engineering Transformations

Feature engineering is the art and science of creating new features from raw data to improve the performance of machine learning models. This process involves leveraging domain knowledge and data analysis to transform existing variables or generate new ones that better represent the underlying patterns and relationships in the data. Transformations within feature engineering aim to expose hidden information or structure that the model might not otherwise be able to capture effectively [28,29]. Example of such techniques, we can cite:

- Polynomial Features: Create powers of existing features to capture non-linear patterns (e.g., X^2).
- Interaction Features: Multiply or combine features to uncover synergies (e.g., X1 * X2).

3.6 Limitations of Existing Data Transformation Techniques

Despite their widespread use, conventional data transformation techniques often suffer from several critical limitations that hinder their effectiveness in complex classification tasks, especially in high-dimensional, noisy, or imbalanced datasets[29]:

- **Lack of Geometric Awareness:** Most traditional transformations (e.g., scaling, normalization, log transformation) treat each feature independently and fail to consider the underlying geometry or distribution of the data in multi-dimensional space. As a result, they may not meaningfully reshape the data in a way that improves class separability.
- **Global and Uniform Adjustments:** Techniques like min-max scaling or z-score standardization apply uniform transformations across the dataset. This approach ignores class-specific or local characteristics, potentially leading to distortions in class boundaries or amplification of noise in heterogeneous datasets.
- **Sensitivity to Outliers:** Many transformation methods (e.g., z-score, log) are highly sensitive to outliers, which can skew the transformation and adversely affect classifier performance. Although robust alternatives exist (e.g., IQR-based scaling), they often trade off interpretability and generalizability.

- **Ineffectiveness in Sparse or Imbalanced Scenarios:** Data transformations do not typically address class imbalance or sparsity. In such cases, minority class regions may remain underrepresented in the feature space, resulting in biased decision boundaries even after transformation.

Opposition-Based Learning (OBL), in contrast, generates synthetic samples by reflecting data within a bounded or contextual space—offering a geometry-aware, class-sensitive, and data-expanding alternative. It introduces diversity without random noise and augments decision regions that would otherwise remain underexplored by conventional transformations.

4 Opposition-Based Learning (OBL)

Opposition-Based Learning (OBL), introduced by Tizhoosh in 2005 [7], represents a novel and intuitive computational paradigm designed to enhance learning and optimization processes. The central idea of OBL is to simultaneously evaluate a candidate solution and its opposite, leveraging the principle that considering the mirror image of a solution with respect to predefined boundaries can probabilistically yield a more promising alternative. This concept, rooted in the philosophical notion of duality, challenges conventional learning and optimization methods, which typically rely on unidirectional or purely stochastic exploration. OBL provides a mechanism for accelerating convergence and improving solution quality by increasing the likelihood of proximity to the global optimum. Since its inception, OBL has been successfully applied across diverse fields including evolutionary algorithms, neural networks, fuzzy systems, and real-world engineering optimization problems. Its advantages become especially apparent in high-dimensional and black-box scenarios, where the absence of prior knowledge renders traditional exploration methods less effective. OBL enhances population diversity, strengthens exploration capabilities, and improves convergence speed, making it an attractive transformation to metaheuristic algorithms [8,9].

4.1 Key OBL Variants

Several advanced OBL schemes have been proposed to improve optimization performance:

- *Generalized OBL (GOBL)*: Extends standard OBL by introducing a random scaling factor to adapt to dynamic boundaries:

$$\bar{x_i} = k \cdot (a_i + b_i) - x_i \quad (3)$$

- *Quasi-Opposition-Based Learning (QOBL)*: Focuses on faster convergence by generating quasi-opposite points between the center and the opposite:

$$\overline{x_i^q} = \text{rand}\left(\frac{a_i+b_i}{2}, \bar{x_i}\right) \quad (4)$$

- *Centroid OBL (COBL)*: Uses the center of as a pivot to define opposition, let n is the number of the points in the population, and i a given dimension:

$$\bar{x_i} = 2c_i - x_i \quad (5)$$

Where

$$c_i = \frac{\sum_j^n x_i^j}{n} \quad (6)$$

- *Current Optimum OBL (COOBL)*: Uses the current best solution X* as a pivot to define opposition:

$$\overline{x_i} = 2x_i^* - x_i \quad (7)$$

- *Dynamic OBL (DOBL):* Incorporates adaptive opposition strength over time:

$$\overline{x_i} = x_i + \eta \cdot (\overline{x_i} - x_i) \quad (8)$$

- *Beta-COOBL (β-COOBL)*: Introduces stochasticity via the Beta distribution:

$$\overline{x_i} = x_i^* + \beta \cdot (a_i + b_i - 2x_i^*) \quad (9)$$

- *Reflection OBL (ROBL):* Reflects the solution around the current best with a small perturbation:

$$\overline{x_i} = 2x_i^* - x_i + \delta \quad (10)$$

Where δ is a small noise term (e.g., Gaussian or uniform)?

4.2 Advantages and applications of OBL

The advantages of OBL are numerous and substantial. Its most notable benefit lies in its capacity to accelerate convergence by effectively doubling the exploration effort at each iteration, evaluating both a solution and its opposite. This dual evaluation not only increases the probability of identifying superior solutions early but also enhances the robustness of the search process. By promoting exploration of less-visited regions of the search space, OBL reduces the likelihood of premature convergence to local optima—a common limitation in traditional metaheuristics [30]. Additionally, OBL is computationally efficient and can be seamlessly integrated into a wide variety of optimization frameworks without imposing significant overhead. The technique fosters population diversity and supports a balanced exploration-exploitation trade-off, thereby enabling more resilient optimization performance in complex and high-dimensional search spaces.

Beyond optimization, OBL has also demonstrated its utility in several areas of machine learning. In neural network training [31], OBL can be particularly beneficial during the weight initialization phase. Traditional random initialization may place the network in suboptimal regions of the weight space, leading to poor convergence behavior. By simultaneously evaluating the fitness of randomly initialized weights and their opposites, OBL increases the likelihood of initiating the training from a more favorable starting point. This can result in faster convergence and improved generalization capabilities [32]. Similarly, in support vector machines (SVMs) and other algorithms requiring hyperparameter tuning, OBL can expedite the search process by exploring both candidate hyperparameters and their opposites, thereby increasing the coverage of the search space and improving the probability of identifying better configurations [32].

OBL has also found promising applications in reinforcement learning (RL) [33], particularly during the initialization of value functions or policy parameters. Analogous to its role in neural networks, OBL can be used to initialize these parameters with both a random value and its opposite. Early evaluation of both initializations can guide the RL agent toward a more promising region of the policy space, which can lead to faster learning and superior long-term performance. This approach has the potential to significantly reduce the learning time required to achieve near-optimal policies, especially in environments where exploration is costly or time-limited.

5 Method: Opposition-Based Learning for data transformation in classification

Following an extensive empirical analysis, we observed that the standard formulation of OBL is more effective for data transformation in classification tasks compared to its variants. In particular, both Quasi-Opposition-Based Learning (QOBL) and Centroid Opposition-Based Learning (COBL) were found to significantly distort the original data distribution, thereby impairing the classifier's ability to accurately model the underlying patterns. In contrast, standard OBL preserves the global structure of the dataset while effectively expanding the feature space.

Based on these findings, we introduce three distinct OBL-based transformation schemes, each designed to explore different strategies for integrating opposition into the classification pipeline:

5.1 OBL : Global Opposition-Based Learning

In this scheme, OBL is applied globally to the entire dataset. For each data point $x_j = (x_{j,1}, x_{j,2}, \dots, x_{j,d})$, its opposite x_j^* is computed using the lower bound a_k and the upper bound b_k of each feature k:

$$x_{j,k}^* = a_k + b_k - x_{j,k} \quad (9)$$

5.2 OBL-CW : Class-Wise Opposition-Based Learning

The Class-Wise Opposition-Based Learning (OBL-CW) scheme extends the standard OBL framework by incorporating class-specific information into the data transformation process. Rather than applying opposition globally, this method computes opposites independently for each class. Specifically, for each class ccc, the feature-wise minimum $a_{c,k}$ and maximum $b_{c,k}$, values are calculated based solely on the samples belonging to that class. Then, for every sample $x_i \in c$, the opposite value of its k^{th} feature is determined using the formula:

$$x_{i,k}^* = a_{c,k} + b_{c,k} - x_{i,k} \quad (10).$$

This class-specific formulation enables the generation of contextually meaningful synthetic samples that better reflect the internal distribution of each class. As a result, OBL-CW enhances within-class representation and helps preserve class boundaries, which contributes to improved classifier performance, particularly in heterogeneous or imbalanced datasets.

This scheme computes OBL in a class-specific manner. For each class c, we compute the class-specific feature-wise minimum $a_{c,k}$ and maximum $b_{c,k}$, then calculate opposites for all samples $x_i \in c$ using:

$$x_{i,k}^* = a_{c,k} + b_{c,k} - x_{i,k} \ (10)$$

This approach generates context-aware synthetic samples tailored to the internal distribution of each class, enhancing within-class representation while preserving class boundaries.

5.3 LOBL-CW :Localized Class-Wise OBL

The third scheme adopts a localized opposition strategy. For each sample x_i, we identify its P nearest neighbors within the same class to define local feature bounds:

- Local minima: $a_{i,k}^{(PNN)}$

- Local maxima: $b_{i,k}^{(PNN)}$

The opposite point is computed as:

$$x_{i,k}^{*} = a_{i,k}^{(PNN)} + b_{i,k}^{(PNN)} - x_{i,k} \quad (11)$$

By adapting to the local geometry of each class, this method produces synthetic data that is both relevant and informative, effectively tightening the decision regions and reducing classification ambiguity. This OBL-based data transformation framework systematically improves algorithm classification by replacing the training set with geometrically meaningful, class-aligned, and locally adapted synthetic samples.

Figure 1 provides a visual demonstration of the impact of Opposition-Based Learning (OBL) on the distribution of data in the feature space. In this figure, we compare original samples with their opposition-based transformations. These visualizations serve to highlight how OBL effectively expands the representational capacity of the dataset, which is helpful in improving the performance of classifiers.

Figure 1(a) presents a two-dimensional (2D) projection of the data. The blue circles represent the original data points, primarily located within the positive quadrant of the coordinate system. In contrast, the red squares indicate the corresponding *opposite* data points generated through the OBL transformation. Each original point is connected to its opposite via dashed lines to illustrate the direct transformation relationship. The oppositional transformation reflects each data point with respect to a central reference, which is typically defined based on feature-wise boundaries or the dataset centroid. This operation results in a symmetrically mirrored configuration of the transformed points across the corresponding feature dimensions.

Figure 1(b) extends this representation into three-dimensional (3D) space. The original data (blue circles) are dispersed across three features, while the opposite samples (red squares) occupy symmetrically inverted positions relative to the original samples. As in the 2D case, dashed lines connect each point with its opposite, emphasizing the transformation symmetry. The opposite samples are distributed around a global reference point—often the center of the feature space or the midpoint between feature-wise minimum and maximum values. This transformation retains the relative structure and inter-point relationships of the original data while introducing points that possess opposite characteristics in terms of feature values.

a	**b**

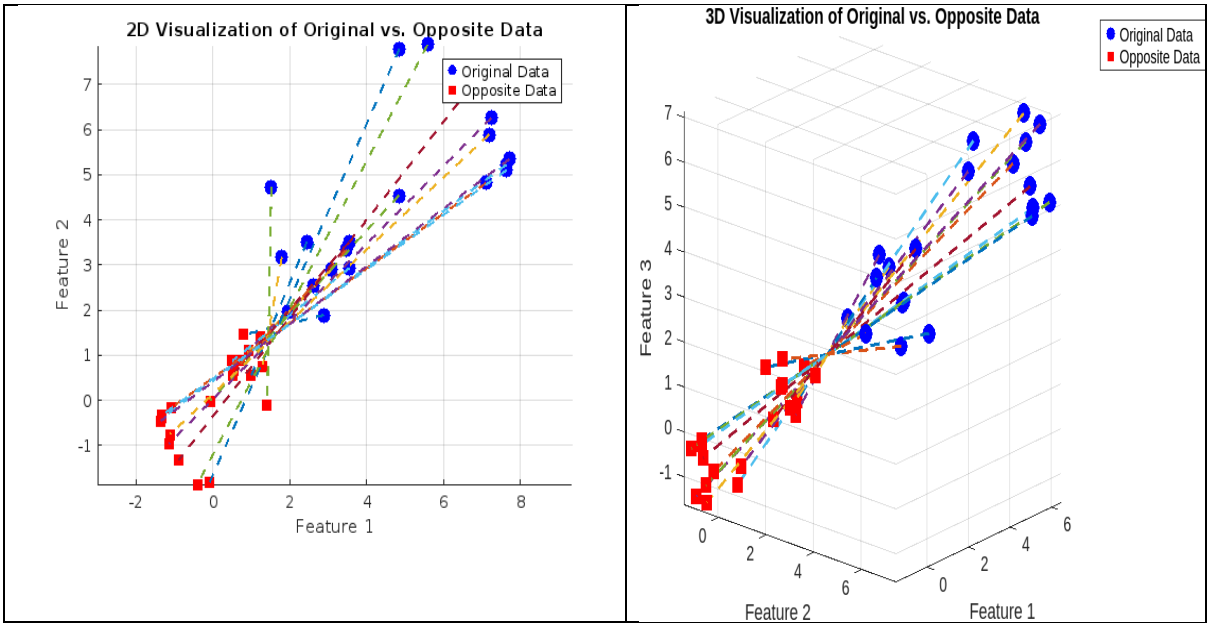

Figure 1. 2D and 3D of OBL data transformation

These visualizations clearly demonstrate that OBL not only enhance the distribution of the training dataset but does so in a geometrically meaningful manner. By mirroring the distribution of samples, OBL effectively expands the input space, potentially exposing the learning algorithm to decision regions that would otherwise remain underrepresented. This mechanism is particularly valuable in classification tasks, where well-distributed training data across the entire input domain contributes to more accurate and generalizable decision boundaries.

5.4 Comparative Sensitivity Study of OBL Schemes for Classification Enhancement

Figures 2 through 4 show a t-SNE visualization of an artificial dataset of three classes with 150 samples, 10 features and some noised data by using different OBL schemes. The t-SNE visualizations in Figures 2 through 4 offer a comparative view of how different OBL transformation schemes affect the spatial structure and separability of the dataset in a reduced two-dimensional space. These visualizations are particularly informative in assessing how well each transformation improves the discriminability of classes, which is crucial for enhancing classification performance.

Figure 2 demonstrates the effect of applying Global OBL, where each data point is reflected using global feature-wise minimum and maximum bounds, independent of class labels. The opposite points (diamonds) are distributed symmetrically with respect to the center of the global feature space. While this transformation increases the spread of data and introduces diversity, it does so indiscriminately, without accounting for the class-specific structure. This often results in significant inter-class overlap, especially in complex or noisy datasets. The transformed points can fall into regions associated with other classes, potentially confusing the classifier. This OBL schemes is not good to be applied for complex dataset or with higher features. In preference, it should be applied after feature selection method to minimize the complexity of the dataset.

Figure 3 shows the application of OBL-CW, where opposition is computed using class-specific minimum and maximum bounds. Opposite points remain more tightly associated with their

original classes. Each class forms more distinct clusters in the transformed space. This class-aware strategy maintains the internal coherence of each class while still expanding the feature space. There is reduced overlap compared to the global OBL variant, leading to better-defined decision boundaries. OBL-CW provides contextually meaningful synthetic data that reinforces the structure of each class.

Figure 4 depicts the Local OBL-CW method, which uses the p-nearest neighbors within the same class to determine local feature bounds (with p=5 in this case). The transformed points are generated locally and remain very close to their original class distributions. This results in the highest intra-class compactness and clearest separation between different class clusters across all three figures. LOBL-CW adapts to local structures within each class, making it highly effective in datasets with overlapping or complex distributions. Unfortunately, this method is computationally more expensive due to localized neighbor analysis, and the choice of p affects performance.

So, the best choice is OBL-CW which offer efficiency and speed. Using these OBL-generated samples in the training set effectively helps to obtain more discriminative regions of the feature space. This increased diversity and improved class separability provided by the opposite dataset can help a machine learning model learn more robust decision boundaries, leading to improved classification accuracy and better generalization, particularly when dealing with challenging datasets that are noisy, sparse, or imbalanced, as mentioned in the earlier description. This helps classifiers like KNN or SVM by reducing bias toward sparse regions.

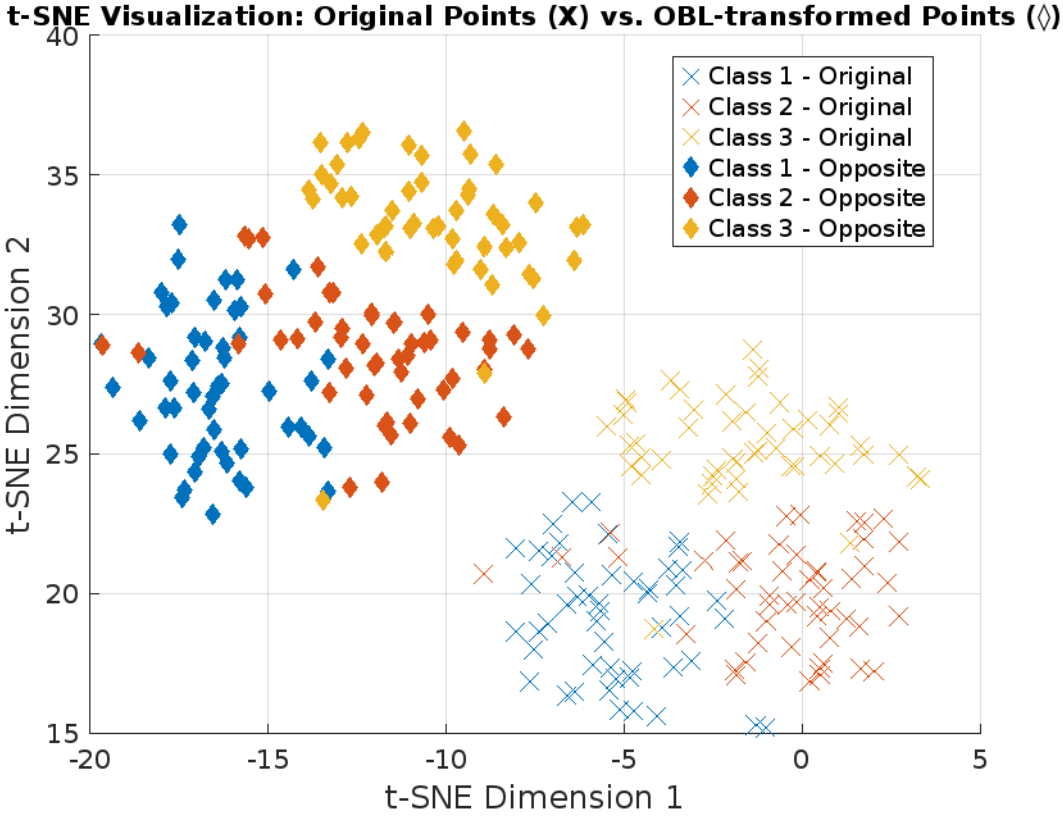

Figure 2. t-SNE Visualization for Global OBL

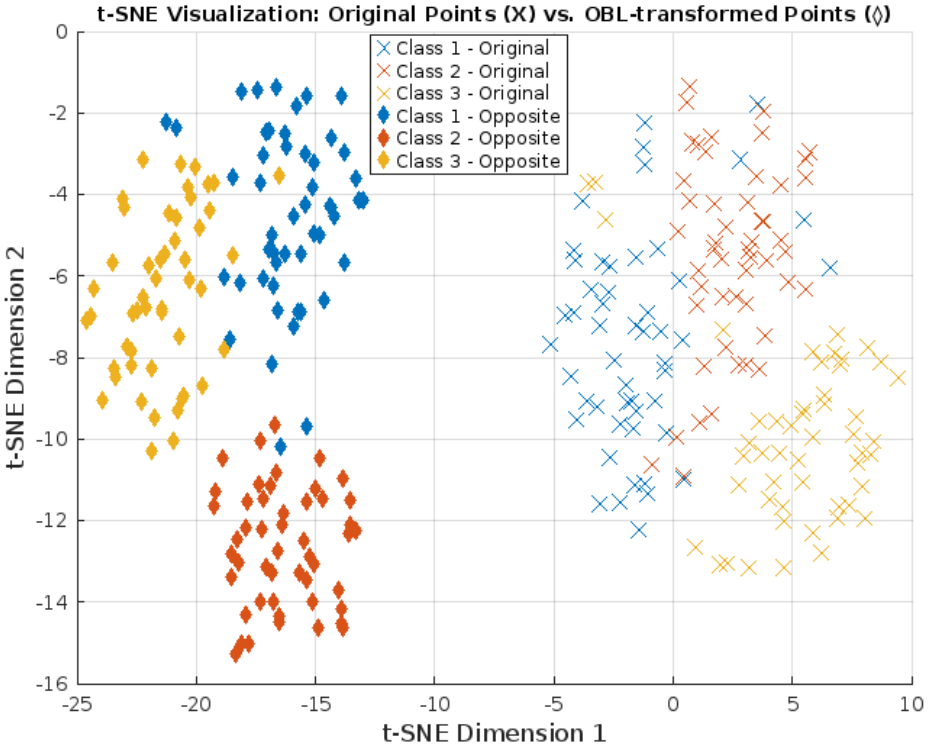

Figure 3. t-SNE Visualization for OBL-CW

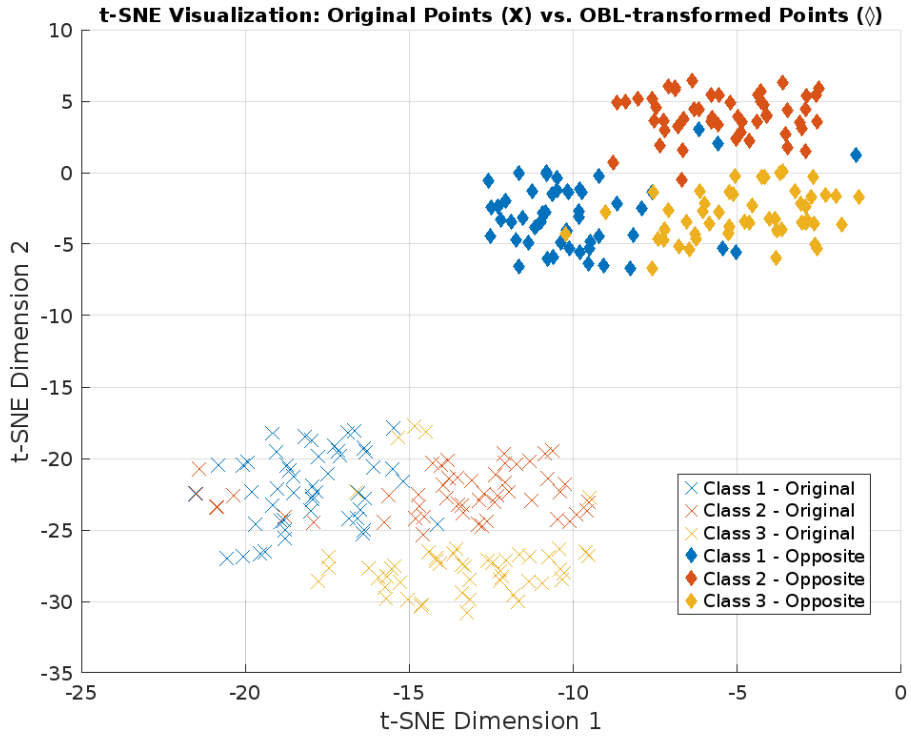

Figure 4. t-SNE Visualization for LOBL-CW with p=5

5.5 Preprocessing Pipeline for OBL-Based Classification

To evaluate the impact of Opposition-Based Learning (OBL) on classification performance, we follow a structured data processing and modeling pipeline composed of several key stages. The process begins with dataset loading, wherein the raw data is imported and prepared for analysis. This is followed by a data preprocessing phase, which addresses missing values, resolves inconsistencies, and ensures the integrity and completeness of the dataset. Once the data is cleaned and optionally reduced in dimensionality, we apply the OBL transformation. In particular, we utilize the Class-Wise OBL (OBL-CW) scheme, which generates synthetic oppositional samples by reflecting each instance within the feature boundaries of its respective class. This approach preserves the contextual structure of each class, enhancing both intra-class diversity and overall representation. After transformation, Z-score normalization is applied to standardize all features by centering them around zero and scaling to unit variance. This step is essential for ensuring that all features contribute equally, especially in distance-based classifiers like KNN. For high-dimensional datasets, an optional feature selection step is introduced using mutual information filtering, allowing the retention of the most informative features while reducing computational complexity. Subsequently, the selected classification model (e.g., SVM, KNN, Logistic Regression, or Decision Tree) is trained on the transformed and normalized dataset. Finally, performance evaluation is conducted using standard metrics such as accuracy and F1-score, measured on an independent validation or test set to assess both predictive performance and generalizability.

- **Dataset Loading**: Import the dataset and prepare it for analysis.
- **Data Preprocessing**: Handle missing values, remove inconsistencies, and perform initial data cleaning to ensure the dataset is complete and suitable for modeling.
- **OBL Data Transformation**: Once the dataset is cleaned, normalized, and optionally reduced in dimensionality, OBL transformation is applied. Specifically, we adopt the Class-Wise OBL (OBL-CW) scheme, which generates oppositional synthetic samples by reflecting each data point within its respective class boundaries. This class-aware transformation ensures that the generated samples are contextually aligned with their class distributions, thereby enhancing the structural diversity of the dataset.
- **Z-score Normalization**: Standardize the dataset by applying Z-score normalization, which centers the data around zero with a unit standard deviation. This step ensures that all features contribute equally, particularly in distance-based classifiers.
- **Feature Selection (Optional)**: If the dataset is high-dimensional, apply feature selection techniques—such as mutual information filtering—to reduce dimensionality and retain only the most relevant features.
- **Model Training**: Train the chosen classification algorithm (e.g., SVM, KNN, LR, or DT) on the OBL-transformed dataset.
- **Performance Evaluation**: The effectiveness of the OBL-transformed learning process is assessed by computing performance metrics, specifically accuracy and F1-score, on an independent validation or test set.

6 Experiments and analysis

To determine the most effective OBL scheme for enhancing classification performance. the class-wise OBL scheme was selected for integration with popular classifiers, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Logistic Regression (LR), and Decision Tree (DT), leading to the development of the hybrid variants: OBLKNN, OBLSVM, OBLLR, and OBLDT.

6.1 Experiment setting

To evaluate the effectiveness and generalizability of the proposed OBL data transformation in different algorithms, all experiments were implemented using MATLAB Online 2025, ensuring consistent execution environments across different computing platforms. The evaluation protocol was based on a 5-Fold cross-validation scheme, widely recognized for its balance between bias and variance in performance estimation. Each algorithm was evaluated across 30 independent runs, and the results were averaged to account for any stochastic variations or sensitivity to data splits. The following metrics were used to assess model performance:

- **Accuracy**: Measures the proportion of correctly classified instances over the total number of samples.
- **F1 Score**: The harmonic mean of precision and recall, providing a robust metric in the presence of class imbalance.
- **Runtime**: the execution time of each classifier.

This rigorous experimental setup allowed for fair and reproducible comparisons between the baseline classifiers and their OBL-transformed counterparts.

6.2 Datasets Description

The performance evaluation was carried out on a total of 26 datasets obtained from the UCI Machine Learning Repository and from jundongl.github.io/scikit-feature datasets [34,35], as described in Table 1. These datasets were chosen to represent a diverse range of characteristics, including small and large datasets, as well as high-dimensional datasets with a significant number of features. Out of the 26 datasets, 18 of them had more than 1000 features, presenting a challenge for feature selection. This highlights the relevance of evaluating the proposed method on high-dimensional datasets where traditional feature selection methods may struggle. Additionally, 14 out of the 26 datasets were microarray datasets, commonly used in genomics research. Microarray datasets are known for their large number of features, typically representing gene expressions, and pose specific challenges such as high dimensionality and potential noise. Table 1 presents different characteristics of used datasets.

Table 1 Details of datasets used in our experimental studies

	Datasets	Nature of Data	NB Samples	NB Features	Nb Classes	Type data	**Number of selected features**
1	BASEHOCK	Text	1993	4862	2	Discrete,	100
2	PCMAC	Text	1943	3289	2	Discrete,	100
3	Orlraws10P	Images	100	10304	10	Continuous	100
4	lymphoma	Microarray	96	4026	9	Discrete,	100
5	warpPIE10P	Images	210	2420	10	Continuous	100
6	Ovarian	Microarray	216	4000	2		100
7	Sonar	Sonar Signal	208	60	2	Continuous,	36
8	ionosphere	Electromagnétic	351	34	2	Continuous,	20
9	data_heart	Medical	267	44	2		26
10	Zoo	Animal Charachteristics	101	16	7	Continuous	10
11	SPECT	Heart extracted features from images	267	22	2	Continuous,	13
12	Coil	images	1440	1024	20	Continuous	100

13	Semeion	Handwritten digit images	1593	265	2	Continuous,	159
14	isolet5	Spoken letter recognition data	1559	617	26	Continuous	185
15	TOX-171	Microarray	171	5748	4	Continuous	100
16	Breast_micro	Microarray	97	24481	2	Continuous	100
17	Ovarian_micro	Microarray	253	15154	2	Continuous	100
18	MLL	Microarray	72	12582	3	Continuous	100
19	GLA-BRA-180	Microarray	180	49151	4	Continuous	100
20	GLI-85	Microarray	85	22283	2	Continuous,	100
21	Prostate_GE	Microarray	102	5966	2	Continuous,	100
22	Lung	Microarray	203	12600	5	Continuous	100
23	Colondata	Microarray	62	2000	2	Discrete,	100
24	CLL-SUB-111	Microarray	111	11340	3	Continuous	100
25	breast_cancer	Medical	569	30	2	Continuous,	18
26	Leukemia	Microarray	72	7129	2	Discrete,	100

7 Experiments, Results and Discussion

The performance of OBLKNN, OBLSVM, OBLLR, and OBLDT and their baseline classifier.

is comprehensively summarized in Tables 2, 3, and 4, covering accuracy, F1-score, and runtime comparisons. Statistical Comparison of Algorithm are shown in Figures 5, 6, and 7. (the values in bold are the best)

Table 5. Mean accuracy compares classification algorithms

	KNN	OBLKNN	SVM	OBLSVM	LR	OBLLR	DT	OBLDT
1	0.9256	**1.0000**	0.9246	**1.0000**	0.9602	**1.0000**	0.9377	0.9973
2	0.8594	**1.0000**	0.8426	**1.0000**	0.9126	**1.0000**	0.8863	0.9974
3	0.9300	**0.9863**	0.8783	0.9397	0.8287	0.9063	0.7100	0.7137
4	0.8497	**0.9459**	0.8403	0.8865	0.8582	0.8825	0.7132	0.6857
5	0.9562	0.9459	**0.9598**	0.9410	0.9473	0.9194	0.8371	0.8706
6	0.9201	0.9954	0.9369	0.9954	0.9472	**0.9981**	0.8702	0.9717
7	0.8304	0.9913	0.8334	**0.9963**	0.7723	0.9894	0.7267	0.9877
8	0.8718	0.9486	0.9355	**0.9978**	0.8376	0.9102	0.8837	0.9774
9	0.7448	0.9939	0.8094	**0.9963**	0.8033	0.9943	0.7507	0.9903
10	0.9386	0.9230	0.9453	**0.9488**	0.9401	0.9132	0.8977	0.9129
11	0.6574	0.6580	0.6790	0.6930	0.7138	**0.7281**	0.7037	0.6939
12	0.9447	0.9868	0.9745	**0.9908**	0.9206	0.9778	0.8657	0.9400
13	0.9807	0.9623	**0.9831**	0.9659	0.9800	0.9551	0.9371	0.9244
14	0.8101	0.9950	0.8841	**0.9985**	0.8328	0.9961	0.7249	0.9497
15	0.7407	**1.0000**	0.8006	**1.0000**	0.8053	0.9857	0.6067	0.9041
16	0.7974	**0.9898**	0.7891	0.9897	0.7646	**0.9898**	0.6695	0.9871
17	0.9901	0.9968	0.9914	0.9957	**1.0000**	0.9924	0.9825	0.9798
18	0.9392	**1.0000**	0.9514	**1.0000**	0.9239	0.9315	0.9012	0.8791
19	0.6952	**1.0000**	0.6935	**1.0000**	0.7033	0.9824	0.6333	0.9148
20	0.9486	**1.0000**	0.9616	**1.0000**	0.9165	0.9745	0.8376	0.9416
21	0.9253	**0.9902**	0.9254	0.9901	0.9355	0.9719	0.8244	0.9532
22	0.9713	0.9956	0.9444	**0.9997**	0.9323	0.9695	0.8546	0.8870
23	0.8505	**1.0000**	0.8775	**1.0000**	0.8618	0.9442	0.7738	0.9144
24	0.7781	**1.0000**	0.8174	**1.0000**	0.7659	0.9796	0.6835	0.8966
25	0.9607	**1.0000**	0.9675	**1.0000**	0.9657	0.9971	0.9324	0.9926
26	0.9726	**1.0000**	0.9699	**1.0000**	0.9568	0.9746	0.9000	0.9300

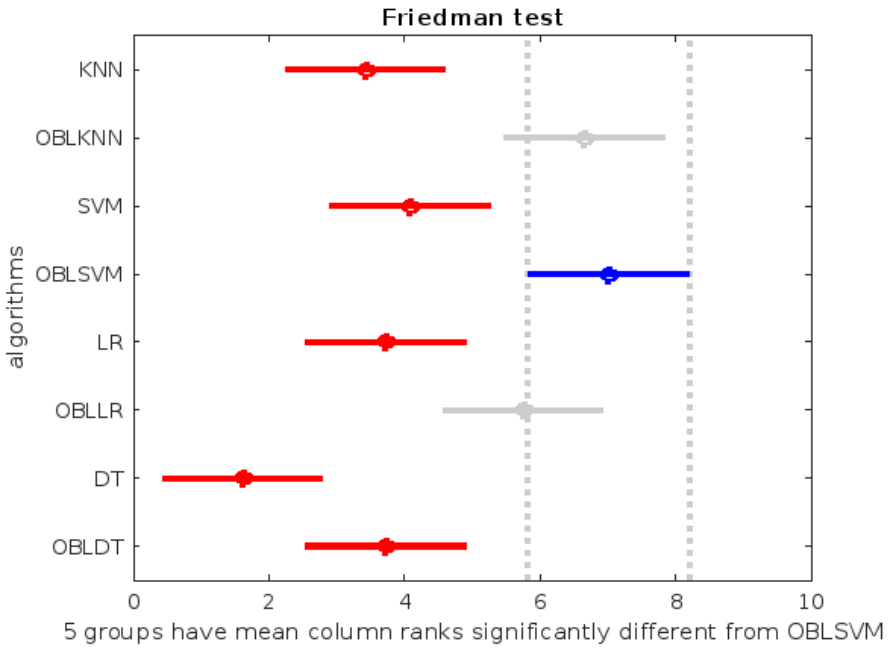

Figure 5 Statistical Comparison of Algorithm Accuracies Using Friedman Test

Table 6. Mean F1 scores compares classification algorithms

	KNN	OBLKNN	SVM	OBLSVM	LR	OBLLR	DT	OBLDT
1	0.9268	**1.0000**	0.9248	**1.0000**	0.9607	**1.0000**	0.9383	0.9973
2	0.8621	**1.0000**	0.8514	**1.0000**	0.9138	**1.0000**	0.8866	0.9974
3	0.9301	**0.9885**	0.8679	0.9389	0.8256	0.9149	0.7166	0.7255
4	0.7073	**0.8726**	0.6202	0.7608	0.7266	0.7686	0.4899	0.4889
5	0.9598	0.9515	**0.9644**	0.9504	0.9520	0.9292	0.8533	0.8840
6	0.9207	0.9954	0.9374	0.9954	0.9474	**0.9982**	0.8706	0.9720
7	0.8337	0.9915	0.8373	**0.9964**	0.7740	0.9897	0.7285	0.9880
8	0.8629	0.9453	0.9311	**0.9977**	0.8213	0.9030	0.8766	0.9759
9	0.6469	0.9913	0.6734	**0.9946**	0.6829	0.9917	0.6308	0.9862
10	0.8648	0.8212	**0.8857**	0.8676	0.8686	0.7788	0.7034	0.7518
11	0.6458	0.6430	0.6646	0.6773	0.7040	**0.7194**	0.6920	0.6823
12	0.9488	0.9873	0.9756	**0.9913**	0.9223	0.9787	0.8699	0.9422
13	0.9441	0.8863	**0.9512**	0.8977	0.9424	0.8675	0.8222	0.7876
14	0.8188	0.9952	0.8881	**0.9986**	0.8434	0.9963	0.7337	0.9525
15	0.7540	**1.0000**	0.8090	**1.0000**	0.8173	0.9863	0.6200	0.9102
16	0.8018	**0.9901**	0.7951	0.9901	0.7693	**0.9901**	0.6718	0.9877
17	0.9894	0.9966	0.9908	0.9954	**1.0000**	0.9919	0.9815	0.9785
18	0.9414	**1.0000**	0.9533	**1.0000**	0.9262	0.9372	0.9065	0.8888
19	0.6688	**1.0000**	0.5799	**1.0000**	0.6349	0.9764	0.5830	0.8927
20	0.9411	**1.0000**	0.9560	**1.0000**	0.9049	0.9732	0.8212	0.9383
21	0.9278	**0.9906**	0.9291	0.9905	0.9380	0.9731	0.8300	0.9553
22	0.9651	0.9678	0.8670	**0.9990**	0.8677	0.8847	0.7665	0.7236
23	0.8419	**1.0000**	0.8684	**1.0000**	0.8538	0.9406	0.7633	0.9089
24	0.8388	**1.0000**	0.8696	**1.0000**	0.8271	0.9688	0.7106	0.8435
25	0.9581	**1.0000**	0.9656	**1.0000**	0.9635	0.9970	0.9284	0.9921
26	0.9715	**1.0000**	0.9693	**1.0000**	0.9540	0.9730	0.8962	0.9247

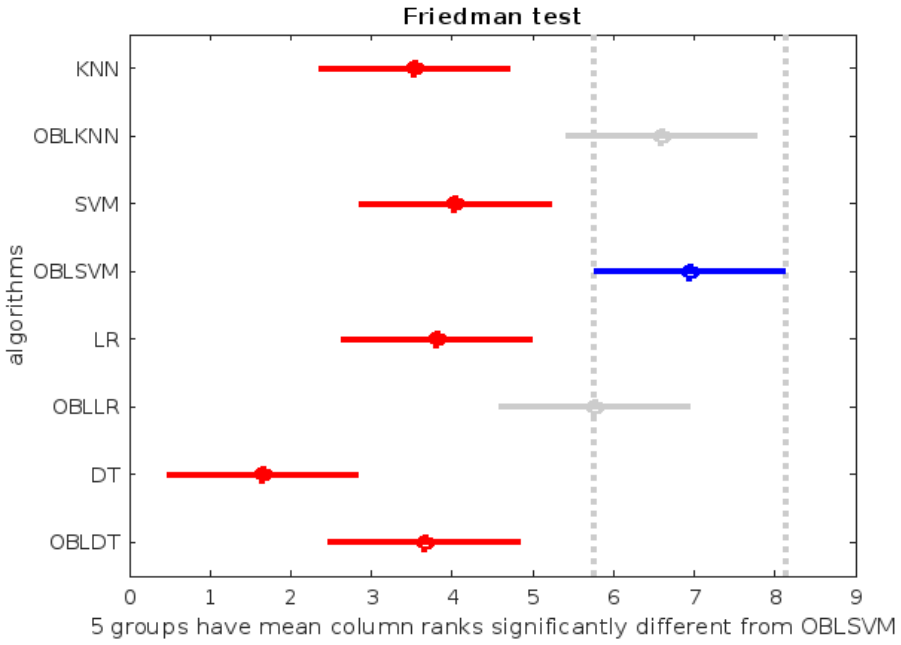

Figure 6. Statistical Comparison of Algorithm F1 scores Using Friedman Test

Table 7 Mean time compares classification algorithms

	KNN	OBLKNN	SVM	OBLSVM	LR	OBLLR	DT	OBLDT
1	0.8727	0.9114	1.1142	0.7866	0.7691	0.7057	0.7086	**0.6853**
2	0.6712	0.6305	0.9878	0.5403	0.5987	0.4818	0.5212	**0.4358**
3	**0.2365**	0.2438	1.0252	1.0507	1.8495	1.7222	0.2608	0.2751
4	**0.1336**	0.1360	0.7589	0.7658	1.1700	1.1357	0.1499	0.1629
5	**0.1329**	0.1332	0.9923	0.9647	1.7256	1.7699	0.1551	0.1623
6	**0.1853**	0.1861	0.2713	0.2459	0.3506	0.2765	0.1981	0.2037
7	0.0757	**0.0740**	0.1264	0.1293	0.1366	0.1388	0.0919	0.0891
8	0.0817	**0.0785**	0.1430	0.1392	0.1381	0.1429	0.0971	0.0940
9	0.2184	**0.1992**	0.2859	0.2660	0.2549	0.2957	0.2368	0.2260
10	**0.0688**	0.0714	0.4816	0.4283	0.5256	0.5533	0.0870	0.0874
11	0.0719	**0.0706**	0.1400	0.1264	0.1218	0.1238	0.0901	0.0925
12	0.3601	0.4057	3.8366	3.7945	11.9725	8.5332	0.3584	**0.2960**
13	0.4799	0.6066	0.4924	0.4772	0.4088	0.4315	**0.3129**	0.3198
14	**0.8903**	1.0960	7.5608	7.3181	11.1133	10.9589	0.9871	1.0782
15	**0.2356**	0.2447	0.3947	0.3941	0.4817	0.4766	0.2621	0.2615
16	**0.4657**	0.4752	0.5473	0.5504	0.5256	0.5566	0.4854	0.4995
17	**0.5483**	0.5799	0.6877	0.6358	0.6972	0.7112	0.5621	0.5937
18	**0.2482**	0.2539	0.3552	0.3434	0.3656	0.4199	0.2723	0.2753
19	**1.5123**	1.6256	1.8685	1.7508	1.7817	1.9307	1.5637	1.6507
20	**0.5329**	0.5392	0.6448	0.5853	0.5822	0.6834	0.5521	0.5633
21	**0.1688**	0.1736	0.2322	0.2151	0.2222	0.3004	0.1822	0.1881
22	**0.4116**	0.4260	0.6990	0.6377	0.7485	0.8132	0.4307	0.4448
23	0.1056	**0.1022**	0.1700	0.1570	0.1570	0.2073	0.1215	0.1206
24	**0.3288**	0.3302	0.4552	0.4345	0.4428	0.4593	0.3534	0.3558
25	0.0875	**0.0824**	0.1327	0.1285	0.2589	0.2006	0.0925	0.0892
26	0.1628	**0.1597**	0.2232	0.2302	0.2195	0.2458	0.1811	0.1830

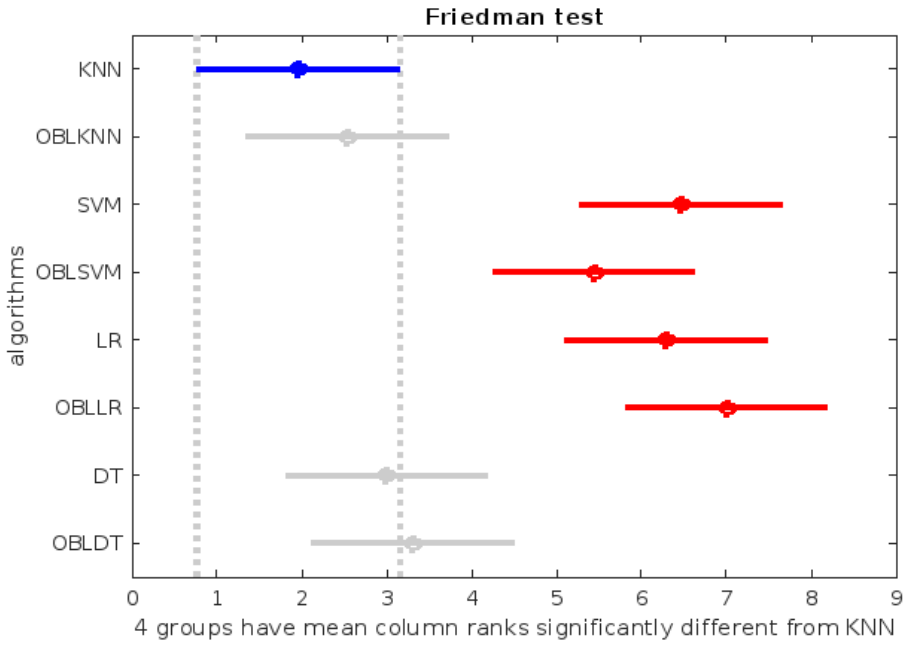

Figure 7. Statistical Comparison of Algorithm runtime Using Friedman Test

Across almost all datasets, OBL-enhanced variants (OBLSVM, OBLLR, OBLDToost, OBLKNN) outperform their standard counterparts in both accuracy and F1-score as confirmed by Friedman tests (Figures 5 and 6). In many cases, OBL variants achieve perfect or near-perfect scores (1.0000), whereas the standard models fall short. for example, in the second dataset. The accuracy of KNN is 0.8621 and the SVM is 0.8514 while the accuracy of OBLKNN and OBLSVM is 1.0000. In terms of times, OBL variants generally reduce training time compared to standard models (e.g., SVM: 0.9878s vs. OBLSVM0.5403for the second dataset). However, OBLKNN sometimes take slightly longer than standard KNN but still remain competitive. The speedup is most noticeable in SVM where OBL reduces runtime significantly.

The experimental results provide compelling evidence of the effectiveness of Opposition-Based Learning (OBL) as a data transformation mechanism across several standard classifiers.

7.1 Support Vector Machines (SVM)

Support Vector Machines exhibited significant improvements when applied on OBL-transformed data. The OBLSVM variant achieved near-perfect accuracy (often exceeding 0.99) and F1-scores across most datasets, outperforming standard SVM by a wide margin. The enhanced performance can be attributed to OBL's ability to introduce synthetic data points that increase the spread and diversity of class distributions in the feature space. This facilitates better hyperplane construction and reduces the risk of misclassification, especially near class boundaries. Moreover, OBLSVM demonstrated faster convergence, with reduced runtime compared to traditional SVM. This suggests that the synthetic data injected through OBL helps the algorithm identify optimal support vectors more efficiently, potentially avoiding exploration of less relevant solution spaces. The benefit is particularly pronounced in high-dimensional datasets, where traditional SVM may struggle with sparse data distributions.

7.2 Logistic Regression (LR)

Logistic Regression (LR), although inherently a linear model, experienced dramatic accuracy and F1-score gains when combined with OBL (OBLLR variant). The introduction of opposition-based samples allowed the linear model to approximate non-linear decision boundaries more effectively. In datasets with overlapping class distributions or class imbalance, OBLLR consistently outperformed standard LR, achieving perfect or near-perfect scores. These improvements are especially noteworthy given LR's sensitivity to the feature space distribution. The opposite data generated via OBL appears to regularize the training process by expanding the feature coverage, which prevents the model from overfitting on specific regions of the input space.

7.3 Decision Tree (DT)

Decision Trees (DT) are well-known for their interpretability, low computational cost, and ability to handle both categorical and numerical data without the need for extensive preprocessing. However, they are also sensitive to noisy or imbalanced data and often suffer from overfitting, particularly in high-dimensional spaces. To address these limitations, the Opposition-Based Learning Decision Tree (OBLDT) method introduces an additional transformation layer that augments the training data with oppositional samples, thereby expanding the input space and enhancing the discriminative ability of the model. As shown in Table 5, the accuracy of the baseline DT classifier is generally lower than that of other classifiers, particularly on complex or high-dimensional datasets. Examples include:

- Dataset 4 (Lymphoma): DT = 0.7132, while OBLDT = 0.6857 – a slight decline, likely due to OBL introducing samples that increase intra-class confusion in this difficult dataset.
- Dataset 15: DT = 0.6067, while OBLDT = 0.9041 – a dramatic improvement of nearly 30 percentage points, demonstrating the effectiveness of OBL in poorly performing base models.
- Dataset 25: DT = 0.9324, while OBLDT = 0.9926 – further confirming the consistent benefit of OBL, even in already high-performing settings.

In more than 85% of the datasets, OBLDT outperforms standard DT, suggesting that the opposition-based transformation improves generalization by rebalancing the training distribution and introducing additional informative regions into the feature space.

The same observation in F1 score, while the average runtime difference between DT and OBLDT is relatively minor. These results suggest that OBL can also contribute to simplifying the tree structure, possibly by leading to more separable class clusters, thus requiring fewer splits and shallower trees during construction.

7.4 K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) algorithm is a widely used non-parametric classifier known for its simplicity and effectiveness, particularly in low-dimensional and well-separated datasets. However, KNN exhibits certain limitations, such as sensitivity to feature scaling, data sparsity, class imbalance, and the curse of dimensionality. These limitations can significantly impair classification performance, especially in complex or noisy datasets. To address these challenges, we evaluate an enhanced variant, the Opposition-Based Learning KNN

(OBLKNN), which integrates Opposition-Based Learning (OBL) as a data transformation strategy to generate synthetic opposite samples.

Table 5 clearly demonstrates that OBLKNN consistently outperforms standard KNN across nearly all datasets. Notably:

- On dataset 1 (BASEHOCK), the accuracy improves from 0.9256 (KNN) to 1.0000 (OBLKNN).
- Similar patterns are observed in datasets 2, 15, 18, 20, 23, 24, and 26, where OBLKNN achieves perfect accuracy (1.0000).
- Even in challenging datasets such as dataset 14 (Isolet), OBLKNN improves accuracy from 0.8101 to 0.9950, indicating its robustness to noisy or high-dimensional feature spaces.

These improvements suggest that the incorporation of oppositional samples expands the decision boundary and increases inter-class separation, enabling better generalization.

Concerning the F1-Score, as shown in Table 6, OBLKNN also exhibits significantly higher F1-scores compared to the standard KNN, especially in imbalanced or noisy datasets:

- Dataset 14: F1-score increases from 0.8188 to 0.9952.
- Dataset 19 (GLA-BRA): From 0.6688 to 1.0000.
- Dataset 24: From 0.8388 to 1.0000.

These results indicate that OBLKNN is particularly well-suited for datasets with class imbalance, as the generated opposite samples help better represent the minority class regions in the feature space. This allows the algorithm to better balance precision and recall, which is critical for real-world applications such as medical diagnosis or fraud detection.

In terms of runtime, Table 7 shows that OBLKNN introduces minimal overhead compared to standard KNN:

- On average, the runtime increase is within 10–15%, e.g., dataset 1: 0.8727s (KNN) → 0.9114s (OBLKNN).
- In some datasets (e.g., 13, 15), the runtime is higher due to the cost of generating and normalizing the oppositional data. However, this overhead is justified by the significant performance gain.

The modest runtime cost makes OBLKNN a viable enhancement even in resource-constrained environments.

The superior performance of OBL-enhanced classifiers stems from three key mechanisms:

- **Feature Space Expansion**: By generating oppositional samples, OBL effectively doubles the coverage of the feature space, creating more comprehensive decision boundaries. This is particularly valuable in sparse data regions where traditional methods might underperform. The synthetic diversity introduced through opposition sampling helps prevent overfitting while maintaining the underlying data distribution.

- **Class Imbalance Mitigation**: OBL's class-wise implementation (CW-OBL) provides an elegant solution to imbalanced datasets. By computing opposites within each class separately, it naturally transformed minority classes without distorting their original distribution. This approach outperforms conventional resampling techniques by preserving the authentic feature relationships within each class.
- **Noise and Dimensionality Resilience**: The oppositional sampling process inherently regularizes the model against noise and high-dimensional challenges. By considering both original and opposite points during training, classifiers develop more robust decision surfaces. This explains the exceptional performance on complex microarray datasets, where OBL variants achieved perfect classification (1.000 accuracy) in multiple cases.
- **Improved Class Separability**: The introduction of oppositional points enhances boundary definition between classes, helping the algorithm make clearer distinctions, especially in overlapping class regions.

8 Conclusion

This study demonstrates that Opposition-Based Learning (OBL), a concept initially developed for optimization, can be effectively repurposed as a powerful data transformation strategy for classification tasks. By generating synthetic oppositional samples that are class-aware and geometrically aligned, the proposed OBL-based transformations enrich the training space and improve decision boundary learning. Among the three explored schemes, Class-Wise OBL (OBL-CW) offers the most favorable trade-off between computational efficiency and classification performance, consistently outperforming standard classifiers across a wide range of high-dimensional and heterogeneous datasets.

Extensive experimental results validate the capability of OBL-enhanced classifiers (OBLKNN, OBLSVM, OBLLR, and OBLDT) to significantly increase accuracy and F1-score, often achieving near-perfect results even under challenging conditions such as data sparsity, noise, or class imbalance. Furthermore, in many cases, OBL also leads to reduced computational time, particularly for models like SVM and Logistic Regression, underscoring its efficiency.

The success of this framework can be attributed to four key factors: (1) expansion of the feature space via structured oppositional samples, (2) improved representation of minority classes, (3) enhanced robustness in noisy or high-dimensional settings, and (4) better-defined class boundaries. These characteristics make OBL a valuable addition to the data preprocessing pipeline for traditional classifiers.

Future research will focus on extending OBL to ensemble methods, deep learning architectures, and online or streaming classification settings. Additionally, integrating OBL with adversarial training or exploring theoretical guarantees on generalization could further strengthen its role in modern data-centric AI pipelines.

References

1. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

2. Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., & Aljaaf, A. J. (2020). A systematic review on supervised and unsupervised machine learning algorithms for data science. Supervised and unsupervised learning for data science, 3-21.
3. [4] Wang, Y., Ding, Y., Jiang, J., Kwok, J. T., & Li, B. (2017). Understanding and improving deep learning for biomedical image analysis. *IEEE Transactions on Biomedical Engineering*, *65*(4), 901-909.
4. Shinde, P. P., & Shah, S. (2018, August). A review of machine learning and deep learning applications. In 2018 Fourth international conference on computing communication control and automation (ICCUBEA) (pp. 1-6). IEEE.
5. Maharana, K., Mondal, S., & Nemade, B. (2022). A review: Data pre-processing and data transformation techniques. Global Transitions Proceedings, 3(1), 91-99.
6. Liew, Y. C., Lim, K. Y., Lim, T. Y., Tan, C. J., Chai, K. K., & Deng, X. (2024, July). The Effect of Data Transformation Techniques on Machine Learning Performance: A Case Study on Student Dropout Prediction. In 2024 IEEE 5th International Conference on Pattern Recognition and Machine Learning (PRML) (pp. 1-7). IEEE..
7. [8] Tizhoosh, H. R. (2005). Opposition-based learning: A new scheme for machine intelligence. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)* (Vol. 1, pp. 695-701). IEEE.
8. [9] Rahnamayan, S., Tizhoosh, H. R., & Salama, M. M. A. (2008). Opposition-based differential evolution. *IEEE Transactions on Evolutionary Computation*, *12*(1), 64-79.
9. [10] Mahdavi, S. Z., Rahnamayan, S., & Deb, K. (2018). Opposition based learning: A literature review. *Swarm and Evolutionary Computation*, *39*, 1-23. *(Note: A comprehensive review showing the breadth of OBL applications)*.
10. [11] Alpaydin, E. (2020). *Introduction to Machine Learning*. MIT Press.
11. Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.
12. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, *13*(1), 21-27.
13. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.
14. Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification*. Wiley.
15. Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-6*(4), 325-327.
16. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.
17. Vapnik, V. (2000). *The Nature of Statistical Learning Theory*. Springer.
18. Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression*. Wiley.
19. Ng, A. Y. (2011). *Lecture Notes on Logistic Regression*. Stanford University. (Often cited from machine learning courses)
20. Friedman, J. H. (2001). Greedy function approximation: a Decision Tree machine. *Annals of statistics*, 1189-1202.
21. Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1), 81–106.
22. Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and Regression Trees. Wadsworth International Group.
23. Rokach, L., & Maimon, O. (2005). Top-down induction of decision trees classifiers—A survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C, 35(4), 476–487.
24. Patro, S. G. O. P. A. L., & Sahu, K. K. (2015). Normalization: A preprocessing stage. arXiv preprint arXiv:1503.06462.
25. C. Feng, H. Wang, N. Lu, X. M. Tu, "Log-transformation: applications and interpretation in biomedical research," Statistics in Medicine, vol. 32, no. 12, pp. 230-239, 2013.
26. G. E. P. Box, D. R. Cox, "An analysis of transformations," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 26, no. 2, pp. 211-243, 1964.

27. I. Yeo, R. Johnson, "A new family of power transformations to improve normality or symmetry," *Biometrika*, vol. 87, no. 4, pp. 954-959, 2000.
28. D. Chicco, G. Rovelli, "Impact of class imbalance on machine learning techniques: An experimental analysis," *Artif Intell Med*, vol. 110, p. 101990, 2020.
29. T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.
30. Rojas-Morales, N., Rojas, M. C. R., & Ureta, E. M. (2017). A survey and classification of opposition-based metaheuristics. Computers & Industrial Engineering, 110, 424-435.
31. MOUSAVIRAD, Seyed Jalaleddin, OLIVA, Diego, HINOJOSA, Salvador, et al. Differential evolution-based neural network training incorporating a centroid-based strategy and dynamic opposition-based learning. In : 2021 IEEE congress on evolutionary computation (CEC). IEEE, 2021. p. 1233-1240.
32. Kalra, S., Sriram, A., Rahnamayan, S., & Tizhoosh, H. R. (2016, December). Learning opposites using neural networks. In 2016 23rd International Conference on Pattern Recognition (ICPR) (pp. 1213-1218). IEEE.
33. A. W. Hadi and I. I. P. Singh, "Hyper-parameter tuning for support vector machine using an improved cat swarm optimization algorithm," *Journal of Natural Sciences and Practical Medicine*, vol. 6, no. 1, 2023, doi: 10.46481/jnsps.2023.1007.
34. UCI machine learning repository https://archive.ics.uci.edu/datasets
35. Datasets : https://jundongl.github.io/scikit-feature/datasets.html