STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **13**, March 2025, pp 1162–1191.
Published online in International Academic Press (www.IAPress.org)

Fuzzified Clustering and Sample Reduction for Intelligent High Performance Distributed Classification of Heterogeneous Uncertain Big Data

Sherouk Samir Moawad [1,*], Magued Osman [1], Ahmed Shawky Moussa [2]

[1]*Statistics Department, Faculty of Economics and Political Sciences, Cairo University, Giza, Egypt*
[2]*Computer Science Department, Faculty of Computers and Artificial Intelligence, Cairo University, Giza, Egypt*

Abstract
Big Data has had a significant impact on various fields. In the realm of Big Data Analytics, it is imperative to constantly search for new statistical techniques to extract insights from large and diverse datasets, in a timely manner. This paper introduces a Fuzzified Clustering technique with sample reduction and distributed Parallel Classification (FCPC). The objective of this classification methodology is to represent Big Data using a smaller sample size while effectively capturing underlying structures in the data to enhance classification performance. Benchmark Big Data sets are used to compare FCPC with traditional classifiers that can be used if the whole training data fits in memory. Four classification techniques were evaluated in terms of classification evaluation metrics. The proposed model demonstrated improved classification predictive power with a sample reduction of approximately 90%, leading to enhanced performance and potential reductions in computational resources. Additionally, a comparison with state-of-the-art methods shows that FCPC outperforms existing techniques in terms of classification performance. A case study on a real-world Big Data set further investigates the impact of cluster number determination and the number of classes on FCPC's performance, demonstrating its robustness and adaptability.

Keywords Big Data, Fuzzified Clustering, Classifier Ensemble, Weighted Subsampling, Parallel Classification, Sample Reduction, Veracity.

DOI: 10.19139/soic-2310-5070-2275

1. Introduction

Recently, Big Data has made a significant impact across numerous fields, primarily due to its ability to gather, store, manage, and manipulate vast amounts of diverse data. Big data is defined as a data source with specific characteristics including Volume, Variety, and Veracity. The volume of data is usually too massive to be stored or processed using a single computing node so, High Performance Computing (HPC) systems are used. Variety refers to the fact that a single dataset can contain multiple formats, while Veracity represents the unreliability and uncertainty inherent in some sources of data [16, 27]. This uncertainty is further compounded by the typically imprecise and incomplete nature of data. Consequently, intelligent approaches have introduced Soft Computing techniques that utilize and combine Fuzzy Computing with other computing machine learning paradigms to model the uncertainty and produce computationally intelligent reasoning [21, 8]. While technologies such as Map-Reduce, Hadoop, and Spark have been developed to manage Big Data efficiently through distributed computing [37, 7], the complexity of Big Data also requires novel machine learning approaches for effective data analysis. In this context, classification techniques that integrate clustering and subsampling are gaining attention. The following sections will give a brief overview of Classification, Clustering, and Cluster-based Classification.

*Correspondence to: Sherouk Samir Moawad (Email:shorouk.moawad@feps.edu.eg). Statistics Department, Faculty of Economics and Political Sciences, Cairo University, 1 Gamaa Street, Giza, Egypt (12613).

ISSN 2310-5070 (online) ISSN 2311-004X (print)

1.1. Classification

Classification is a key aspect of supervised learning, where observations in the training dataset are labeled. The task assumes that the set of factors (covariates, X_s) influences each observation's assignment to a specific category of the categorical variable Y. The primary goal of any classifier is to learn a function mapping the inputs X_s to the outputs Y, enabling the prediction of Y for new observations based on the values of their predictors. Classification can be either binary or multiclass. In binary classification, the goal is to classify observations into one of two categories, where $C = 2$ is the number of classes, and the set of classes is $L = \{l_1, l_2\}$. In multiclass classification, the goal is to classify observations into one of several categories, where $C > 2$ and $L = \{l_1, l_2, \ldots, l_C\}$. Common machine learning classifiers include Logistic Regression (LR), Naive Bayes (NB), Artificial Neural Networks (ANN), Support Vector Machines (SVM), K-Nearest Neighbor (KNN), Decision Trees (DT), and Random Forests (RF) [29, 36].

1.2. Clustering

Clustering is a key unsupervised learning technique aimed at exploring inter-relationships among characterizing variables to identify similarities and dissimilarities in observations. Prominent clustering techniques include K-Means, K-Medoids, Fuzzy C-Means Clustering, Gaussian Mixture Model (GMM), Probabilistic Distance Clustering, and Density-based Clustering [6].

Heterogeneity often arises when data is collected from various sources, representing different sub-populations, each exhibiting unique features. In smaller datasets, data points in distant sub-populations may be deemed "outliers." However, the vast volume of Big Data presents an opportunity to reconsider these outliers as clusters, enabling sophisticated statistical techniques to model heterogeneity among sub-populations [14, 16]. This study contends that merging clustering with classification can enhance classification accuracy and overall performance in heterogeneous Big Data, as has been successfully demonstrated in other data challenges.

1.3. Cluster-based Classification

Cluster-based classification methods have attracted attention in the literature for their ability to enhance classification accuracy by leveraging the underlying structure of the data. These methods typically partition the dataset into distinct clusters based on similarities among observations before applying classification techniques. For instance, [1] proposed a framework that first clusters the data and then integrates cluster IDs into the classified data, enriching the classification model with insights gained from the clustering phase.

Similarly, the field of machine learning has seen the emergence of cluster-based classifier ensembles. As defined by [23], a classifier ensemble consists of a group of base classifiers whose collective decisions are combined to yield a single outcome, aiming to improve performance beyond that of any individual classifier. When clustering techniques are applied for data partitioning into homogenous observation sets, these ensembles are termed cluster-based classifier ensembles. However, applying cluster-based ensembles to Big Data introduces two significant challenges: volume and veracity. The large volume often necessitates sampling a smaller subset of the dataset for modeling [44], while veracity, which pertains to data uncertainty and reliability, complicates analysis. To address these challenges, we propose intelligent subsampling from each cluster before training the base classifiers, which manages both data volume and the uncertainty associated with overlapping clusters, thereby enhancing classification performance.

The motivation behind this research stems from the challenges posed by uncertainty, overlap, and the volume and veracity of Big Data in classification tasks, where traditional methods often struggle to provide accurate and efficient solutions. The primary objective of this paper is to introduce Fuzzified Clustering with Weighted Subsampling and Distributed Parallel Classification (FCPC), a cluster-based ensemble technique designed to address these challenges. FCPC builds upon the fuzzy cluster-based classification ensemble proposed by [32] (under review), which utilizes fuzzified clustering to represent data uncertainty and overlapping characteristics. In this work, we further enhance the approach by introducing intelligent subsampling to reduce dataset size and preserve essential information.

This methodology uses Weighted Random Sampling (WRS) to create efficient subsamples, which are then used to train classifiers in parallel. By combining their predictions, FCPC enhances classification accuracy, reduces computational time, and ensures scalability for Big Data sets. This work contributes a comprehensive framework for Big Data classification, advancing both theory and practical applications in handling complex datasets.

The rest of the paper is organized as follows: Section 2 presents related works, Section 3 introduces the proposed FCPC methodology, Section 4 presents and discusses the experimental results, and the paper concludes in the final section.

2. Related work

Big Data modeling can be divided into two main approaches: Divide and Conquer or subsampling. The divide and conquer approach involves three steps: first, partitioning a Big Data set into K blocks; second, processing each block separately, and finally, aggregating the solutions from each block to form a final solution for the entire dataset. On the other hand, subsampling-based methods begin by selecting a small subset of the full data and then using this sample to estimate the model parameters in the ”model-fitting step” [44]. One well-known paradigm that follows the divide and conquer strategy is Map Reduce. [38] introduced a Map Reduce-based algorithm that utilizes multiple classifiers. In this approach, each mapper handles the training for a specific block, while the same test data is distributed to all mappers. Once all mappers complete their tasks, the outputs are shuffled and sorted. The reduce phase then determines the class of the test sample by selecting the label with the highest probability from all the mappers. [34] conducted a detailed review focusing on recent classification techniques and platforms for Big Data. Their work cataloged various approaches to Big Data classification, emphasizing traditional and machine learning-based techniques. Additionally, they highlighted platforms and technologies such as Sklearn, TensorFlow, and Weka which are widely used to enable distributed and parallel processing of massive datasets. Their study briefly mentioned challenges like heterogeneous data, high volume and velocity, garbage data, imbalanced classifications, and data uncertainty. Regarding classification ensembles in Big Data, [24] proposed a distributed heterogeneous boosting-inspired ensemble classifier (DHBoost) for big data classification, leveraging the MapReduce paradigm within Apache Spark to ensure scalability and efficiency. Building on their previous work, HBoost, the authors introduced a method that distributes the training data into partitions, assigns different learning algorithms to each, and generates weak classifiers using the AdaBoost.M1 technique. Classifiers are then sorted, pruned based on entropy to enhance diversity, and aggregated using weighted majority voting to form the final model. In addition, a recent study by [42] addressed the classification task of large-scale data within a Spark cluster. The authors proposed a new ensemble-based paradigm that splits a large data file into smaller blocks and applies two ensemble methods: bagging and boosting. The paradigm uses four predictive models combining these splitting methods with ensemble techniques.

The following subsections review previous research in the emerging areas of sampling techniques and soft computing methods applied to Big Data classification. These approaches have gained significant attention due to their ability to handle the complexities and challenges posed by large, heterogeneous datasets.

2.1. Soft computing for Big Data Classification

Soft computing techniques have become crucial in addressing the complexities and uncertainties of Big Data. [18] emphasized their role in aligning problem-solving techniques more closely with human thinking, particularly in contexts involving ambiguities. They highlighted advancements such as fuzzy logic, neural networks, and genetic algorithms, which have shown significant potential in enhancing industrial prediction and classification. Building on this foundation, [45] provided a comprehensive review of fuzzy set techniques in Big Data processing, outlining their ability to handle uncertainties across all phases of Big Data workflows. They identified fuzzy sets as essential for representing and managing imprecise information and highlighted their integration with granular computing to address challenges in areas such as clustering, classification, and decision-making. The study emphasized the potential of fuzzy sets to enhance scalability and efficiency by addressing complexities such as high volume, velocity, and variety. Additionally, the authors pointed out the opportunities for integrating fuzzy sets with other tools, such as neural networks and rough sets, to create sophisticated solutions for Big Data challenges. [11] provided an overview of the latest distributed fuzzy classification models for Big Data using MapReduce. In their work, [11] overviewed several methods for Big Data classification in the context of fuzzy or DT-based techniques. Chi-based Fuzzy Rule-Based Classification System for Big Data (Chi-FRBCS-BigData) is a fuzzy rule-based classification system designed for Big Data, employing a Chi-squared measure for feature selection. Chi-based Big Data Classification (CHI_BD) is a Chi-based fuzzy classification model optimized for large-scale datasets. Distributed Fuzzy Aggregation Classification with Fuzzy Feature Processing (DFACFFP) utilizes fuzzy DTs with an advanced feature selection process. Distributed Parallel Adaptive Evaluation Scheme with Randomized Classification Strategy (DPAES_RCS) focuses on distributed parallel algorithms for efficient fuzzy classification. Distributed Parallel Adaptive Evaluation Scheme with Fuzzy Decision Trees - Global Learning (DPAES-FDTGL) is an extension of DPAESRCS that integrates fuzzy DTs with a global learning framework. The Multi-way Fuzzy Decision Trees (Multiway FDT) method enhances DT models by utilizing multi-way splits rather than binary ones, which can improve classification accuracy on complex datasets. Finally, Fuzzy Multi-objective Decision Trees 5 (FMDT5) combines fuzzy DTs with a multi-objective optimization approach to further enhance classification performance. Another notable contribution is the MR-FRBDT algorithm, a fuzzy rule-based DT that integrates parallel computing and ensemble learning [33]. By leveraging the MapReduce framework and reducing parameters, MR-FRBDT effectively handles large-scale datasets, reduces computing time, and avoids memory limitations. [5] introduced FDR2-BD, a meta-learning framework designed to recommend optimal feature and instance selection strategies specifically for tabular Big Data. This approach emphasizes both computational and classification efficiency, utilizing DT from Spark's MLlib as base classifiers. In the context of Big Data, hierarchical classification systems are often employed to model complex relationships between class labels, typically using tree-like structures. [47] introduced a novel method for addressing imbalanced binary classification in large-scale datasets by combining undersampling and ensemble learning. Unlike traditional undersampling approaches that randomly reduce the majority class and risk losing important data points, they proposed a fuzzy data reduction scheme to select informative instances from clustered subsets of the majority class. The process maintains the intrinsic distribution of the data and prevents information loss. They further introduced a method that employs a fuzzy integral for classifier fusion, modeling relationships between base classifiers more effectively than traditional ensemble techniques.

These studies collectively underline the importance of combining fuzzy logic, DTs, and distributed processing for Big Data classification, highlighting the need for methods like the proposed FCPC algorithm, which specifically addresses uncertainties and overlaps in data.

2.2. Sampling for Big Data Classification

Data Reduction, Instance Selection, Sub Data Selection, or Subsampling are all terms used to describe the process of taking a representative sample of a Big Data training set. Data Reduction is defined as preprocessing algorithms that simplify and clean raw data while retaining as much information as possible. According to [30], Data Reduction techniques can be classified into Instance Selection (IS) or Instance Generation (IG). IG methods generate artificial data points to better represent the training set, while IS involves obtaining a subset of the training data that does not contain redundant or noisy examples but still maintains almost the same predictive accuracy as the original dataset. [31] presented a comprehensive survey of data partitioning and sampling methods to support Big Data analysis. They addressed several challenges in Big Data analysis, particularly in distributed computing environments. These include bottlenecks in cluster computing due to data volume exceeding memory capacity, difficulties in efficient online sampling for distributed and unbounded data, and the need for statistically aware data partitioning methods to improve representativeness and reduce inefficiencies in traditional partitioning approaches. They discuss fundamental strategies, such as data partitioning and sampling, essential for speeding up Big Data computations and increasing scalability in cluster computing environments. Their work outlines various partitioning techniques, including range partitioning and random partitioning. They also explore data partitioning on Hadoop clusters. In the context of sampling, they highlighted classical methods, including Simple Random Sampling (SRS) and stratified sampling, noting their limitations when applied to large, distributed datasets.

The Fast Condensed Nearest Neighbor (FCNN) technique introduced by [2] is one of the earliest IS techniques developed for Big Data. The initial algorithm, Condensed Nearest Neighbor (CNN), involved subsetting the entire data to sampled data S and the remaining training data T. The algorithm used S to iteratively train the KNN classifier, transferring misclassified points from T to S. The algorithm terminated when no points were transferred from T to S. In FCNN, S is initialized to the centroids of the classes in the training set T, and points move from T to S during iterations if they have a different class label than their nearest point in S. FCNN has been shown to outperform other techniques in terms of time and accuracy. [10] applied a similar concept to update Scalable SRS and Subsampled Double Bootstrap sampling techniques, classifying items in the sample as either well or misclassified. [15] proposed an IS approach based on classifier ensemble. Their algorithm involves partitioning the data into several rounds, where each round partitions the original dataset into disjoint subsets. The IS algorithm is then applied to each subset separately, and the subsets of data from each round are considered as base classifiers in an ensemble. [20] presented the Parallel Sampling method based on HyperSurface algorithm (PSHS), designed to handle big data with uncertain distributions. Their approach addresses the challenge of uncertain structures in large datasets. The PSHS method works in three stages using MapReduce. It first assigns each sample to a region based on its dimensional values, then determines whether the region is pure or not, and finally samples pure regions for further classification. [28] addressed the challenge of efficiently handling large, heterogeneous data sets generated in manufacturing systems. They proposed a cluster-based data filtering method, which reduces large data sets into smaller, informative subsets. The method partitions raw data into clusters and proportionally selects a subset from each cluster, balancing the trade-off between data size and information preservation. A new filtering information criterion (FIC) is introduced to help determine the appropriate proportion of data to retain, ensuring that the data filtered maintains high quality.

2.3. Cluster-Based Subsampling for Big Data Classification

Several studies in the literature have utilized cluster-based subsampling techniques for data classification purpose. [13] introduced a cluster classifier that used clustering algorithms to reduce the size of large datasets by using cluster centroids to create a smaller dataset representing the Big Data set. [46] enhanced the KNN algorithm to overcome the shortcomings of the traditional KNN algorithm in processing large datasets. They proposed a KNN algorithm based on cluster denoising and density cropping of training set samples to accelerate the KNN search speed and enhance the classification efficiency of the KNN algorithm. [41] employed K-Means clustering to decrease the original large-scale dataset by iteratively repeating the clustering step and selecting representative instances from each cluster, resulting in a reduced sample representing the dataset. Subsequently, a multi-stage SVM classifier is applied to the resulting labeled reduced data. [26] suggested an approach for classifying Big Data using Classification by Clustering. They developed a classification model based on the similarity of instances rather than the class labels of the instances. The Big Data set was segmented into several small sub-datasets, and clustering techniques and DTs were constructed from these sub-datasets. Finally, all the DTs were merged to create a single DT that closely resembled the tree that would have been generated if the whole data had fit in memory. [35] introduced a cluster-oriented instance selection algorithm that runs K-Means Clustering on the dataset and selects instances from the centers and borders of the clusters with a predetermined selection rate.

The literature reviewed and presented utilized clustering and classification for reducing the volume and complexity of datasets. However, it did not account for capturing the uncertainty in data and the resulting potential overlapping observations between multiple clusters.

In this paper, we propose a Cluster-Based Classifier Ensemble based on fuzzified Sample Reduction that can be implemented with any clustering and classification algorithm to achieve higher performance in parallel classification. The proposed methodology aims to model Big Data with the intrinsic and inherent uncertainty characterized as Veracity in Big Data. Additionally, the proposed methodology aims to facilitate analytics on parallel and distributed computing environment, reducing the need for higher computational power.

3. FCPC Methodology

The FCPC methodology consists of three stages: fuzzified clustering, sample reduction, and parallel classification and prediction. After splitting the initial set into a training set and a test set, the training set is clustered based on interactions among attributes, resulting in K clusters. Subsequently, a fuzzy membership function (FMF) is constructed based on the distance from each cluster centroid. The FMF values serve as weights that will be utilized in the subsampling stage using WRS. Next, parallel classifiers are applied to each subsample, which can be processed independently across distributed systems. Finally, in the prediction step, a test point is classified based on a specific model determined by the maximum membership value to cluster K in the initial stage.

3.1. Framework

To simplify the presentation of the methodology framework, we have summarized it in a diagram in Figure 1 and Algorithm 1 Pseudo code.

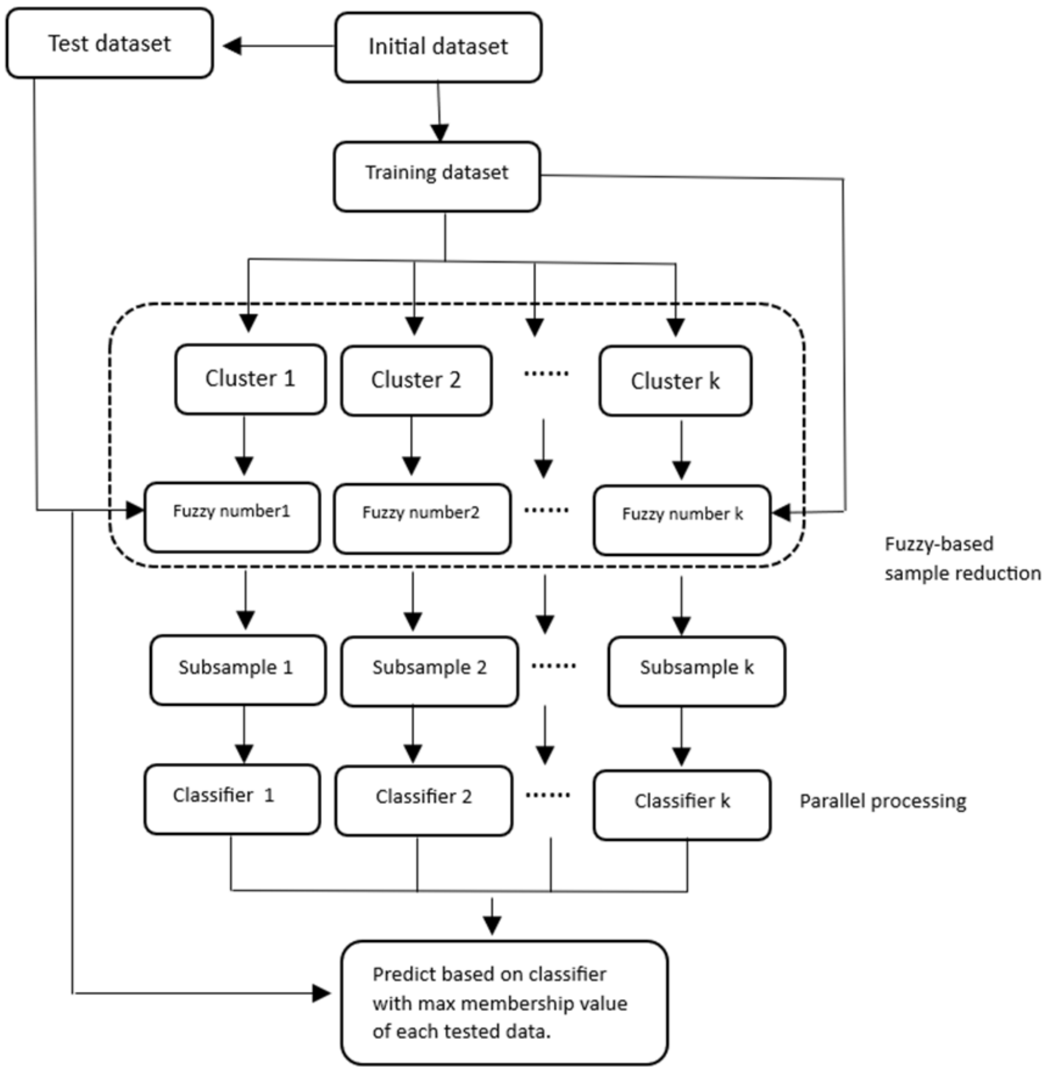

Figure 1. The Framework of FCPC

Algorithm 1 FCPC

```
Input: Training set $S_{\text{train}}$, Test set $S_{\text{test}}$
Output: The labels prediction $y$ in $S_{\text{test}}$
Steps:
  1. Cluster $S_{\text{train}}$ to a predetermined number of clusters $k$
  2. For $j = 1$ to $k$ do:
      (a) Generate FMF values $\mu_j$
      (b) Generate WRS based on $\mu_j$
      (c) Train a classifier based on cluster subsample j
     End For
  3. Generate the final prediction result based on the combination strategy
```

3.2. Fuzzified Clustering

Fuzzification is the process of converting a crisp input into a fuzzy value. The clusters do not have mutually exclusive, crisp boundaries. They overlap, and some observations may belong to multiple clusters simultaneously with varying degrees of belonging.

3.2.1. Clustering. As previously mentioned, in addition to being massive in size, Big Data is typically heterogeneous in nature. Clustering is used to uncover hidden patterns in the training dataset by dividing it into K clusters. In this current research, we conducted experiments on the K-Medoids clustering technique using the Clustering Large Applications (Clara) extension, as introduced by [25], which is designed to efficiently handle large datasets. Clara applies K-Medoids to random samples of the data in a repetitive manner.

However, there is no restriction on the type of clustering to be used. The methodology can be extended to other clustering techniques such as K-Means, Fuzzy C-Mean Clustering, and Probabilistic Distance Clustering. In this research, we used the Elbow technique [17] to determine the optimal k a priori for clustering in phase 1.

3.2.2. Fuzzification. This step involves building a FMF that defines how input data are mapped to a fuzzy degree of membership between 0 and 1, as described by [39]. Each observation in the dataset has a fuzzy membership value in each of the K clusters, ranging from 0 (The observation does not belong to the cluster at all) to 1 (The observation belongs to the cluster with 100% certainty). Other observations will have membership values in the range of $[0, 1]$. Accordingly, the first step is to develop a trapezoidal FMF with k sets of parameters, one for each cluster. The FMF utilized here is the same as the one originally proposed by [?]. Figure 2 and Equation 1 present the graphical and analytical representation of the FMF.

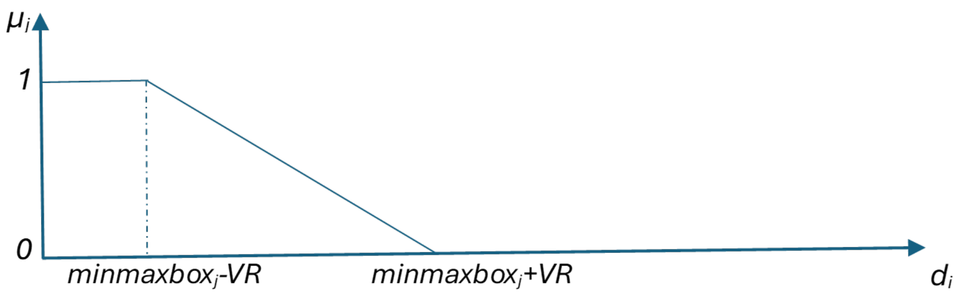

Figure 2. Graphical representation of FMF

$$\mu_j = \begin{cases} 1 & \text{if } d_{ij} \le \mathit{minmaxbox}_j - VR, \\ 1 - \left(\frac{d_{ij} - \mathit{minmaxbox}_j + VR}{2VR}\right) & \text{if } \mathit{minmaxbox}_j - VR < d_{ij} \le \mathit{minmaxbox}_j + VR, \\ 0 & \text{if } d_{ij} > \mathit{minmaxbox}_j + VR. \end{cases} \tag{1}$$

where

- μ_j is the Fuzzy Membership Function (FMF) associated with cluster j, $j = 1, 2, \ldots, k$.
- $d_{ij}, \quad i = 1, \ldots, N$, is the distance of each observation to the center of cluster j, and N is the training data size.
- $\mathit{minmaxbox}_j = \min(\mathit{max}_j, \mathit{maxbox}_j)$, where max_j is the maximum distance of items belonging to cluster j to the center of cluster j, and maxbox_j is the upper limit in a box plot of distances of items belonging to cluster j to the center of cluster j. Here, $\mathit{maxbox} = Q3 + 1.5 \cdot \mathit{IQR}$, where $\mathit{IQR} = Q3 - Q1$; $Q3$ is the third quartile of distances of items belonging to cluster j to the center of cluster j, and $Q1$ is the first quartile of distances of items belonging to cluster j to the center of cluster j.
- $VR = \frac{\mathit{IQRI}_j}{\mathit{IQRO}_j}$, where n_j is the size of cluster j. Here, ***IQRI***: IQR Inner is a measure of variation within a cluster, specifically the IQR of distances to the center of cluster j for items belonging to cluster j. Conversely, ***IQRO***: IQR Outer is a measure of variation across clusters, representing the IQR of distances to the center of cluster j for all items in the dataset.

The higher the VR ratio, the greater the overlap between clusters, resulting in a broader FMF that permits membership values for observations from outside the crisp cluster.

3.3. Subsampling Phase and Size Determination

The main objective of this phase is to reduce the sample size by selecting K representative samples from the training set, with each representing a specific cluster. The observations in the sample must meet the following criteria:

- The majority of the sample should consist of observations with minimal distance to the corresponding cluster core.
- Overlapped observations from other clusters should participate in the specific subsample with lower percentages.
- Outlying observations should be eliminated.
- Observations with large distances from the cluster centroid should participate in the specific subsample with lower percentages.

To achieve this, we draw a WRS from each cluster, where the weights are the FMF values associated with each cluster. Each observation in the training dataset is linked to K membership values, which act as weights in the K subsamples taken from each cluster. The sampling process involves drawing a sample from the entire dataset, where the participation percentage from each cluster increases with the membership value. As a result, we obtain K subsamples from K clusters, with most of the sample participation coming from points in the core of each cluster.

At this stage, we use a special type of WRS that speeds up the weighted sampling process by implementing one-pass random sampling. Its running time is almost the same as that of SRS, especially in large samples [12].

The representative sample size fraction f to be drawn from the training set varies across clusters. This variation depends on the proportion of the cluster size relative to the total data size and the level of variation within the cluster itself. This fraction can be calculated using the following formula:

$$f_j = \sigma(\mu_j) \cdot \frac{N_{C_j}}{N} \tag{2}$$

where $\sigma(\mu_j)$ represents the standard deviation of the jth fuzzy membership function, N_{C_j} is the size of cluster j, and N is the total size of the dataset.

Ensuring Representativity:
To ensure that the sample is representative of the overall dataset, two factors are considered:

- **Cluster Size Proportionality:** Larger clusters (N_{C_j}) contribute a higher proportion to the sample, preventing under-sampling of denser regions of the data. This helps maintain the representation of all parts of the dataset.
- **Variability Proportionality:** The standard deviation $\sigma(\mu_j)$ of the fuzzy membership function is used to assess intra-cluster variability. Clusters with higher variability are given more weight in the sample, ensuring that regions with greater diversity are adequately represented, preserving intra-cluster heterogeneity in the final sample.

3.4. Parallel Classification and Prediction

In this phase, the base classifiers are trained on different subsamples in parallel, either using multiple processors or cores within a single system, or across multiple nodes in a distributed computing cluster.

The final prediction is determined by combining the outputs of the base classifiers. There are two output forms for classifier predictions in classification problems:

- **Label type:** Represents the predicted value of the dependent variable Y. In binary classification, it takes values either 0 or 1, while in multi-class classification, it represents one of several possible categories.
- **Probability type:** Represents the confidence level in the result, with values ranging within [0,1]. Values closer to 0 or 1 indicate higher certainty in the prediction.

Ensemble classifiers commonly employ aggregation rules to combine the outputs of base classifiers [9]. In this study, we introduce a *Weighted Average Probabilities* rule, a soft aggregation method that combines the predicted probabilities with the FMF values of the tested observations.

The final aggregated probability for each class l is computed as follows:

$$p_{\text{final},l} = \frac{\sum_{j=1}^{k} w_j \cdot p_{l,j}}{\sum_{j=1}^{k} w_j} \tag{3}$$

where:

- C is the number of classes,
- $p_{l,j}$ is the probability that the base classifier j predicts the tested observation belongs to class l,
- $p_{\text{final},l}$ is the final aggregated probability of class l,
- w_j is the fuzzy membership value of the tested observation for cluster j.

For each tested observation, there are k associated fuzzy membership values w_j, calculated based on the distance between the tested instance and the centroid of each of the k clusters.

3.5. Case Study: Gas Sensor Array Temperature Modulation Dataset

The case study section presents an evaluation of the proposed FCPC methodology, applied to the Gas Modulation dataset obtained from the University of California, Irvine (UCI) Machine Learning Repository [43]. It compares the performance of FCPC against traditional classifiers to highlight the advantages of the proposed approach in handling uncertainties in data classification.

3.5.1. Dataset Description. The Gas Modulation dataset, containing 4,095,000 samples and 18 predictive features, originates from experiments conducted using a chemical detection platform composed of 14 temperature-modulated metal oxide semiconductor (MOX) gas sensors. The sensors were exposed to dynamic mixtures of carbon monoxide (CO) and humid synthetic air in a controlled gas chamber. The dataset's predictive variable, discretized from continuous measurements, introduce complexity that makes it particularly suitable for evaluating clustering and classification methodologies like FCPC.

This dataset offers flexibility in adjusting both:

- Number of Clusters: Allows assessment of how varying cluster granularity influences classification performance.
- Number of Target Categories (”y”): Supports evaluation of FCPC’s adaptability to different levels of target complexity.

The Gas Modulation dataset aligns well with the objectives of FCPC:

- Target Category Variability: It permits controlled variation of target categories to analyze FCPC’s response to increasing classification complexity.
- Cluster Number Effects: The dataset structure enables testing how the number of clusters impacts accuracy and related metrics.
- Heterogeneity and Uncertainty: Discretized variables introduce data uncertainty, which FCPC’s fuzzification and weighted sampling techniques are designed to manage effectively.

In addition to its methodological relevance, the dataset has significant real-world applications in gas sensing, environmental monitoring, and industrial processes. Accurate gas classification is critical for safety, efficiency, and scientific progress. In summary, the Gas Modulation dataset’s combination of discretized predictive variables, adjustable clustering granularity, and target category flexibility makes it an ideal case study for validating FCPC’s robustness, scalability, and adaptability to real-world challenges.

3.5.2. FCPC Implementation The FCPC methodology for the Gas Modulation dataset was implemented following these steps:

Step 1: Clustering the Dataset

- Input: The training dataset (70% of the total samples).
- Output: Cluster centroids and assignments.
- Tool: Clara (Clustering Large Applications).
- Details: The elbow curve method (Figure3) was applied to determine the optimal number of clusters, resulting in $k = 2$. This step partitions the data into manageable groups while maintaining inherent data structure.

Step 2: Fuzzification of Membership Functions

- Input: Clustering results.
- Output: Fuzzy membership values for each observation, representing their degree of belonging to different clusters.
- Tool: Trapezoid fuzzy membership function 1.
- Details: Fuzzification introduces flexibility in assigning data points to clusters, enabling overlap and better representation of real-world uncertainty.

Step 3: WRS

- Input: Fuzzy membership values and clusters.
- Output: Reduced subsamples from each cluster.
- Tool: WRS.
- Details: Sampling ensures that the data used for training classifiers is representative while reducing redundancy.

Step 4: Parallel Classifier Training

- Input: Subsamples generated in Step 3.
- Output: Trained classifiers for each cluster.
- Tool: Parallel processing frameworks implemented using the ‘caret’ R package.
- Details: Each subsample is used to train a base classifier through 5-fold cross-validation with parameter tuning. Parallelization reduces computational time significantly.

Step 5: Prediction Aggregation

- Input: Predictions from individual classifiers.
- Output: Final predicted labels for test data.
- Tool: Weighted aggregation strategy (Equation 3).

3.5.3. Evaluation Metrics. The six most used classification evaluation metrics in the literature are Accuracy, Recall (Sensitivity), Specificity, Precision, the F1-measure, and Area Under Roc Curve (AUC) as stated by [36].

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{6}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

$$\text{Geometric Mean (GM)} = \sqrt{\text{Recall} \times \text{Specificity}} \tag{9}$$

[5]

Where TP (True Positive) is the number of correctly predicted positives, TN (True Negative) is the number of correctly predicted negatives, FP (False Positive) is the number of negative samples predicted as positive, and FN (False Negative) is the number of positive samples predicted as negative.

The ROC curve plots sensitivity vs specificity at different classification thresholds. AUC is calculated as the two-dimensional area underneath the entire ROC curve.

In this subsection we will use the comprehensive performance metrics (Accuracy, AUC, and F1 score) for ease of presentation while other metrics will be used further in the comprehensive comparison (Section 4)

Table 1. Evaluation Metrics of Gas Modulation Dataset for FCPC and Ordinary Classifiers.

Classifier	**Method**	**Accuracy**	**AUC**	**F1 Score**
LR	FCPC	0.830138	0.9278	0.8305411
	Ordinary	0.8182928	0.9169	0.8155213
NB	FCPC	0.798709	0.8827	0.7751365
	Ordinary	0.7819763	0.8101	0.7531038
DT	FCPC	0.887511	0.9243	0.8870755
	Ordinary	0.8782321	0.8970	0.8786934
RF	FCPC	0.9685086	0.9932	0.9682059
	Ordinary	0.9675597	0.9920	0.9671091

3.5.4. Comparison Between FCPC and Ordinary Classifiers The predicted variable Y was categorized into two levels: Low ($Y < 10$) and High ($Y \geq 10$). Table 1 compares the evaluation metrics (Accuracy, AUC, and F1 score) of the FCPC method and the ordinary classifiers (LR, NB, DT, and RF) on the Gas Modulation dataset using the same partitioned training data.

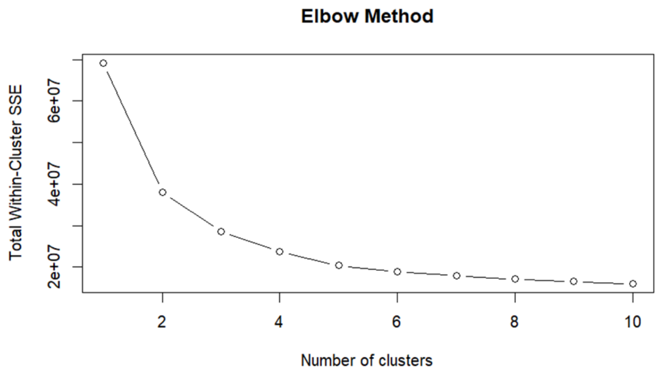

Figure 3. Elbow Technique curve for Gas Data.

Summary of the results:

- LR: The improvements are relatively small but still significant, highlighting FCPC's ability to enhance LR's performance by reducing uncertainty in classification.
- NB: These improvements are more noticeable compared to LR, suggesting that FCPC has a more substantial impact on classifiers that typically handle uncertainty less effectively, like NB.
- DT: The improvements in this case are quite noticeable, suggesting that FCPC helps in refining decision-making within DTs by enhancing the handling of data uncertainty.
- RF: While the improvements are minor in comparison to other classifiers, FCPC still contributes to a better overall performance, reinforcing its potential in optimizing ensemble methods like RF.

Overall, FCPC consistently improves the performance of all classifiers, with the most notable improvements seen with NB and DT. The results suggest that FCPC is particularly effective in enhancing classifiers that deal with uncertainty, like NB, by refining classification performance across various metrics. Even for more complex models like RF, FCPC shows a slight but consistent edge, which points to its potential for enhancing classifier robustness and accuracy.

3.5.5. Effect of Number of Clusters K on FCPC. Initially, we relied on the elbow method to determine the optimal number of clusters K. In this section, we examine the impact of varying K on the evaluation metrics of different classifiers. To provide insight into the cluster formation at different values of K, visualizations of the clusters using the first two principal components were generated for $K = 2$, $K = 3$, and $K = 4$ as shown in Figures 4 –6. These visualizations demonstrate how changes in the number of clusters influence their structure and distribution. Table 2 summarizes the evaluation metrics of FCPC classifiers with varying K values, offering a quantitative comparison across these configurations.

The results in Table 2, alongside the visual insights from Figures 4–6, demonstrate the effect of varying the number of clusters K on FCPC classifier performance. For LR, the best performance occurs at $K = 3$. This aligns with Figure 5, where three clusters provide a balanced representation of the dataset without excessive complexity. In contrast, at $K = 2$ Figure 4, the cluster boundaries are too coarse, which may oversimplify the data. At $K = 4$ (Figure 6), finer partitions introduce overlaps that hinder LR's generalization, slightly reducing its performance. For NB, performance remains relatively stable across K, with a slight advantage observed at $K = 3$. This behavior is expected, as NB's probabilistic nature makes it less sensitive to cluster granularity. Nevertheless, Figure 5 confirms that the cluster separation at $K = 3$ contributes to its marginal improvement compared to other K values. DT shows

Table 2. Evaluation metrics for FCPC classifiers with varying K values.

Classifier	K	Accuracy	AUC	F1 Score
LR	K=2	0.830138	0.9278	0.8305411
	K=3	0.8672577	0.938	0.8662775
	K=4	0.8406936	0.9351	0.8328267
NB	K=2	0.798709	0.8827	0.7751365
	K=3	0.8002919	0.8896	0.7783269
	K=4	0.7941277	0.8613	0.7728143
DT	K=2	0.887511	0.9243	0.8870755
	K=3	0.8863722	0.9285	0.8864542
	K=4	0.8895596	0.9309	0.8909847
RF	K=2	0.9685086	0.9932	0.9682059
	K=3	0.9677878	0.9929	0.9677845
	K=4	0.9632846	0.9881	0.9626948

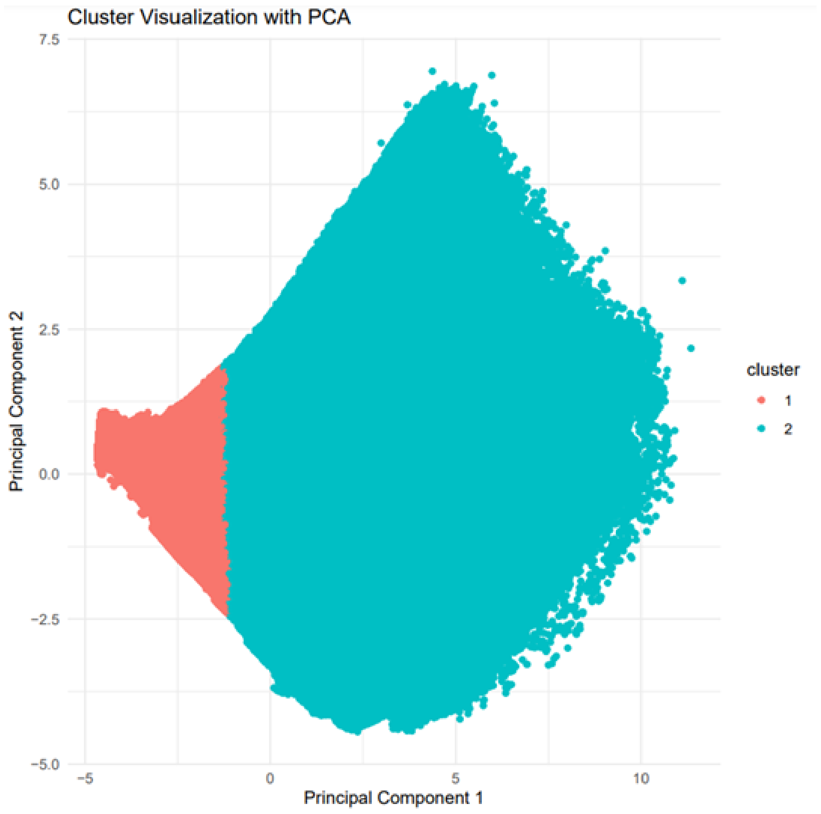

Figure 4. Cluster visualization for Gas Data $K = 2$.

consistent performance across all values of K, with a slight edge at $K = 4$. The increased granularity at $K = 4$, as observed in Figure 6, enables DT to leverage finer cluster boundaries effectively. Unlike LR, DT adapts well to the overlaps introduced by K=4, maintaining high performance without significant trade-offs. RF stands out for its robustness, achieving the highest performance across all K values. RF remains largely unaffected by changes in K. This is visually supported by Figures 4–6, where RF effectively adapts to both coarse and fine cluster boundaries, demonstrating its ability to generalize even as K increases.

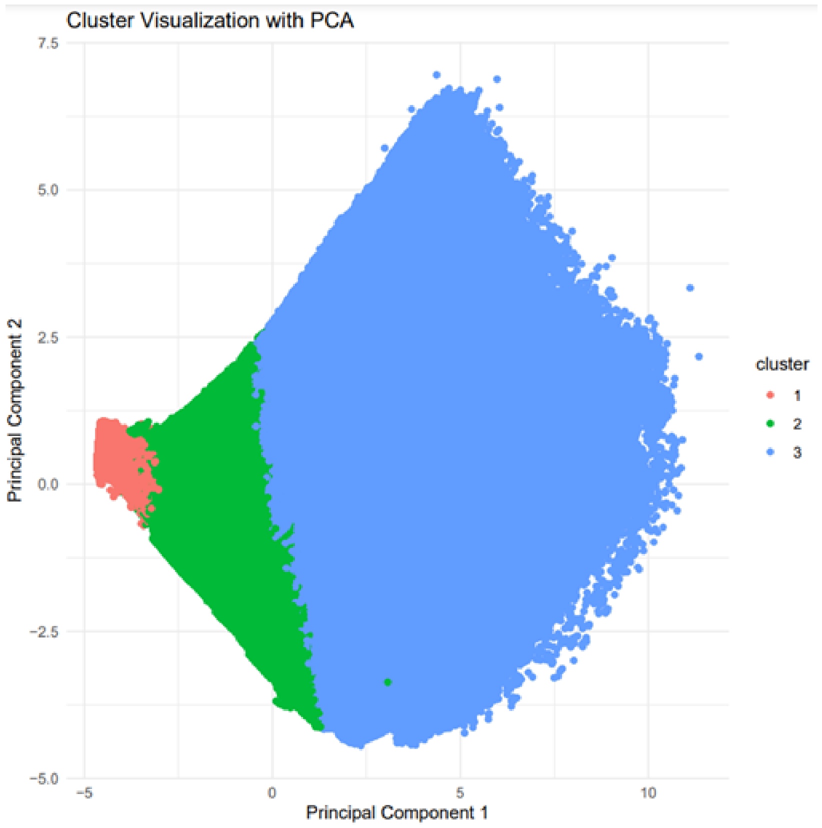

Figure 5. Cluster visualization for Gas Data $K = 3$.

Overall, $K = 3$ emerges as a generally effective choice, particularly for simpler classifiers like LR and NB, where it provides balanced cluster separation and improved generalization. However, DT and RF exhibit robustness across all K values, with RF maintaining peak performance regardless of the cluster configuration. These findings emphasize the importance of tuning K for simpler models while leveraging ensemble methods like RF for their adaptability and resilience to varying cluster granularity. For computationally intensive ensemble methods like RF, the choice of K becomes less critical, as these models effectively handle both coarse and fine-grained clusters.

3.5.6. Effect of Number of Y Categories on FCPC. This subsection examines the impact of discretizing Y into four categories—Low ($Y < 5$), Moderate ($5 \leq Y < 10$), High ($10 \leq Y < 15$), and Very High ($Y \geq 15$)—on the performance of the FCPC classifiers compared to their binary counterparts. To evaluate this, the average Accuracy, AUC, and F1 score are calculated for each pair of classes. Table 3 summarizes these metrics for the categorical Y. Furthermore, Figure 7 provides a visual comparison of the results from both Table 1 and 3, highlighting the differences in performance between FCPC and the ordinary classifiers under binary and categorical scenarios.

The results in Table 3 indicate a clear performance gap between FCPC and ordinary classifiers when Y is categorized into four levels. FCPC consistently outperforms the ordinary classifiers across all metrics (Accuracy, AUC, and F1 score), demonstrating its ability to better handle increased class complexity.

- For LR, FCPC achieves higher Accuracy, AUC, and F1 score compared to the ordinary LR. The improvement indicates that FCPC provides a more refined clustering approach, which enhances LR's ability to model the relationships among the four categories.

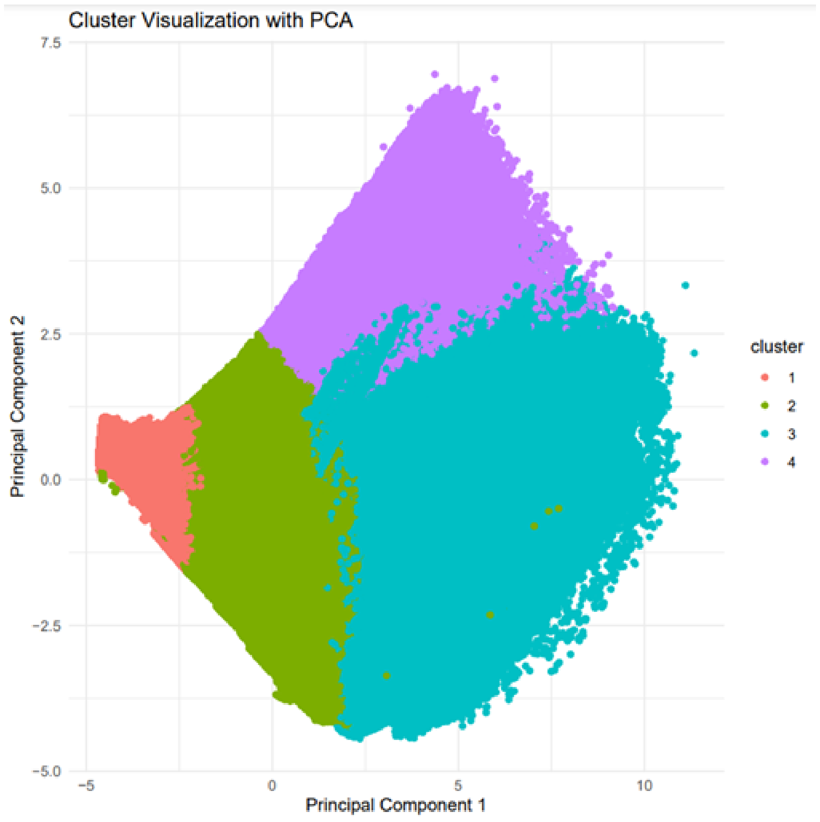

Figure 6. Cluster visualization for Gas Data $K = 4$.

Table 3. Evaluation Metrics of Gas Modulation Dataset for FCPC and Ordinary Classifiers 4 categories Y.

Classifier	**Method**	**Accuracy**	**AUC**	**F1 score**
LR	FCPC	0.6976707	0.8881	0.6751268
	Ordinary	0.6595172	0.8691	0.6293762
NB	FCPC	0.6147936	0.8122	0.5949038
	Ordinary	0.5679363	0.7506	0.5574053
DT	FCPC	0.7455427	0.8802	0.7216278
	Ordinary	0.7215061	0.8589	0.6913156
RF	FCPC	0.9345252	0.9894	0.9302664
	Ordinary	0.9300966	0.9879	0.9255078

- NB shows a similar trend, with FCPC outperforming the ordinary NB across all metrics. These improvements suggest that FCPC's clustering effectively reduces class overlap, which benefits NB's probabilistic classification.
- For DT, FCPC achieves noticeable gains. The ability of FCPC to create more meaningful clusters likely enables DT to make better splits, leading to improved performance.
- RF continues to exhibit the highest performance overall, reflecting its robustness. While both FCPC and the ordinary RF perform well, FCPC achieves slightly higher Accuracy compared to the ordinary RF.
  When comparing the results in Table 3 (categorical Y) to Table 1 (binary Y), it is evident that performance decreases for all classifiers as the complexity of Y increases. For example, LR's Accuracy and NB's Accuracy

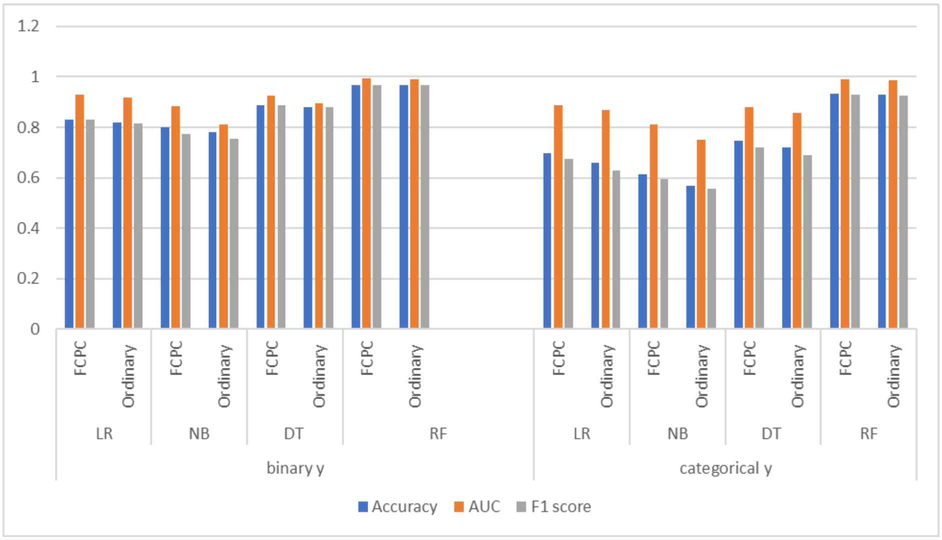

Figure 7. Accuracy, AUC, and F1 measure for FCPC vs Ordinary classifiers for Binary Y and for Categorical Y

decrease from 0.7987 to 0.6148. This trend is expected, as increasing the number of categories introduces additional class boundaries, making the classification task more challenging.

However, the consistent advantage of FCPC over the ordinary classifiers in both binary and categorical scenarios highlights its effectiveness. FCPC's clustering process appears to mitigate the difficulties associated with multi-class classification, as it organizes the data into more meaningful clusters that enhance the classifiers' ability to distinguish between categories. Overall, Figure 7 effectively summarizes the performance differences across both binary and categorical scenarios, illustrating how FCPC maintains a measurable advantage over ordinary classifiers regardless of class complexity. The results confirm FCPC's robustness and adaptability in handling datasets with varying numbers of target classes.

3.6. *Theoretical Properties of FCPC*

To mathematically validate the soundness of our methodology, this section provides derivations across two interconnected components: fuzzification effects and generalization error analysis of ensemble. These components form the foundation of the proposed methodology, ensuring representativity, reduced variance, and minimized generalization error.

3.6.1. Component 1: Fuzzification Effects. In weighted sampling, the probability of selecting a data point x_i is proportional to its assigned weight w_i. The probability is given by:

$$P(x_i \in S) = \frac{w_i}{\sum_{l=1}^{N} w_l} \tag{10}$$

[12]

Where:

- S is the sampled subset.
- w_i is the weight of data point x_i.
- N is the size of the population with N weighted items.

For fuzzified clusters, the membership degree $\mu_j(i) \in [0,1]$ of a point x_i in cluster j determines its probability of inclusion:

$$P(x_i \in S_j) = \frac{\mu_j(i)}{\sum_{l=1}^{N_{Cj}} \mu_j(l)} \tag{11}$$

Where:

- $\mu_j(i)$ is the fuzzy membership value of the point x_i in cluster j.
- S_j is the sampled subset for cluster j.
- N_{C_j} is the size of cluster j.

This ensures that points with higher membership values are more likely to be included in the subset, while still accounting for the full distribution of data within the cluster.

Representativity of Membership Degrees: To ensure that the sampled subsets represent the full data distribution, the following considerations are made:

1. Variance of Membership Values: The standard deviation $\sigma(\mu_j)$ of the membership values within cluster j measures the spread of fuzziness within the cluster. A higher $\sigma(\mu_j)$ indicates a greater diversity in the membership values, suggesting a more heterogeneous cluster. In this context, a higher variability leads to a larger sample size fraction f_j for that cluster to preserve this diversity during the subsampling process.
2. Balanced Contribution: Sampling is conducted such that the contribution of both central (high membership value) and peripheral (low membership value) points within each cluster is proportional to their membership values $\mu_j(i)$. This ensures that the diversity within the cluster is preserved. The peripheral points with lower membership values are still included in the sample, but their contribution is weighted less compared to the central points with higher membership values. By doing this, the sampling process reflects the underlying structure of the cluster while maintaining the diversity of data points, regardless of their proximity to the cluster's center.
3. Sample Size Fraction f_j: The sample size fraction f_j is designed to take into account both the cluster size N_{C_j} and the variability of the membership values $\sigma(\mu_j)$, ensuring that clusters with higher variability (greater fuzziness) have a larger contribution to the sampled subset, and that the sampled data remains representative of the entire data distribution.

3.6.2. Component 2: Generalization Error Analysis. The generalization error E_g of the ensemble model can be expressed as the sum of three key components:

$$E_g = E_{\text{bias}}^2 + E_{\text{variance}} + E_{\text{irreducible}}, \tag{12}$$

[19]
where:

- E_{bias}^2 represents the error due to incorrect assumptions in the model.
- E_{variance} accounts for the variability in model predictions due to fluctuations in the training data.
- $E_{\text{irreducible}}$ is the noise inherent in the data that cannot be reduced by any model.

Bias and Variance Trade-offs

1. Bias Reduction: The challenge is to minimize bias without excessively increasing variance. In the FCPC approach, the proportional sampling strategy ensures that all clusters, including smaller or less represented ones, are adequately sampled. This reduces the risk of underrepresentation bias, ensuring the ensemble models do not develop strong biases toward any particular cluster. By better representing the entire data distribution, the ensemble model can more accurately approximate the true data distribution, leading to lower bias in its predictions.
2. Variance Control: The weighted sampling strategy, where points closer to the centroid of each cluster are given higher membership values, helps control the variance. This approach reduces the variability in training sets by ensuring that classifiers are trained on sufficiently diverse data subsets. As a result, the variance between different classifiers is minimized, leading to a more stable and robust ensemble model that generalizes better to new data.

Combined Error Bound The total generalization error of an ensemble can be bounded as follows:

$$E_g \leq E_{\text{bias}} + \frac{\sigma^2}{k}, \tag{13}$$

where the variance term $\frac{\sigma^2}{k}$ represents the variance, which decreases as the number of classifiers increases. This suggests that increasing the size of the ensemble helps reduce variance and, in turn, the generalization error. By balancing bias reduction and variance control, the ensemble algorithm converges to a stable state, ensuring effective classification performance on unseen data.

These two components together ensure that the algorithm converges to a stable state where the model can be used for classification effectively.

3.7. Complexity Analysis of the FCPC Algorithm

The complexity analysis of our algorithm can be broken down into key steps:

CLARA Clustering CLARA is used for clustering the dataset into k clusters. The complexity of CLARA is influenced by its repeated sampling process, where it draws multiple random samples from the data and applies the clustering process to each sample. CLARA algorithm clusters a number of random samples T, each of size n, into k clusters. The time complexity is:

$$O(T \cdot n \cdot k) \tag{14}$$

[40]

Fuzzy Membership Function Calculation The fuzzy membership values for all data points are computed with respect to each of the k clusters. This computation requires:

$$O(k \cdot N) \tag{15}$$

One-Pass WRS One-pass WRS, as described by [12], allows for an efficient selection of samples based on precomputed weights. The key here is that the sampling process is done in a single pass over the data, which means that, like SRS, the complexity per cluster is $O(N_C)$, where N_C is the number of points in the cluster. The total time complexity for sampling from k clusters is therefore:

$$O(k \cdot f_j \cdot N_C \cdot \log\left(\frac{N_C}{f_j \cdot N_C}\right)) \tag{16}$$

This ensures that the time complexity remains close to that of SRS, even when weights (e.g., FMF values) are applied.

Ensemble Complexity with Base Classifiers (Parallel) The training of base classifiers is performed in parallel, where each base classifier is trained on a subset of the data. For k classifiers, each trained on f_j samples with p variables (features) and requiring I iterations for convergence, the time complexity per classifier is:

$$O(f_j \cdot N_C \cdot P \cdot I) \tag{17}$$

[9] where:

- f_j is the number of samples used for training each base classifier.
- P is the number of trained variables (features).
- I is the number of iterations for convergence.

Since the classifiers are trained in parallel, the total time complexity for this stage becomes the maximum training time of any single classifier.

Total Complexity of FCPC Algorithm Now combining all the components, the overall time complexity would be:

$$O(T \cdot n \cdot k) + O(k \cdot N) + O(k \cdot f_j \cdot N_C \cdot \log\left(\frac{1}{f_j}\right)) + O(f_j \cdot N_C \cdot P \cdot I) \tag{18}$$

Given that the complexity terms are generally linear with respect to N (the number of data points), and there are no exponential terms (which would result in infeasibility for very large datasets), the algorithm should remain scalable even as the dataset grows. By managing these complexities, the algorithm can efficiently handle large datasets, providing good scalability without the typical bottlenecks seen in more computationally intensive methods.

4. Experimentation

To evaluate the effectiveness of the proposed methodology, we used three benchmark datasets from the University of California, Irvine (UCI) Machine Learning Repository [43]. The datasets used in our experiments are listed in Table 4. We compare the performance of the proposed algorithm using four classification techniques: LR, NB, DT, RF.

The datasets chosen for the experiment span various domains and contain both large sample sizes and high-dimensional attributes. Table 4 provides descriptions of the datasets, along with the percentage of data sampled from each cluster as calculated using Equation 2.

Table 4. Description of the datasets used in the experiments

Dataset	Data Size	Number of Attributes	Attribute Characteristics	Number of Clusters	Percentage of Data Sampled
SUSY	5,000,000	18	Real	3	(10%, 7%, 9%)
HEPMASS	10,500,000	28	Real	2	(8%, 13%)
HIGGS	11,000,000	28	Real	2	(12%, 11%)

4.1. Experimental Setups

In our experiments, all outputs are generated through parallel processing across multiple cores. Each dataset is split into a 70% training set and a 30% test set. For consistency, the same parameters are fixed, and the model is evaluated using the same test partition across different algorithms.

To compare the efficiency of FCPC in terms of both complexity and predictive power, we adopt two schemes:

1. Baseline Performance: The ordinary classifier is applied assuming the entire training set fits in memory.
2. Effect of Fuzzification: We compare FCPC with its hard version, the Cluster-based Distributed Parallel Classifier Ensemble (CPC). In this comparison, a SRS is applied to each cluster to reduce its size, and a Cluster-based Classifier Ensemble is trained on the resulting subsets in parallel. The fuzzy WRS in FCPC is replaced by SRS, ensuring the same sample size in both cases, as calculated using Equation 2. For prediction, a special case of the weighted average (with equal weights) is used in CPC.

4.1.1. Hardware Setup. All experiments were conducted on a machine with the following specifications:

- 6 physical Intel cores, 12 logical processors (operating at 2.6 GHz)
- 16 GB of RAM Parallel processing was implemented across the 6 cores, enabling simultaneous training of multiple base classifiers.

4.1.2. Software Versions. The computations were performed using the R programming language, with task parallelization managed by the `foreach` package and the `doParallel` backend to distribute tasks across available cores. The following R packages were used for clustering and classifier training:

1. Clustering:The `clara` function from the `cluster` package was used to implement CLARA clustering.

2. WRS: The sample_int_crank function from the `wrswoR` package was used to perform weighted sampling with one-pass random sampling.
3. LR The `multinom` function from the `nnet` package was used to train LR, with default parameters.
4. NB: The naive_bayes function from the `naivebayes` package was used for NB training.
5. DT: The `rpart` function from the `rpart` package was used to train DT.
6. RF: The `randomForest` function from the `randomForest` package was used to train RF. The number of trees was set to 300 for the SUSY and HEPMASS datasets, and 200 for the HIGGS dataset, with all other parameters set to the default values.

4.1.3. Evaluation. The performance of the classification approaches was evaluated based on both evaluation metrics (Accuracy, Recall, , Specificity, AUC, Precision, F1, and GM) and estimation runtime.

A better classifier is characterized by lower runtime and higher evaluation metrics. The runtime refers to the total time taken for the classifier to complete the estimation process, starting from clustering the entire training set to generating predictions.

4.2. Experimental Results

Tables 5 –7 present the results of the running time and all evaluation metrics of different classification algorithms on the three datasets using CPC, FCPC, and ordinary classifiers. The results demonstrate the superior performance of FCPC across most metrics and datasets, alongside significant reductions in computational time. Both CPC and FCPC consistently reduce the running time of classification models by 20% to 90%, depending on the classifier, compared to ordinary methods. After approximation, the running times of CPC and FCPC converge, allowing their values to be reported as a single result. Susy Dataset In the Sussy dataset evaluation, FCPC outperforms both CPC and ordinary methods across all classifiers, showcasing the benefits of fuzzified clustering combined with parallel processing. It achieves higher Accuracy, AUC, F1 score, and GM for FCPC compared to both CPC and ordinary methods particularly NB and RF classifiers. FCPC is also computationally efficient, with RF running in 4.7 minutes compared to 43.1 minutes for the ordinary method. Hepmass Dataset For the Hepmass dataset, FCPC consistently outperforms CPC and ordinary methods across all classifiers. NB achieves high performance, with FCPC delivering superior accuracy, AUC, F1 score, and GM compared to the other methods, demonstrating its ability to handle overlapping clusters effectively. DT also shows competitive results, particularly in AUC, reinforcing the robustness of FCPC across classifiers. RF benefits significantly from FCPC, achieving higher AUC, F1 score, and GM compared to both CPC and ordinary methods, further emphasizing FCPC's strength in delivering balanced performance. Computationally, FCPC is faster than ordinary methods, with RF running in 8.2 minutes compared to 23.3 minutes for the ordinary method.

Higgs Dataset In the Higgs dataset, FCPC demonstrates its robustness by outperforming CPC and ordinary methods across most classifiers. It achieves substantial gains in accuracy, AUC, F1 score, and GM, particularly in RF, underscoring its superior balanced performance. Computationally, FCPC remains efficient, with RF running in 13.1 minutes compared to 81.6 minutes for the ordinary method.

4.3. Key Findings and Conclusion:

The results reveal that FCPC consistently outperformed both CPC and the ordinary classifier across most metrics, despite a significant reduction in training data size. Metrics that combine TP and TN, such as Accuracy, F1 score, and AUC, are plotted in Figures 8 –10. Specifically, the RF and NB classifiers demonstrated notable enhancements with FCPC, while DT benefited as well, particularly in achieving higher AUC values. LR showed only marginal improvements, especially in the F1 score.

In contrast, CPC exhibited lower performance than the ordinary classifier in some cases, such as NB on the Sussy and Higgs datasets, while DT and RF demonstrated slight improvements or maintained similar performance.

In conclusion, the FCPC methodology effectively supports the modeling of Big Data classifiers by significantly reducing the sample size without compromising the classification model's efficiency, delivering improved or competitive performance across diverse datasets and classifiers.

Table 5. Evaluation metrics and running time of the SUSY dataset.

Classifier	Method	Accuracy	Recall	Specificity	AUC	Precision	F1 Score	GM	Time (min)
LR	CPC	0.788	0.662	0.893	0.8567	0.840	0.741	0.769	1.2
	FCPC	0.792	0.696	0.872	0.8632	0.821	0.754	0.779	
	Ordinary	0.788	0.673	0.885	0.8586	0.832	0.744	0.772	2.6
NB	CPC	0.716	0.677	0.748	0.8152	0.693	0.685	0.712	1.1
	FCPC	0.751	0.668	0.820	0.8275	0.757	0.710	0.740	
	Ordinary	0.735	0.552	0.889	0.8108	0.807	0.656	0.701	1.5
DT	CPC	0.755	0.688	0.811	0.8009	0.754	0.719	0.747	1.4
	FCPC	0.765	0.697	0.822	0.8064	0.767	0.731	0.757	
	Ordinary	0.763	0.704	0.814	0.7943	0.761	0.731	0.757	15.1
RF	CPC	0.784	0.758	0.807	0.8914	0.768	0.763	0.782	4.7
	FCPC	0.810	0.726	0.881	0.9012	0.839	0.779	0.800	
	Ordinary	0.801	0.722	0.868	0.8700	0.823	0.769	0.792	43.1

Table 6. Evaluation metrics and running time of the Hepmass dataset.

Classifier	Method	Accuracy	Recall	Specificity	AUC	Precision	F1 Score	GM	Time (min)
LR	CPC	0.835	0.810	0.861	0.9170	0.854	0.831	0.835	1.8
	FCPC	0.840	0.843	0.836	0.9205	0.837	0.840	0.839	
	Ordinary	0.837	0.836	0.838	0.9176	0.837	0.837	0.837	3.9
NB	CPC	0.792	0.731	0.853	0.8700	0.833	0.779	0.790	1.5
	FCPC	0.807	0.778	0.836	0.8821	0.826	0.802	0.806	
	Ordinary	0.799	0.734	0.865	0.8747	0.844	0.785	0.797	1.9
DT	CPC	0.820	0.809	0.832	0.8200	0.828	0.819	0.820	3.9
	FCPC	0.822	0.871	0.773	0.8891	0.794	0.831	0.821	
	Ordinary	0.820	0.868	0.772	0.8702	0.792	0.828	0.819	24.9
RF	CPC	0.835	0.792	0.878	0.9400	0.867	0.828	0.834	8.2
	FCPC	0.854	0.882	0.824	0.9478	0.840	0.860	0.853	
	Ordinary	0.827	0.776	0.877	0.9456	0.862	0.817	0.825	23.3

Table 7. Evaluation metrics and running time of the Higgs dataset.

Classifier	Method	Accuracy	Recall	Specificity	AUC	Precision	F1 Score	GM	Time (min)
LR	CPC	0.635	0.690	0.573	0.6855	0.645	0.667	0.629	2.1
	FCPC	0.647	0.760	0.520	0.6840	0.641	0.695	0.629	
	Ordinary	0.643	0.744	0.530	0.6723	0.641	0.688	0.628	9.5
NB	CPC	0.602	0.823	0.353	0.6140	0.589	0.687	0.539	1.7
	FCPC	0.638	0.778	0.479	0.6459	0.628	0.695	0.610	
	Ordinary	0.603	0.836	0.340	0.6148	0.588	0.691	0.533	2.2
DT	CPC	0.646	0.584	0.717	0.6691	0.700	0.637	0.647	5.3
	FCPC	0.653	0.607	0.704	0.6797	0.698	0.650	0.654	
	Ordinary	0.653	0.614	0.698	0.6712	0.696	0.652	0.655	66.8
RF	CPC	0.720	0.744	0.693	0.7970	0.732	0.738	0.689	13.1
	FCPC	0.767	0.788	0.743	0.8490	0.776	0.782	0.721	
	Ordinary	0.737	0.760	0.712	0.8250	0.748	0.754	0.695	81.6

4.4. Comparative Analysis

This section compares the performance of the proposed FCPC method with several learning techniques experimented in the literature on the same benchmark datasets used in this research.

4.4.1. Comparison with techniques in [3]: [3] utilizes MLlib and Weka frameworks to several classifiers. The comparison is performed using NB, RF, and DT classifiers across the SUSY, HEPMASS, and HIGGS datasets, with the focus on the AUC metric as presented in Table 8.

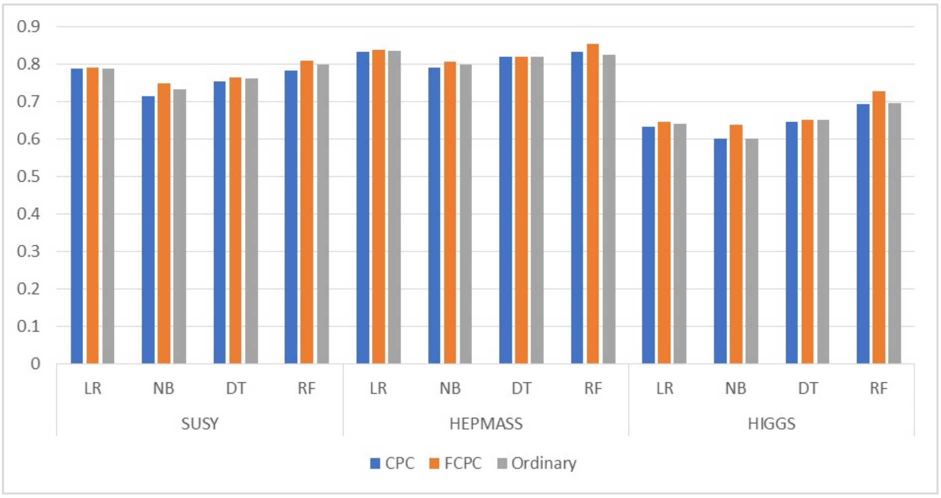

Figure 8. Accuracy Comparison of Techniques (CPC, FCPC, Ordinary) Across Classifiers and Datasets.

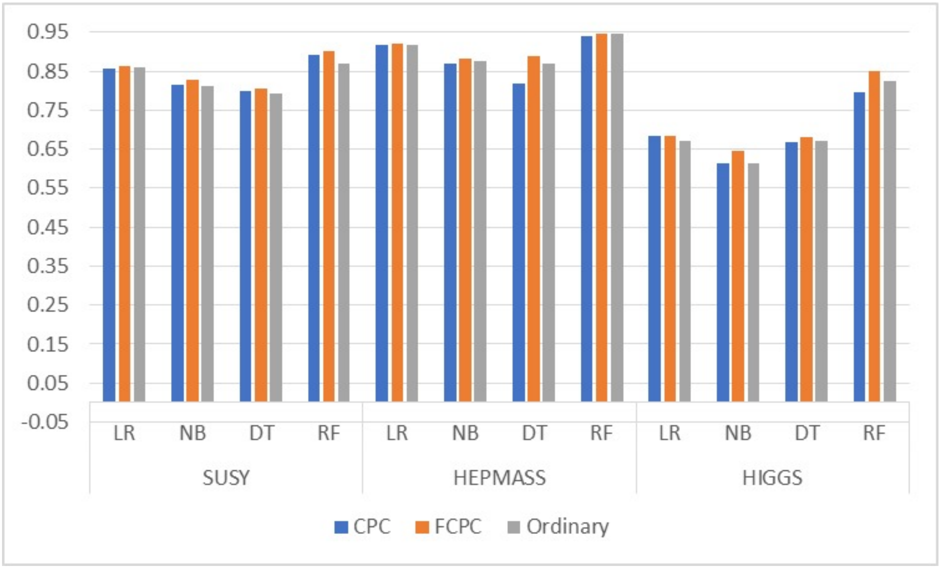

Figure 9. AUC Comparison of Techniques (CPC, FCPC, Ordinary) Across Classifiers and Datasets.

- SUSY Dataset: FCPC outperforms both MLlib and Weka across all methods (NB, DT, RF). The FCPC RF classifier achieves the highest accuracy (0.9012), significantly surpassing MLlib and Weka's performance in all metrics.
- HEPMASS Dataset: FCPC again leads with RF (0.9478), surpassing MLlib and Weka. While MLlib shows strong performance with NB and DT, it still lags behind FCPC RF. Weka performs slightly lower than both FCPC and MLlib.
- HIGGS Dataset: FCPC RF remains the top performer (0.849), again outperforming MLlib and Weka across all methods. MLlib performs better than Weka, but both fall short of FCPC RF, which is clearly the superior classifier in this dataset.

Overall, FCPC demonstrates competitive or superior performance compared to state-of-the-art methods (MLlib and Weka) across multiple datasets and classifiers. Although FCPC does not consistently achieve the highest

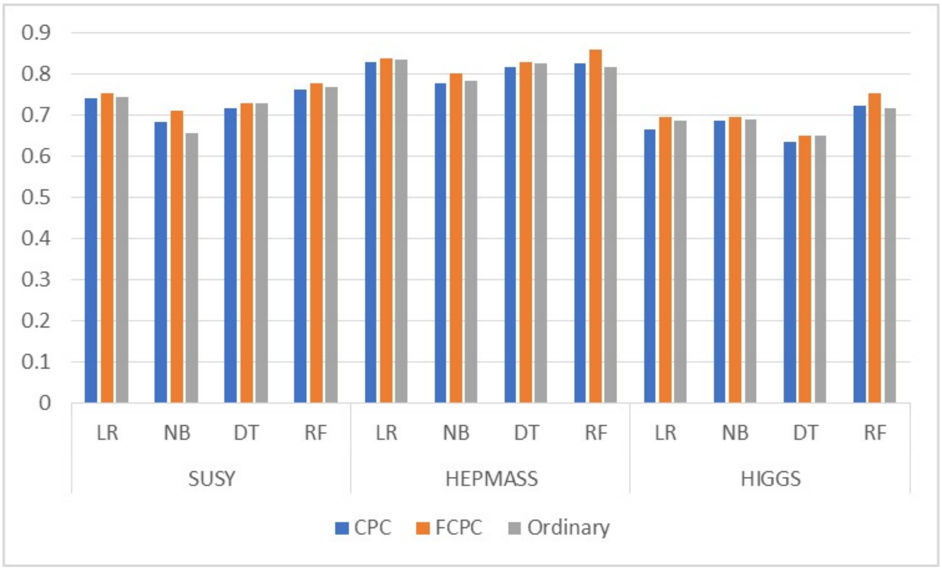

Figure 10. F1 Comparison of Techniques (CPC, FCPC, Ordinary) Across Classifiers and Datasets.

Table 8. Comparison with techniques in [3] for datasets Susy, Hepmass, and Higgs.

Data	**Method**	**NB**	**DT**	**RF**
SUSY	FCPC	0.8275	0.8064	0.9012
	MLlib	0.7853	0.7528	0.7614
	Weka	0.8031	0.7690	0.7731
HEPMASS	FCPC	0.8821	0.8891	0.9478
	MLlib	0.9114	0.8995	0.8948
	Weka	0.9398	0.8971	0.8961
HIGGS	FCPC	0.6459	0.6797	0.8490
	MLlib	0.5741	0.5829	0.5680
	Weka	0.5722	0.5873	0.5712

AUC in every scenario (e.g., NB on HEPMASS), its robust performance across various datasets highlights its effectiveness as a Big Data classification technique.

4.4.2. Comparison with techniques in [4]: [4] implemented Boosted Decision Trees (BDT), Shallow Neural Networks (NN), and Deep Neural Networks (DN). The comparison focuses on the AUC metric across the SUSY and HIGGS datasets. Table 9 presents the AUC results for FCPC classifiers vs classifiers techniques in [4].

- Susy Dataset: FCPC's RF achieves the highest AUC, outperforming BDT, NN, and DN. LR also delivers strong performance with an AUC of 0.8632, comparable to BDT.
- Higgs Dataset: DN achieves the best AUC, while FCPC's RF serves as a competitive alternative, surpassing BDT and NN.

FCPC demonstrates competitive or superior performance compared to BDT, NN, and DN across the SUSY and HIGGS datasets. Its RF approach consistently achieves high AUC, particularly for the SUSY dataset, where it outperforms all other methods. While DN excels on the HIGGS dataset, FCPC's RF offers a robust alternative, especially for large-scale datasets where computational efficiency and scalability are essential. These results highlight FCPC as a reliable and effective method for Big Data classification tasks, with its strengths evident across a range of classifiers and datasets.

Table 9. Comparison with Techniques in [4] for datasets Susy and Higgs.

Data	Method	AUC
SUSY	FCPC-LR	0.8632
	FCPC-NB	0.8275
	FCPC-DT	0.8064
	FCPC-RF	0.9012
	BDT	0.8630
	NN	0.8750
	DN	0.8760
HIGGS	FCPC-LR	0.6840
	FCPC-NB	0.6459
	FCPC-DT	0.6797
	FCPC-RF	0.8490
	BDT	0.8100
	NN	0.8160
	DN	0.8850

4.4.3. Comparison with techniques in [11]: [11] reviewed several methods for Big Data classification, including ChiFRBCSBigData, CHI_BD, DFACFFP, DPAESRCS, DPAESFDTGL, Multiway FDT, and FMDT5, as outlined in the literature review section. These techniques were evaluated on the SUSY and HIGGS datasets, with performance measured primarily in terms of accuracy as presented in Table 10.

Table 10. Comparison with Techniques in [11] for datasets Susy and Higgs.

Data	Method	Accuracy
SUSY	FCPC-LR	0.792
	FCPC-NB	0.751
	FCPC-DT	0.765
	FCPC-RF	0.810
	Chi-FRBCS-BigData	0.558
	CHI_BD	0.654
	DFAC-FFP	0.783
	DPAES-RCS	0.781
	DPAES-FDT-GL	0.786
	Multi-way FDT	0.796
	FMDT5	0.790
HIGGS	FCPC-LR	0.647
	FCPC-NB	0.638
	FCPC-DT	0.653
	FCPC-RF	0.767
	Chi-FRBCS-BigData	0.559
	CHI_BD	0.578
	DFAC-FFP	0.660
	DPAES-RCS	0.650
	DPAES-FDT-GL	0.650
	Multi-way FDT	0.7125
	FMDT5	0.723

- Susy: The FCPC classifiers consistently outperform all techniques from the referenced paper. FCPC RF achieves the highest accuracy of 0.810, exceeding the best paper technique, Multi-way FDT (79.63%). Even FCPC LR achieves 0.792, surpassing all the compared techniques except for Multi-way FDT
- Higgs: FCPC RF leads the HIGGS dataset, significantly outperforming the closest paper technique, `FMDT5`.Other FCPC classifiers, such as LR and DT, also outperform several paper techniques, including ChiFRBCSBigData and CHI_BD.

FCPC classifiers consistently outperform the methods from the paper on both SUSY and HIGGS datasets, with RF achieving the best results. Even lower-ranked FCPC classifiers, such as LR and DT, show competitiveness, often surpassing many of the paper's techniques. RF stands out with superior accuracy, while other FCPC classifiers provide robust alternatives. Overall, the FCPC framework offers significant improvements in classification accuracy, making it a strong contender for Big Data classification tasks.

4.4.4. Comparison with [5]: [5] introduced FDR2BD, as outlined in the literature review section. The comparison evaluates GM classification performance across the Susy, Hepmass, and Higgs datasets presented in Table 11.

Table 11. Comparison with FDR2-BD for datasets Susy, Hepmass, and Higgs.

Data	**Method**	**GM**
SUSY	FCPC-LR	0.779
	FCPC-NB	0.740
	FCPC-DT	0.757
	FCPC-RF	0.800
	FDR2-BD	0.762
HEPMASS	FCPC-LR	0.839
	FCPC-NB	0.806
	FCPC-DT	0.821
	FCPC-RF	0.853
	FDR2-BD	0.825
HIGGS	FCPC-LR	0.629
	FCPC-NB	0.610
	FCPC-DT	0.654
	FCPC-RF	0.721
	FDR2-BD	0.649

- Susy: FCPC RF achieves the highest GM, surpassing FDR2BD. FCPC LR and FCPC DT both outperform FDR2BD as well.
- Hepmass: FCPC RF leads with a GM of 0.853, outperforming FDR2BD. FCPC LR and FCPC DT also exceed FDR2BD.
- Higgs: FCPC RF achieves the highest GM of 0.721, outperforming FDR2BD. FCPC LR and FCPC DT also outperform FDR2BD.

FCPC RF outperforms all other methods across the datasets, achieving the highest GM on each and significantly surpassing FDR2BD. FCPC LR and DT also outperform FDR2BD in all datasets, though not as strongly as FCPC RF. Overall, while FDR2BD performs well, it consistently falls short of the FCPC classifiers, highlighting the strength and robustness of the FCPC approach for Big Data classification tasks.

4.4.5. Comparison with Techniques in [22]: [22] implemented Gradient-boosted Tree (GBT) and RF using PySpark, RapidMiner, and Sklearn on Susy, Hepmass, and Higgs datasets. The comparison focuses on performance metrics, namely Accuracy and AUC listed in Table 12.

Table 12. Comparison with Techniques in [22] for datasets Susy, Hepmass, and Higgs.

Data	**Method**	**Accuracy**	**AUC**
SUSY	FCPC-LR	0.792	0.8632
	FCPC-NB	0.751	0.8275
	FCPC-DT	0.765	0.8064
	FCPC-RF	0.810	0.9012
	GBT PySpark	0.80	0.829
	GBT RapidMiner	0.64	0.819
HEPMASS	FCPC-LR	0.840	0.9205
	FCPC-NB	0.807	0.8821
	FCPC-DT	0.822	0.8891
	FCPC-RF	0.854	0.9478
	RF PySpark	0.890	0.832
	GBT RapidMiner	0.86	0.944
HIGGS	FCPC-LR	0.647	0.684
	FCPC-NB	0.638	0.6459
	FCPC-DT	0.653	0.6797
	FCPC-RF	0.767	0.849
	RF PySpark	0.894	0.700
	GBT RapidMiner	0.624	0.680
	GBT Sklearn	0.714	0.710

- Susy: FCPC RF achieves the highest Accuracy and AUC, surpassing GBT PySpark and GBT RapidMiner. FCPC LR and FCPC DT also outperform GBT RapidMiner and GBT PySpark in both metrics.
- Hepmass: FCPC RF achieves the highest Accuracy and AUC, outperforming GBT RapidMiner and RF PySpark. FCPC LR and FCPC DT perform better than GBT RapidMiner and RF PySpark in terms of AUC.
- Higgs: FCPC RF surpasses RF PySpark. FCPC LR and FCPC DT outperform both GBT RapidMiner and GBT Sklearn.

Across all the datasets, FCPC classifiers, particularly FCPC RF, consistently outperform the GBT and RF methods from [22] in both Accuracy and AUC, with particularly strong performance in HEPMASS and SUSY. FCPC LR and FCPC DT also demonstrate strong performance, particularly on the SUSY and HEPMASS datasets, surpassing many of the GBT implementations in AUC. Overall, FCPC provides a competitive or superior alternative to GBT and RF classifiers.

4.4.6. Summary Figures 11 –13 summarize the results of evaluation metrics of the proposed FCPC in comparison with the previously mentioned state-of-art techniques in the literature for each of the studied benchmark datasets: Susy, Hepmass, and Higgs respectively.

We can conclude that FCPC consistently demonstrates competitive or superior performance compared to state-of-the-art methods across a variety of datasets and classifiers. It excels in RF classification, outperforming methods like MLlib, Weka, and others in terms of AUC, GM values, and accuracy, particularly on challenging datasets like SUSY and HEPMASS. Despite some competition from more complex models like DN on specific datasets, FCPC offers a highly efficient and robust alternative for Big Data classification tasks, particularly for large-scale datasets where computational efficiency is critical.

5. Conclusion and Future Work

In this study, we introduced FCPC as an innovative approach for subsampling Big Data classification. The methodology begins by clustering the dataset to model its heterogeneity, followed by soft cluster-based weighted

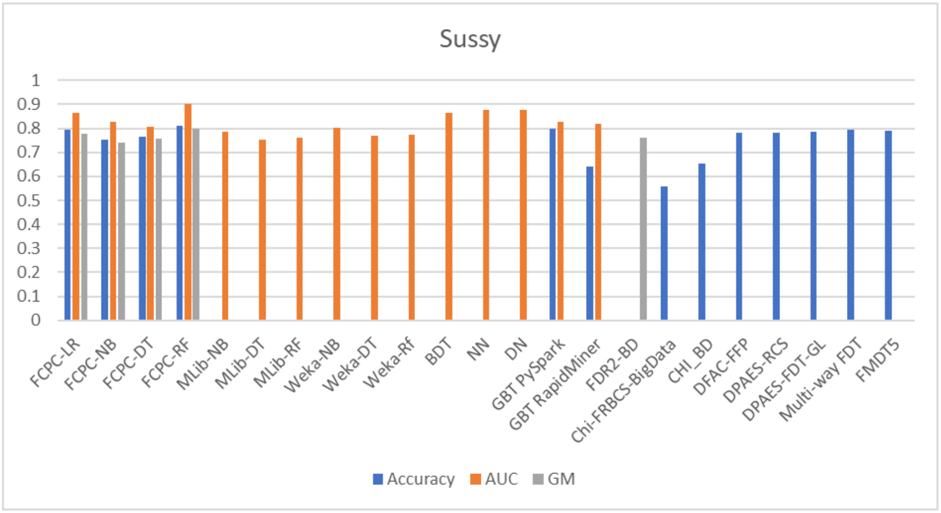

Figure 11. Accuracy, AUC, GM of FCPC vs State-of-art Techniques for Susy dataset.

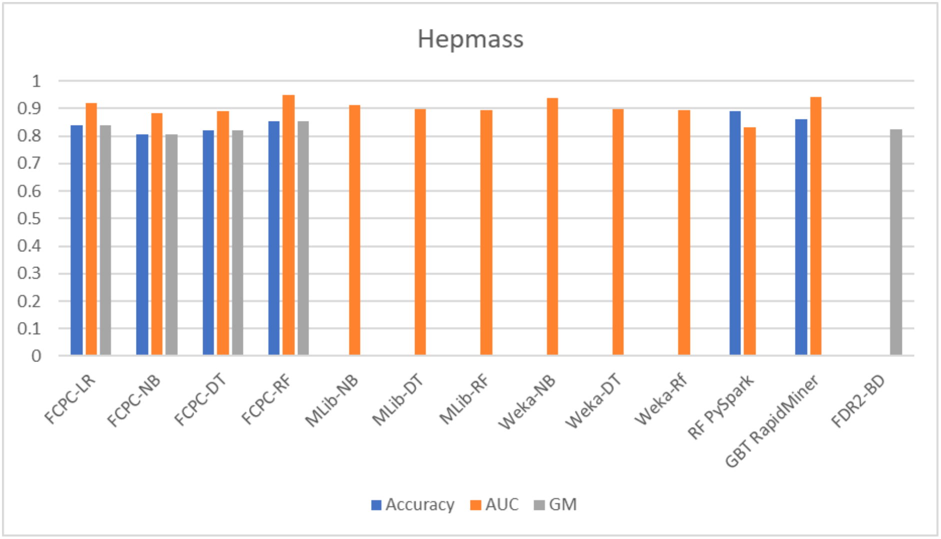

Figure 12. Accuracy, AUC, GM of FCPC vs State-of-art Techniques for Hepmass dataset.

sample reduction, where observation weights are determined by their distance from the cluster centroid. Parallel classifiers are applied to the subsamples, with their predictions integrated in the final phase. The experimental results demonstrated that FCPC outperformed traditional classifiers by reducing both computational time and data size by up to 90%, while maintaining or improving classification accuracy. This improvement is attributed to the use of observation similarities, fuzzification, and soft cluster-based subsampling, which minimizes information loss and enhances model robustness. A comparative analysis with several state-of-the-art methods highlighted FCPC's competitive edge. When compared to techniques using frameworks like MLlib and Weka, FCPC showed competitive or superior performance across benchmark datasets. Although it did not always achieve the highest AUC in every scenario, FCPC's robust performance across various classifiers and datasets underlines its potential as an alternative for Big Data classification tasks. In comparisons with methods like Boosted DTs, Neural Networks, and other fuzzy approaches, FCPC's RF classifier consistently performed well, particularly on large-scale datasets, demonstrating its scalability and computational efficiency. In the case study of the Gas Sensor

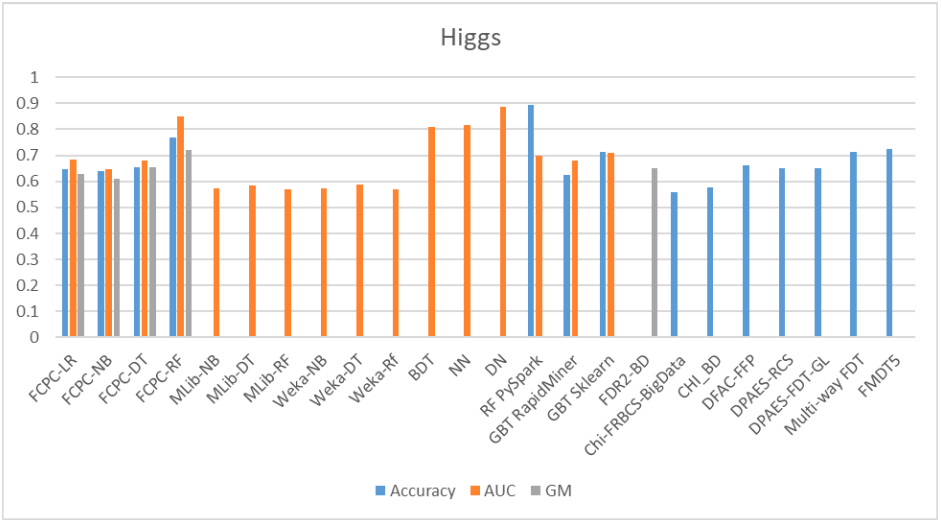

Figure 13. Accuracy, AUC, GM of FCPC vs State-of-art Techniques for Higgs dataset.

Array Temperature Modulation dataset, we explored the effect of varying the number of clusters K on FCPC's performance. $K = 3$ provided the best balance for simpler classifiers like LR and NB, while ensemble methods like RF showed robustness across all cluster configurations. Additionally, the analysis of the number of target categories demonstrated FCPC's adaptability and robustness, as it outperformed ordinary classifiers in both binary and categorical target scenarios. Looking ahead, future research could extend FCPC to regression tasks, offering a solution for large-scale regression problems. Investigating the optimal number of clusters and categories, along with exploring other ensemble learning methods, could further refine the methodology's performance. Additionally, plans to explore the potential of FCPC in anomaly detection tasks remain a key direction for future work. Further experimentation with different datasets and classification models will provide deeper insights into the full potential of FCPC in Big Data analytics.

Acknowledgement

This work was supported by computational resources provided by the Bibliotheca Alexandrina (BA) on its High-Performance Computing (HPC) infrastructure and HPC team.

REFERENCES

1. Y.K. Alapati, and K. Sindhu, *Combining clustering with classification: a technique to improve classification accuracy*, Lung Cancer, vol. 32, no. 57, p. 3, 2016.
2. F. Angiulli, *Fast nearest neighbor condensation for large data sets classification*, IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 11, pp. 1450–1464, 2007.
3. M. Assefi, E. Behravesh, G. Liu, and A. P. Tafti, *Big data machine learning using Apache Spark MLlib*, in *2017 IEEE International Conference on Big Data (Big Data)*, pp. 3492–3498, Dec. 2017.
4. P. Baldi, P. Sadowski, and D. Whiteson, *Searching for exotic particles in high-energy physics with deep learning*, *Nature Communications*, vol. 5, no. 1, p. 4308, 2014.
5. M.J. Basgall, M. Naiouf, and A. Fernández, *FDR2-BD: A fast data reduction recommendation tool for tabular big data classification problems*, Electronics, vol. 10, no. 15, p. 1757, 2021.
6. P. Berkhin, *A survey of clustering data mining techniques*, in Grouping Multidimensional Data, Springer, Berlin, Heidelberg, pp. 25–71, 2006.
7. A. Bhadani, and D. Jothimani, *Big data: Challenges, opportunities and realities*, in Effective Big Data Management and Opportunities for Implementation, M.K. Singh, and D.G. Kumar, Eds., IGI Global, Pennsylvania, USA, pp. 1–24, 2016.

8. C. Borgelt, G.G. Rodríguez, W. Trutschnig, M.A. Lubiano, M.A. Gil, P. Grzegorzewski, and O. Hryniewicz, Eds., *Combining Soft Computing and Statistical Methods in Data Analysis*, vol. 77, Springer Science & Business Media, 2010.
9. S. Cui, Y. Wang, Y. Yin, T.C.E. Cheng, D. Wang, and M. Zhai, *A cluster-based intelligence ensemble learning method for classification problems*, Information Sciences, vol. 560, pp. 386–409, 2021.
10. K. Djouzi, K. Beghdad-Bey, and A. Amamra, *A new adaptive sampling algorithm for big data classification*, Journal of Computational Science, vol. 61, p. 101653, 2022.
11. P. Ducange, M. Fazzolari, and F. Marcelloni, *An overview of recent distributed algorithms for learning fuzzy models in Big Data classification*, Journal of Big Data, vol. 7, no. 1, p. 19, 2020.
12. P.S. Efraimidis, and P.G. Spirakis, *Weighted random sampling with a reservoir*, Information Processing Letters, vol. 97, no. 5, pp. 181–185, 2006.
13. R. Evans, B. Pfahringer, and G. Holmes, *Clustering for classification*, in 2011 7th International Conference on Information Technology in Asia, IEEE, pp. 1–8, 2011.
14. J. Fan, F. Han, and H. Liu, *Challenges of big data analysis*, National Science Review, vol. 1, no. 2, pp. 293–314, 2014.
15. C. García-Osorio, A. de Haro-García, and N. García-Pedrajas, *Democratic instance selection: a linear complexity instance selection algorithm based on classifier ensemble concepts*, Artificial Intelligence, vol. 174, no. 5–6, pp. 410–441, 2010.
16. A. Gandomi, and M. Haider, *Beyond the hype: Big data concepts, methods, and analytics*, International Journal of Information Management, vol. 35, no. 2, pp. 137–144, 2015.
17. A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*, O'Reilly Media, Inc., 2022.
18. B. B. Gupta, D. P. Agrawal, S. Yamaguchi, and M. Sheng, *Soft computing techniques for big data and cloud computing*, Soft Computing, vol. 24, no. 12, pp. 5483–5484, 2020.
19. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.
20. Q. He, H. Wang, F. Zhuang, T. Shang, and Z. Shi, *Parallel sampling from big data with uncertainty distribution*, Fuzzy Sets and Systems, vol. 258, pp. 117–133, 2015.
21. D. Ibrahim, *An overview of soft computing*, Procedia Computer Science, vol. 102, pp. 34–38, 2016.
22. M. Junaid, S. Ali, I. F. Siddiqui, C. Nam, N. M. F. Qureshi, J. Kim, and D. R. Shin, *Performance evaluation of data-driven intelligent algorithms for big data ecosystem*, *Wireless Personal Communications*, vol. 126, no. 3, pp. 2403–2423, 2022.
23. A. Jurek, Y. Bi, S. Wu, and C. Nugent, *A survey of commonly used ensemble-based classification techniques*, The Knowledge Engineering Review, vol. 29, no. 5, pp. 551–581, 2014.
24. H. Kadkhodaei, A.M.E. Moghadam, and M. Dehghan, *Big Data Classification Using Heterogeneous Ensemble Classifiers in Apache Spark Based on MapReduce Paradigm*, Journal of Big Data, **10**(1), pp. 1–24, 2023.
25. L. Kaufman, and P.J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, New York, 1990.
26. S.S. Khan, S. Ahamed, M. Jannat, S. Shatabda, and D.M. Farid, *Classification by clustering (CBC): An approach of classifying big data based on similarities*, in Proceedings of International Joint Conference on Computational Intelligence: IJCCI 2018, Springer, pp. 593–605, 2020.
27. R. Kitchin and G. McArdle, *What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets*, *Big Data & Society*, vol. 3, no. 1, 2016.
28. Y. Li, X. Deng, S. Ba, W. R. Myers, W. A. Brenneman, S. J. Lange, R. Zink, and R. Jin, *Cluster-based data filtering for manufacturing big data systems*, Journal of Quality Technology, vol. 54, no. 3, pp. 290–302, 2022.
29. S. Liu, J. McGree, Z. Ge, and Y. Xie, *Computational and statistical methods for analysing big data with applications*, Academic Press, 2015.
30. J. Luengo, D. García-Gil, S. Ramírez-Gallego, S. García, and F. Herrera, *Big Data Preprocessing*, Springer, Cham, 2020.
31. M. S. Mahmud, J. Z. Huang, S. Salloum, T. Z. Emara, and K. Sadatdiynov, *A survey of data partitioning and sampling methods to support big data analysis*, Big Data Mining and Analytics, vol. 3, no. 2, pp. 85–101, 2020.
32. S. S. Moawad, M. Osman, and A. S. Moussa, *Computationally intelligent classification using fuzzified clustering for vaguely overlapping groups of uncertain data*, submitted, under review.
33. Y. Mu, X. Liu, L. Wang, and J. Zhou, *A parallel fuzzy rule-base based decision tree in the framework of map-reduce*, Pattern Recognition, vol. 103, p. 107326, 2020.
34. P.K.D. Pramanik, S. Pal, M. Mukhopadhyay, and S.P. Singh, *Big Data classification: techniques and tools*, in Applications of Big Data in Healthcare, Academic Press, pp. 1–43, 2021.
35. S. Saha, P.S. Sarker, A. Al Saud, S. Shatabda, and M.H. Newton, *Cluster-oriented instance selection for classification problems*, Information Sciences, vol. 602, pp. 143–158, 2022.
36. L. Serrano, *Grokking Machine Learning*, Simon and Schuster, 2021.
37. J. Singh, and V. Singla, *Big data: Tools and technologies in big data*, International Journal of Computer Applications, vol. 112, no. 15, 2015.
38. V. Singh, R.K. Gupta, R.K. Sevakula, and N.K. Verma, *Comparative analysis of Gaussian mixture model, logistic regression and random forest for big data classification using map reduce*, in 2016 11th International Conference on Industrial and Information Systems (ICIIS), IEEE, pp. 333–338, 2016.
39. A. Shukla, R. Tiwari, and R. Kala, *Real life applications of soft computing*, CRC Press, 2010.
40. A. Struyf, M. Hubert, and P. Rousseeuw, *Clustering in an object-oriented environment*, Journal of Statistical Software, vol. 1, pp. 1–30, 1997.
41. T. Tang, S. Chen, M. Zhao, W. Huang, and J. Luo, *Very large-scale data classification based on K-means clustering and multi-kernel SVM*, Soft Computing, vol. 23, no. 11, pp. 3793–3801, 2019.
42. T. Trinh, H. Le, N. VuongThi, H. HoangDuc, and K. VuThi, *A novel ensemble-based paradigm to process large-scale data*, Multimedia Tools and Applications, vol. 83, no. 9, pp. 26663–26685, 2024.
43. University of California, Irvine (UCI), *Machine Learning Repository*, Available at: http://archive.ics.uci.edu/, accessed on October 21, 2024.

44. H. Wang, *Divide-and-conquer information-based optimal subdata selection algorithm*, Journal of Statistical Theory and Practice, vol. 13, no. 3, p. 46, 2019.
45. H. Wang, Z. Xu, and W. Pedrycz, *An overview on the roles of fuzzy set techniques in big data processing: Trends, challenges and opportunities*, Knowledge-Based Systems, vol. 118, pp. 15–30, 2017.
46. W. Xing, and Y. Bei, *Medical health big data classification based on KNN classification algorithm*, IEEE Access, vol. 8, pp. 28808–28819, 2019.
47. J. Zhai, M. Wang, and S. Zhang, *Binary imbalanced big data classification based on fuzzy data reduction and classifier fusion*, Soft Computing, vol. 26, no. 6, pp. 2781–2792, 2022.